Lehrstuhl für Technische Informatik — Rechnernetze

Technische Universität München

CORBA-basiertes Enterprise Management: Interoperabilität und Managementinstrumentierung verteilter kooperativer Managementsysteme in heterogener Umgebung

Alexander Keller

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Chr. Zenger
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. H.-G. Hegering
	2.	Univ.-Prof. Dr. E. Jessen

Die Dissertation wurde am 30.10.1998 bei der Technischen Universtität München eingereicht und durch die Fakultät für Informatik am 9.12.1998 angenommen.

Danksagung

Die vorliegende Arbeit begann während meiner Mitarbeit im Projekt „Systemmanagement in heterogenen Systemen auf der Basis von CORBA“, das im Rahmen einer Forschungskooperation der IBM Deutschland Informationssysteme GmbH mit dem Institut für Informatik der Ludwig-Maximilians-Universität München gefördert wurde. Mein besonderer Dank gilt meinem Doktorvater, Herrn Prof. Dr. Heinz-Gerd Hegering, der mir bei der Erstellung dieser Arbeit eine ausgesprochen konstruktive und intensive Betreuung zuteil werden ließ und zahlreiche Stunden für die Lektüre früherer Fassungen der vorliegenden Arbeit aufgewendet hat. Herzlich bedanken möchte ich mich ebenfalls bei Herrn Prof. Dr. Eike Jessen, dessen kritische Anregungen mir bei der Erhöhung der Qualität dieser Arbeit sehr geholfen haben. Ganz besonderen Dank möchte ich außerdem meinen Kollegen Dr. Bernhard Neumair und Stephen Heilbronner sowie Herrn Dr. Ulf Hollberg vom IBM ENC aussprechen, deren stete Diskussionsbereitschaft und Motivation wesentlich zum Gelingen dieser Arbeit beigetragen hat. Besonderer Dank gebührt aber auch allen anderen Kolleginnen und Kollegen des MNM-Teams, die mir ein hervorragendes Arbeitsklima geschaffen haben, das mich auf meinem Weg sehr motiviert hat. Dank ebenfalls an diejenigen Studierenden, die im Rahmen ihrer Diplom- oder Praktikumsarbeiten nicht nur zum Gelingen dieses Werkes beigetragen, sondern auch die Arbeit des MNM-Teams unterstützt haben. Nicht zuletzt danke ich meiner Ehefrau Charlotte für ihre Geduld, ihre Liebe und ihre Unterstützung in jeglicher Hinsicht, sowie meiner Mutter Barbara und meinem Bruder Stefan, die mir immer wieder mit guten Ratschlägen zur Seite gestanden haben.

München, Oktober 1998

Zusammenfassung

Heutige umfangreiche DV-Infrastrukturen sind nur noch dann mit vertretbarem Aufwand administrierbar, wenn integrierte Managementlösungen eingesetzt werden. In letzter Zeit trat auf dem Weg zum integrierten Management allerdings eine zusätzliche Komplikation auf: Neben proprietären Ansätzen wurden mehrere Managementarchitekturen standardisiert, die teilweise in Konkurrenz zueinander stehen. Zur Heterogenität der zu administrierenden Anwendungen, Systeme und Komponenten kommt somit die Heterogenität des Managements hinzu. Gelöst werden kann diese Problematik nur durch die Schaffung von Übergängen zwischen einzelnen Managementsystemen, um die Interoperabilität und Kooperation unterschiedlicher Managementarchitekturen sicherzustellen.

Der dieser Arbeit zugrundeliegende Ansatz, diesen neuen Herausforderungen an das Management zu begegnen, ist die Einführung eines umfassenden, systemübergreifenden und insbesondere auf die Ziele des Betreibers fokussierenden **Enterprise Managements**, dessen Aufgabe darin besteht, die Vielzahl an unterschiedlichen Managementsystemen in ein einheitliches Rahmenwerk zu integrieren. Von zentraler Bedeutung sind hierbei Management-Gateways, die als Spezialfall von Managementsystemen angesehen werden können und – neben der Überwachung und Steuerung der von ihnen verwalteten Ressourcen – sowohl eine Transformation der Managementinformation als auch die Umsetzung der Managementprotokolle vornehmen. Diese im Rahmen der vorliegenden Arbeit entworfenen und prototypisch implementierten Systeme schaffen damit die Voraussetzungen, um Managementanforderungen und Zielvorgaben von Betreibern auf einheitliche Weise zu definieren und anzuwenden sowie deren Durchsetzung zu überwachen.

Naturgemäß handelt es sich bei Managementsystemen und Management-Gateways um verteilte Anwendungen, die wiederum ihrerseits administriert werden müssen. In dieser Arbeit wird daher eine konstruktive Methodik zur Ermittlung der notwendigen Managementinformation und Managementdienste vorgestellt, um Managementsysteme und Management-Gateways in heterogener Umgebung effektiv und effizient überwachen und steuern zu können. Sowohl die Analyse der Anforderungen als auch das Design und die Spezifikation der Systeme erfolgen dabei anhand objektorientierter Modellierungstechniken und unter Einsatz von CASE-Tools; die hierbei entstandenen Objektmodelle werden anschließend auf CORBA abgebildet, eine Architektur für verteilte objektorientierte Programmierung, deren Eignung für das integrierte Management durch die vorliegende Arbeit nachgewiesen wird.

INHALT

KAPITEL 1

Einleitung

1.1 Enterprise Management als strategischer Faktor

Die Implikationen des Client/Server Computing, das kontinuierliche Zusammenwachsen der Daten- und Telekommunikation in Verbindung mit der Verfügbarkeit hoher Bandbreiten zu einem akzeptablen Preis, die Vereinfachung der Nutzung integrierter Dienste (Telefonie, Electronic Mail, Datentransfer, Multimedia) sowie die Kostensenkungspotentiale aufgrund der Liberalisierung der Telekommunikationsmärkte bilden die Fundamente für die fortschreitende Vernetzung von Unternehmen. Auswirkungen hiervon, wie die Einbeziehung einer stetig steigenden Zahl von Mitarbeitern in DV-technisch unterstützte Geschäftsprozesse, die Vernetzung mit vor- und nachgelagerten Instanzen in der Wertschöpfungskette sowie die Nutzung digitaler Medien für Marketing, Vertrieb und Support von Waren und Dienstleistungen bringt jedoch ebenfalls gestiegenen Administrationsbedarf sowie die Zunahme potentieller Fehlermöglichkeiten mit sich.

Gefordert sind daher integrierte und auf die Gesamtheit der Unternehmens-DV abzielende (Enterprise-) Managementkonzepte, die auf effiziente und ressourcenschonende Art den Netz- und Systemadministratoren komprimierte und aussagekräftige Infomationen über den Zustand der Kommunikationsinfrastruktur, der daran angeschlossenen Systeme sowie der darauf ablaufenden (ggf. verteilten) Anwendungen liefern. Obwohl die Problematik bereits seit geraumer Zeit erkannt ist, werden gegenwärtig erhältliche Managementsysteme diesen Anforderungen häufig in nur unzureichender Weise gerecht.

Dieser Zustand resultiert nicht zuletzt daraus, daß neben den bisher bekannten Einflußfaktoren, wie die hohe Komplexität und Heterogenität der zu administrierenden Systeme sowie deren räumliche Verteilung und verhältnismäßig kurze Produktlebenszyklen, in jüngster Zeit zwei weitere Einflußgrößen stetig an Bedeutung zunehmen:

Dies ist zum einen die Notwendigkeit der Zusammenarbeit unterschiedlicher Organisationen, also die **kooperative Erbringung von Managementdienstleistungen**. Ein aktuelles Beispiel hierfür liefert die Telekommunikationsbranche: Aufgrund der bis vor kurzem bestehenden Monopolsituation konnte der Telekommunikations-Diensterbringer nicht zur Offenlegung von Managementinformation verpflichtet werden, um den Dienstnutzern Anhaltspunkte zur Beurteilung der bereitgestellten Dienstgüte zu liefern. Die Liberali-

sierung auf dem Telekommunikationssektor eröffnet den Betreibern großer Netze nicht nur die Möglichkeit, zwischen unterschiedlichen Anbietern von Telekommunikationsdiensten zu wählen, sondern mit diesen auch Dienstgütevereinbarungen abzuschließen. Dies bedingt einerseits die Offenlegung von Dienstgüteparametern durch den Diensterbringer, andererseits jedoch die Notwendigkeit, diese Informationen auf Dienstnutzerseite verarbeiten zu können. Es ist daher erforderlich, die Interoperabilität der Managementsysteme von Dienstnutzer und Diensterbringer sicherzustellen. Die Tatsache, daß Dienstnutzer/Diensterbringer-Beziehungen auf unterschiedlichen Schichten eines Kommunikationssystems bestehen, erfordert die Kooperation von Managementsystemen, die ihrerseits verschiedene Ebenen des gesamten Rechnernetzes managen. Hieraus ergibt sich insbesondere die Notwendigkeit, auch **Managementsysteme überwachen und steuern zu können**; dieser Problematik ist bisher von der Fachwelt lediglich geringe Aufmerksamkeit gewidmet geworden. Unabdingbare Voraussetzungen für die Überwachung und Steuerung von Managementsystemen ist das Vorhandensein offengelegter Beschreibungen der Management*information* von Managementsystemen sowie die Verfügbarkeit hierfür geeigneter Management*dienste*. Auch hierzu sind bisher keine einschlägigen Aktivitäten bekanntgeworden.

Die zweite Einflußgröße resultiert aus der drastischen Zunahme der Standardisierungsorganisationen, Industriekonsortien und Hersteller, die de-iure und de-facto Standards für das Management offener Systeme auf dem Markt zu etablieren versuchen. Eine (unvollständige) Aufzählung umfaßt gegenwärtig folgende Vertreter: ISO-ITU, IETF, OMG, DMTF, Opengroup, NM Forum, Sun, Microsoft. Die Tatsache, daß diese von unterschiedlichen Gremien etablierten Managementstandards sich nur in Ausnahmefällen ergänzen und größtenteils in Konkurrenz zueinander stehen, stellt für die Betreiber verteilter Systeme und Rechnernetze bei der Integration neuer Systeme in die bereits bestehende Infrastruktur ein signifikantes Problem dar. Zur Heterogenität der zu administrierenden Anwendungen, Systeme und Komponenten kommt somit die **Heterogenität des Managements** hinzu. Gelöst werden kann diese Problematik nur durch die Schaffung von Übergängen zwischen einzelnen Managementsystemen, um eine **Interoperabilität und Kooperation unterschiedlicher Managementarchitekturen** sicherzustellen. Hier sind in jüngster Zeit einige Arbeiten in Entwicklung; von einer umfassenden Lösung des Problems ist man jedoch auch hier noch weit entfernt.

Ein vielversprechender Ansatz, diesen neuen Herausforderungen an das Management zu begegnen, ist die Einführung eines umfassenden, systemübergreifenden und insbesondere auf die Ziele des Betreibers ausgerichteten **Enterprise Managements**, dessen Aufgabe darin besteht, die Vielzahl an unterschiedlichen Managementsystemen in ein einheitliches Rahmenwerk zu integrieren. Es schafft damit die Voraussetzungen, um Managementanforderungen und Zielvorgaben von Betreibern auf einheitliche Weise zu definieren und anzuwenden, sowie deren Durchsetzung zu überwachen.

Solange jedoch die technischen Gegebenheiten zur Kooperation von Managementsystemen nicht abschließend geklärt sind, scheitert die Anwendung und Durchsetzung von Ziel-

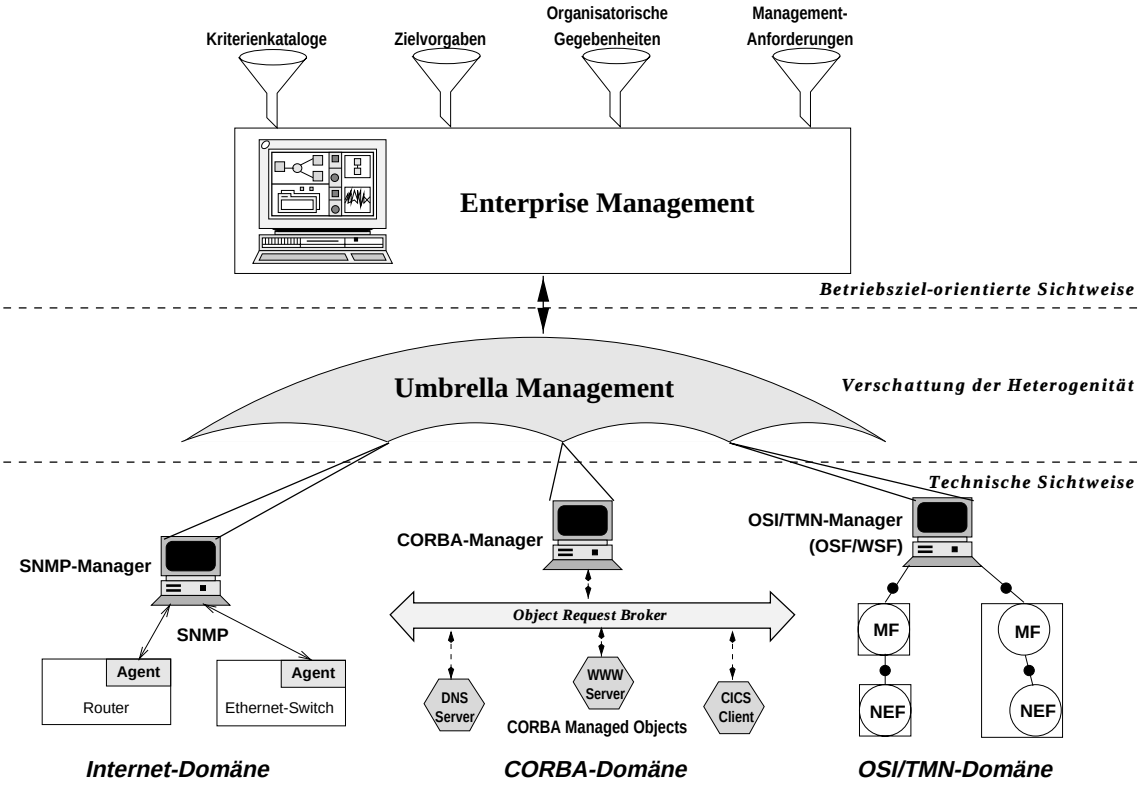


Abbildung 1.1: Einordnung des Enterprise- und Umbrella Managements

vorgaben an deren Heterogenität. Die Sicherstellung der Kooperation bedingt die Zusammenfassung der zahlreichen bei den Betreibern vorhandenen Management(teil)systeme unter einen „Managementschirm“ für unternehmensweites Management, damit diese geeignet koordiniert werden können (vgl. hierzu auch [HeAb 93]). Dieses sogenannte **Umbrella Management** fokussiert auf die *technischen Aspekte* zur Sicherstellung der Kooperation von Managementsystemen. Es befaßt sich insbesondere mit der Bildung von Übergängen zwischen heterogenen Managementarchitekturen sowie der Bereitstellung managementrelevanter Datenbestände zur Auswertung durch verteilte und möglicherweise heterogene Managementsysteme. Umbrella Management bildet durch die Abstraktion von einzelnen Architekturen die technische Grundlage für das betriebszielorientierte Enterprise Management durch die Schaffung von Interaktionsmöglichkeiten zwischen **verteilten**, nunmehr **kooperativen Managementsystemen**. Abbildung 1.1 stellt diesen Zusammenhang graphisch dar.

Die obigen Ausführungen verdeutlichen, daß die Problematik eines **architekturübergreifenden Managements verteilter kooperativer Managementsysteme** von fundamentaler Bedeutung für das Enterprise Management ist, bisher jedoch noch nicht hinreichend behandelt wurde. Die vorliegende Arbeit stellt einen Lösungsansatz auf der Grundlage der *Common Object Request Broker Architecture (CORBA)*[1] für diese Problematik vor.

[1] Eine detaillierte Beschreibung von CORBA sowie die Aspekte der Nutzung von CORBA für integriertes

1.2 Aufgabenstellung

Mit der sprunghaften Zunahme der Anforderungen an die Dienstgüte (Quality of Service, QoS) der zu überwachenden und steuernden Systeme und Anwendungen ist Management großer verteilter Systeme oder Corporate Networks von einer zentralisierten Managementplattform aus nicht mehr effizient machbar. Ein erster Schritt zur Behandlung dieses Problems ist die Einführung mehrerer Managementplattformen, die ihrerseits für einen abgegrenzten Teilbereich des Rechnernetzes bzw. verteilten Systems verantwortlich sind.

Aus dem Vorhandensein zahlreicher isolierter Managementplattformen ergibt sich allerdings ein hoher Koordinierungs- und Steuerungsaufwand dieser Systeme, der von gegenwärtig verfügbaren kommerziellen Produkten nur in unzureichender Weise gelöst wird, da momentan keine ausgereiften Konzepte zur Behandlung dieser Anforderungen zur Verfügung stehen.

Die Problematik des Managements von Managementsystemen erhält eine um so höhere Brisanz, als es sich bei diesen weniger um konkrete Ressourcen handelt, die bereits von ihrem Aufbau her Hinweise zur Modellierung geben, sondern vielmehr um Applikationen, die auf abstrakten Größen (Zähler, Warteschlangen, Prozesse, Logs, Systeme) aufbauen, welche managementrelevante Informationen von realen Ressourcen repräsentieren. Dies impliziert, daß zur Modellierung von Managementsystemen neue Typen von Beziehungen (wie z.B. Abstützungsbeziehungen zwischen Diensten und Prozessen) zwischen Managementobjekten eingeführt werden müssen, die in den gegenwärtigen Architekturen noch nicht vorhanden sind.

Eine zusätzliche Komplexität resultiert aus der Tatsache, daß Managementsysteme und die in ihnen enthaltene Managementfunktionalität auf mehrere heterogene Systeme verteilt sein kann. Diese Heterogenität bezieht sich nicht nur auf die Architektur der Rechner sowie deren Betriebs- und Übertragungssysteme, sondern auch auf die – unter Umständen unterschiedlichen – Managementarchitekturen, die den konzeptionellen Rahmen zur Überwachung und Steuerung dieser Systeme festlegen.

Es sind also managementarchitekturübergreifende Methoden gefordert, die einerseits offen für die Integration bereits bestehender Architekturen sind, aber andererseits die Modellierung komplexer Managementsysteme und deren Beziehungen zueinander effizient gestatten.

Gesucht wird also eine Methodik zur Überwachung und Steuerung verteilter kooperativer Managementsysteme in heterogenen Umgebungen. Sie basiert einerseits auf einem Objektmodell von Managementsystemen und andererseits auf einem Rahmenwerk zur Spezifikation und Implementierung geeigneter Managementdienste, das gestattet, auf der Grundlage standardisierter Verfahren und Architekturen die für den Problembereich „Management von Managementsystemen“ erforderliche Funktionalität dynamisch den Gegebenheiten anzupassen.

Die wichtigsten Fragestellungen bzw. Teilprobleme bestehen folglich in

Management befinden sich in Anhang A bzw. in Kapitel 3.

- der Strukturierung des Problembereichs hinsichtlich der Ausprägungsformen von Managementsystemen sowie der Analyse der Anforderungen an Managementsysteme;
- der Auswahl einer (Management-) Architektur, die es gestattet, den Umfang von Managementfunktionalität flexibel an neue Anforderungen anzupassen, um der Dynamik des Enterprise Managements gerecht zu werden;
- dem Entwurf neuer bzw. die Erweiterung bereits bestehender Verfahren, um die Interoperabilität mit Managementsystemen zu gewährleisten, die auf verschiedenartigen Architekturen beruhen;
- der Entwicklung eines Objektmodells für verteilte kooperative Managementsysteme;
- der Identifikation von Verfahren, die gestatten, das entwickelte Objektmodell möglichst ohne den Verlust an Semantik in die zugrundegelegte Managementarchitektur zu überführen;
- der Definition und Implementierung von Diensten, die für das Management von Managementsystemen unerläßlich sind.

Die vorliegende Arbeit hat zum Ziel, diese Fragestellungen unter der Randbedingung des Konfigurations- und Fehlermanagements zu lösen. Angesichts der Tatsache, daß das Management verteilter kooperativer Managementsysteme sowohl das Management der Applikation „Management“ als auch das Management der jeweiligen Systeminfrastruktur beinhaltet, stehen die Disziplinen „Anwendungsmanagement“ und „Systemmanagement“ im Zentrum der Betrachtung. Wie die Analyse bestehender standardisierter Architekturen, aktueller Forschungsansätze und am Markt erhältlicher Produkte in Kapitel 3 zeigt, ist diese Problematik in der hier vorgeschlagenen Form bislang noch nicht angegangen bzw. gelöst worden.

1.3 Vorgehensmodell und Ergebnisse dieser Arbeit

Der Aufbau und die Vorgehensweise dieser Arbeit folgen einem Top-down-Ansatz, der in Abbildung 1.2 am Ende dieses Kapitels dargestellt ist:

Kapitel 2 stellt neben der Definition relevanter Begriffe anhand mehrerer realer Szenarien charakteristische Teilaspekte der Fragestellung der vorliegenden Arbeit vor, die die derzeitige Situation beim Betrieb großer Kommunikationssysteme widerspiegeln. Hieraus werden Anforderungen an das Management verteilter kooperativer Managementsysteme abgeleitet. Diese grundlegenden Anforderungen haben nicht nur Auswirkungen auf das Kooperationsmodell und die Bildung von Managementdomänen sowie die Repräsentation der Managementinformation, sondern auch Konsequenzen für die Bereitstellung bzw. die Granularität der benötigten Managementfunktionalität.

Kapitel 3 analysiert bestehende Konzepte, standardisierte Architekturen und am Markt erhältliche Technologien und grenzt diese Arbeit gegen bereits bekannte Ansätze ab. Es werden diejenigen Möglichkeiten vorgestellt, die gegenwärtige, zum Teil in der Entwicklung befindliche, Architekturen für das Management verteilter kooperativer Managementsysteme bieten. Hierbei wird insbesondere die *Common Object Request Broker Architecture (CORBA)* einer genauen Betrachtung unterzogen, da sie sich bei unseren Untersuchungen als besonders geeignet für das Enterprise Management herausgestellt hat. Ebenso werden aktuelle Forschungsprojekte mit verwandter Themenstellung sowie kommerziell erhältliche Lösungen und Werkzeuge im Hinblick auf die zentralen Fragestellungen dieser Arbeit kritisch untersucht und bewertet.

Während bisher häufig davon ausgegangen wurde, daß die Unterteilung der zu administrierenden Ressourcen in Domänen hauptsächlich durch organisatorische oder geographische Gegebenheiten festgelegt wird, führt das vierte Kapitel ein weiteres, *technisches*, Kriterium zur Festlegung von Domänen ein. Es behandelt diejenigen Aspekte, die sich aus der Einführung einer neuen Architektur für das System- und Anwendungsmanagement in das heterogene Managementumfeld eines Netzbetreibers ergeben: Schließlich impliziert der Einsatz einer neuen Managementarchitektur häufig nicht, bestehende etablierte Architekturen vollständig zu ersetzen, da dies in der Praxis aus Gründen der Sicherung bestehender Investitionen nicht machbar ist; es müssen vielmehr Wege geschaffen werden, welche die Koexistenz und Kooperation von Systemen erlauben, die auf unterschiedliche Weise administriert werden. Kapitel 4 untersucht daher die technischen Konzepte der Kooperation von Managementarchitekturen (Umbrella Management), die die Basis bilden, um **Enterprise Management** überhaupt erst zu ermöglichen. Wir werden dabei anhand von uns implementierter Prototypen aufzeigen, welche Ansätze hierfür besonders geeignet sind. Es wird ebenfalls aufgezeigt werden, wie Managementdienste, die lediglich in einer Architektur vorhanden sind, auch auf Systeme angewandt werden können, deren zugrundeliegende Architekturen ursprünglich keine solche Funktionalität vorsehen. Dies ist wichtig, da der Umfang bereitgestellter Managementdienste zwischen einzelnen Architekturen erheblich variiert. Insgesamt müssen also zusätzlich zu den betreiberspezifischen Anforderungen auch technische Gegebenheiten in die Modellierung einfließen, um ein Objektmodell zu erhalten, das den Ansprüchen integrierten Managements genügt.

Beim Entwurf eines solchen Objektmodells treten zwei wichtige Aspekte auf: Es ist einerseits die Frage zu klären, welche Managementobjekte sowie die dazugehörigen Attribute und Operationen relevant sind. Dies wird im folgenden als der **Informationsaspekt** bezeichnet, da sich diese Angaben im wesentlichen auf das Informationsmodell des Systems beziehen. Von ebenso großer Wichtigkeit ist die Fragestellung, welche Dienste für das Management des betrachteten Systems erforderlich sind, also der **Funktionsaspekt**.

Ziel des fünften Kapitels ist die Bereitstellung aller wichtigen Parameter sowohl des Informationsaspektes als auch des Funktionsaspektes verteilter kooperativer Managementsysteme. Hierbei wird auf anerkannte und verbreitete objektorientierte Analyse– und Designtechniken wie OMT zurückgegriffen. Wir werden in diesem Zusammenhang zunächst

eine werkzeugunterstützte, konstruktive Methodik vorstellen, die es uns erlaubt, nahezu automatisch bereits bestehende Managementinformationsbeschreibungen in unsere Zielarchitektur zu überführen. Das zugrundeliegende Vorgehensmodell wird dabei anhand eines einfachen Beispiels aus dem Systemmanagement erläutert: Einem Agenten für das Management von UNIX-Workstations, der am Lehrstuhl entworfen und implementiert wurde. Da die Funktionsfähigkeit der verteilten Anwendung „Management“ maßgeblich von der Zuverlässigkeit der darunterliegenden Systeme abhängt, bildet das Systemmanagement die Grundlage für das Management verteilter kooperativer Managementsysteme. Anschließend stellen wir unsere auf einem standardisierten Referenzmodell für offene verteilte Verarbeitung basierende Entwurfsmethodik vor, anhand der die Managementinstrumentierung für verteilte kooperative Managementsysteme entwickelt wurde.

Gegenwärtige objektorientierte *Management Information Bases (MIBs)* (und damit auch Objektmodelle) sind häufig zwei substanziellen Kritikpunkten ausgesetzt:

1. Ihnen geht nahezu ausschließlich eine Bottom-up-Analyse voraus; sie beinhalten daher lediglich eine unabstrahierte 1:1-Abbildung von konkreten Ressourcenparametern wie Zählern, Pegeln und Bezeichnern in das jeweilige Informationsmodell. Eine Verdichtung mehrerer Ressourcenparameter zu aussagekräftigen Kennzahlen geschieht nicht. Die von den MIBs zur Verfügung gestellten Parameter decken sich daher häufig nicht mit den Daten, die von den Anwendern bzw. Netzbetreibern gewünscht werden.

2. Die Vererbungshierarchie in den MIBs ist generell ziemlich flach; dies bedeutet, daß der Grad an Wiederverwendbarkeit von Managementinformation sehr gering ist, da der überwiegende Teil dieser Information in ressourcenspezifischen Klassen definiert wird. Die generischen Basisklassen der Vererbungshierarchie enthalten daher nur äußerst wenige verwertbare Informationen.

Das in dieser Arbeit entworfene Objektmodell vermeidet diese beiden Nachteile, da zuerst konkrete Managementszenarien, die beim Betrieb verteilter kooperativer Managementsysteme auftreten, aufgestellt und klassifiziert werden. Die aus ihnen resultierenden Anforderungen an Managementparameter werden aus den Szenarien gewonnen und in die Modellierung aufgenommen. Diese Top-down-Analyse wird anschließend durch eine Analyse der vorhandenen Eingriffsmöglichkeiten (bottom-up) ergänzt, die gegenwärtige Systeme bieten. Sie ist notwendig, um einen hinreichenden Informationsgehalt der in der Top-down-Analyse identifizierten Objekte zu garantieren. Dem zweiten Kritikpunkt wird durch die Optimierung des Objektmodells begegnet, die unter anderem darauf abzielt, einen möglichst hohen Anteil an Managementinstrumentierung bereits in die Basisklassen der Vererbungshierarchie zu integrieren. Hierdurch wird erreicht, daß der Anteil ressourcenspezifischer Managementinformation zugunsten generischer und damit für eine große Ressourcenmenge gültiger Parameter sinkt; der Zugriff auf eine gewisse Anzahl von Informationen über eine neue Ressource ist somit möglich, ohne daß diese Daten einem

Managementsystem vorher explizit bekanntgegeben werden müssen[2].

Ferner befaßt sich Kapitel 5 mit dem Funktionsaspekt verteilter kooperativer Managementsysteme. Hierunter versteht man die Identifikation, Spezifikation, Modellierung und Implementierung derjenigen Dienste, die für das Management dieses spezifischen Anwendungsbereiches notwendig sind. Dabei kommt es darauf an, in heutigen standardisierten Managementarchitekturen bereits vorhandene Dienste zu nutzen und – auf diesen aufbauend – weitergehende Funktionalität zu definieren. Zu diesem Zweck werden die für unseren Anwendungsbereich relevanten Dienste aus unterschiedlichen Managementarchitekturen identifiziert und miteinander verglichen. Es wird gezeigt, wie Dienste, die in einer Managementarchitektur vorhanden sind, von einer anderen Architektur genutzt werden können. Dies ist notwendig, um die neu zu entwickelnde Menge der für das Management verteilter kooperativer Managementsysteme notwendigen Managementfunktionalität in akzeptablen Grenzen zu halten. Die Identifikation dieser Dienste geschieht anhand der in den Kapiteln 2 und 4 herausgearbeiteten anwendungsspezifischen bzw. technischen Anforderungen. Die Spezifikation und Modellierung dieser Dienste erfolgt analog zu dem bei der Konzeption des Objektmodells erfolgreich angewandten Verfahren. Hieraus ergibt sich insbesondere, daß die Vorteile der Verwendung einer vollständig objektorientierten Realisierungsarchitektur wie CORBA insbesondere darin liegen, daß sowohl der Informationsaspekt als auch der Funktionsaspekt einheitlich behandelt werden können, da sowohl die Managementobjekte, als auch die zu ihrem Management notwendigen Dienste in Form von Objektklassen modelliert und implementiert werden können.

Kapitel 6 stellt unsere Implementierung des in Kapitel 5 entworfenen Objektmodells sowie die dazugehörigen Managementdienste für das Management verteilter kooperativer Managementsysteme vor. Eine für die Skalierbarkeit einer Managementlösung unabdingbare Voraussetzung ist die Delegierbarkeit von Managementfunktionalität von Managementsystemen an untergeordnete Managementsysteme bzw. Agenten. Folglich wird aufgezeigt, wie Managementfunktionalität in einer CORBA-Umgebung delegiert werden kann; auch hier kommen die Vorteile der Nutzung von CORBA für das Management zum Tragen, da diese Architektur gute Mechanismen zur Delegierung allgemeiner Dienste bereitstellt. Diese werden in der vorliegenden Arbeit auf den Problembereich *Management* angewandt. Das Kapitel schließt mit einer Schilderung unserer Erfahrungen mit am Markt erhältlichen CORBA-Entwicklungssystemen.

Das Ziel dieser Arbeit ist es also, die Fragestellung zu beantworten, welche Management*information* und Management*dienste* vorhanden sein müssen, um Managementsysteme in *heterogener* Umgebung effektiv und effizient überwachen und steuern zu können.

Die wichtigsten Ergebnisse der vorliegenden Arbeit lauten wie folgt:

1. Die Eignung zahlreicher (Management-) Architekturen für ein betriebszielorientiertes Enterprise Management wurde eingehend analysiert und bewertet. Dabei konnte gezeigt werden, daß CORBA hierfür die gegenwärtig besten Voraussetzungen bietet.

[2] Im OSI-Informationsmodell wird dies als „Allomorphie“ bezeichnet.

2. Es wurde ein Rahmenwerk für das Umbrella Management definiert, das Mechanismen zur Sicherstellung der Interoperabilität in heterogenen Umgebungen beinhaltet. Die unterschiedlichen Varianten solcher Mechanismen (multiarchitektureller Manager, Management-Gateway, multiarchitektureller Agent) wurden von uns implementiert und auf ihre Einsetzbarkeit hin überprüft und entsprechend bewertet. Hierbei konnten wichtige Erkenntnisse hinsichtlich ihrer praktischen Anwendbarkeit, Effizienz und Skalierbarkeit gewonnen werden.

3. Anhand unserer prototypischen Implementierungen konnte ebenfalls nachgewiesen werden, daß es aussichtsreich und mit heutigen technischen Mitteln machbar ist, nahtlose Übergänge zwischen unterschiedlichen Managementarchitekturen zu schaffen. Die weitverbreitete Annahme, wonach die Interoperabilität von Managementarchitekturen stets nach dem Prinzip des „kleinsten gemeinsamen Nenners“ erfolgt, konnte widerlegt werden.

4. Es wurde ein werkzeugunterstützter, universell anwendbarer Ansatz entwickelt, der die nahezu vollständig automatisierte Gewinnung CORBA-konformer Managementagenten aus bestehenden modularen Implementierungen gestattet, denen andere Managementarchitekturen zugrundeliegen.

5. Wir haben eine neue Methodik entwickelt, die es erlaubt, sowohl die Managementsysteme als auch die zu ihrem Management erforderlichen Dienste auf einheitliche Weise mit Hilfe objektorientierter Analyse- und Designmethoden zu spezifizieren. Durch die Verwendung standardisierter Konzepte für offene verteilte Anwendungen wurde erreicht, daß ein Maximum an generischer Managementinformation bereits nahe an der Wurzel der Vererbungshierarchie definiert wird.

6. Die Erkenntnis, daß CORBA nachweislich (d.h. durch unsere Prototypen) ein solides Fundament für integriertes Enterprise Management bietet; um im realen Betrieb eingesetzt werden zu können, müssen jedoch noch einige spezifische Managementdienste standardisiert werden. Hierfür wurden dem Problembereich dieser Arbeit entsprechende Dienste spezifiziert und implementiert, die ein umfassendes Management verteilter kooperativer Managementsysteme gewährleisten.

7. Diese spezifischen Managementdienste bestehen zu einem gewissen Teil bereits in anderen Architekturen; folglich wurde in dieser Arbeit ein Verfahren entwickelt, um diese „fremden“ Dienste in die OMG–Referenzarchitektur einzubetten.

8. Es konnte nachgewiesen werden, daß durch die Verwendung von „Inband Management“ und eines vollständig objektorientierten Entwicklungsprozesses die Notwendigkeit spezieller Managementprotokolle und -modelle entfällt. Hieraus ergeben sich Vereinfachungen bei der Implementierung von Managementlösungen, da allgemein verfügbare Designmethoden (wie OMT und UML) und Technologien (wie z.B. CASE-Tools) erfolgreich für das Management genutzt werden können.

9. Bisher ist trotz der Standardisierung einer Vielzahl von MIB-Modulen für unterschiedlichste Ressourcenklassen noch kein Vorschlag für eine Managementschnittstelle von Managementsystemen erfolgt. Die vorliegende Arbeit leistet dies, indem Vorgaben für den notwendigen Umfang an von Managementsystemen bereitzustellender Managementinstrumentierung gemacht werden. Dem mit der Abhängigkeit der Managementapplikationen von konkreten Managementplattformen einhergehenden Verlust der Offenheit und Portabilität kann so gezielt begegnet werden.
10. Schließlich haben wir uns mit den Aspekten dynamischer Delegierung von Managementdiensten befaßt mit dem Ziel, verteilte kooperative Managementsysteme nach Bedarf mit geeigneter Managementfunktionalität auszustatten.

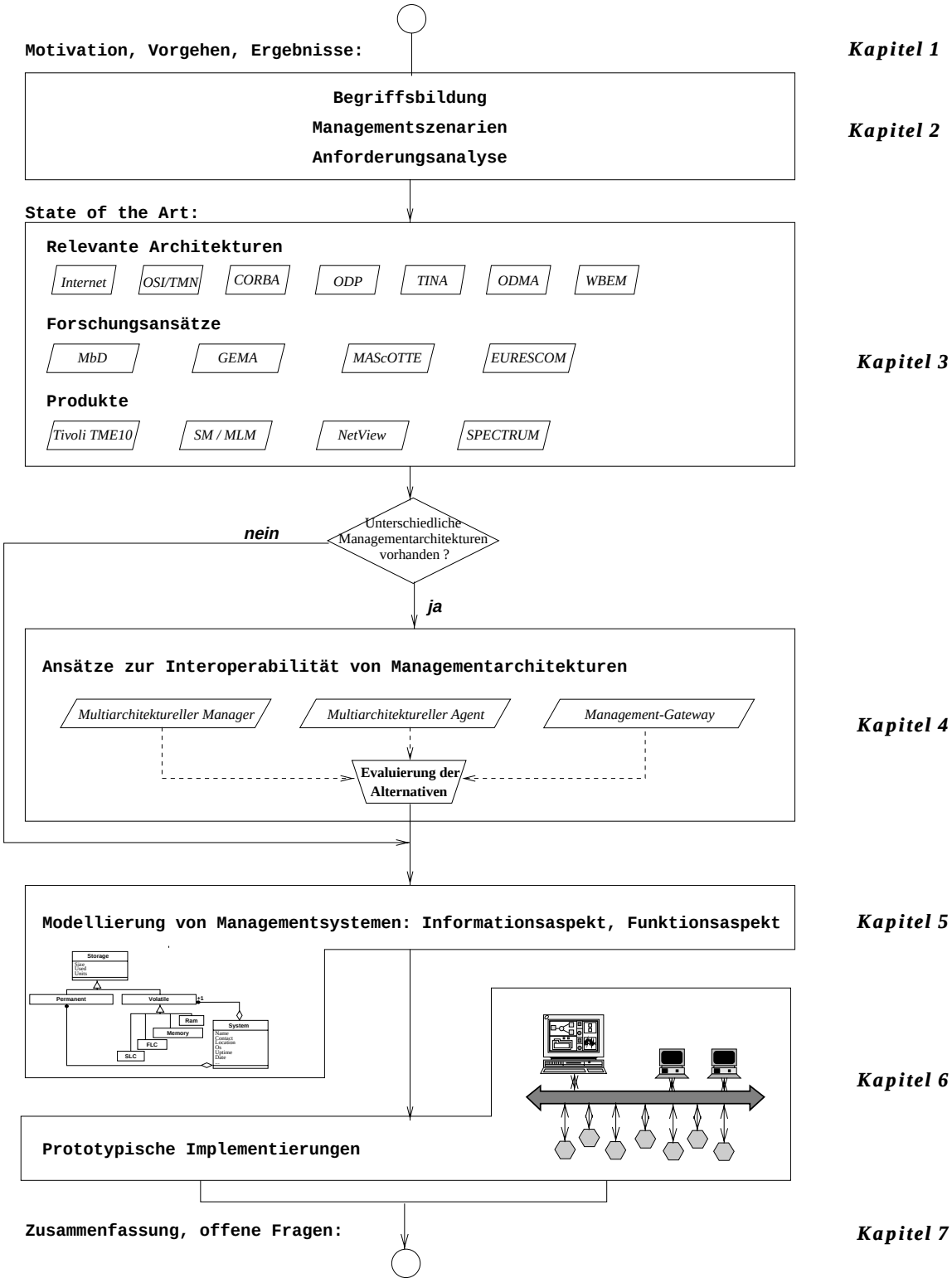


Abbildung 1.2: Vorgehensmodell dieser Arbeit

KAPITEL 2

Analyse der Anforderungen

Das gegenwärtige frühe Stadium des unternehmensweiten integrierten Managements führt dazu, daß die hierbei benutzen Begriffsdefinitionen häufig inkonsistent, mehrdeutig und oft sogar widersprüchlich sind. Wir werden daher im ersten Teil dieses Kapitels die für diese Arbeit relevanten Begriffe einführen und so den begrifflichen Kontext für die weiteren Kapitel festlegen. Um die Tragweite des im Rahmen dieser Arbeit behandelten Problembereichs abschätzen zu können, werden wir in Abschnitt 2.2 einige Szenarien sowohl aus dem Daten- als auch aus dem Telekommunikationsumfeld vorstellen. Sie verdeutlichen einerseits die Praxisrelevanz der in dieser Arbeit behandelten Fragestellung. Andererseits ergeben sich aus diesen Szenarien Anforderungen an das Management verteilter kooperativer Managementsysteme, welche in Abschnitt 2.3 herausgearbeitet werden und ihrerseits die Grundlage für die im weiteren Verlauf zu erstellenden Objektmodelle sowie die erforderlichen Managementdienste bilden. Ebenso lassen sich aus den Managementszenarien Bewertungskriterien ableiten, auf die wir uns im weiteren bei der Auswahl einer für unsere Zwecke geeigneten Managementarchitektur abstützen können.

2.1 Begriffsbildung

Die strategische Rolle des Managements für die Funktionsfähigkeit großer Kommunikationssysteme läßt sich neben zahlreichen anderen Kennzahlen auch an der mittlerweile sehr hohen Anzahl kommerzieller Managementlösungen ablesen. Zur Vermarktung dieser Produkte benutzen Hersteller häufig Begriffe wie „integrierte Managementplattformen“ und „offene Enterprise Management Systeme“ als Synonyme, die einerseits meistens nicht die tatsächlichen Eigenschaften der Produkte widerspiegeln. Andererseits sind diese Begriffe nicht immer mit einer eindeutigen Semantik belegt, die wir jedoch im Laufe dieser Arbeit benötigen, um beispielweise im 3. Kapitel bestehende Ansätze und Produkte bewerten und kategorisieren zu können. Wir werden daher in diesem Abschnitt die für diese Arbeit wichtigsten Begriffsdefinitionen einführen, um ein einheitliches begriffliches Rahmenwerk für die vorliegende Arbeit zu schaffen.

2.1.1 Integriertes Management

Einige Faktoren der Bedeutungszunahme integrierten Managements für Client/Server-basierte kooperative IV-Versorgungsstrukturen wurden bereits in der Einführung zu dieser Arbeit identifiziert. Es ist deutlich erkennbar, daß mit zunehmender Verbreitung offener Systeme und leistungsfähiger Kommunikationsnetze die Kooperation verteilter und heterogener Hard- und Softwarekomponenten eine immer größere Rolle einnimmt. Der Preis dafür ist ein komplexeres technisches Management dieser Systeme, das in heterogener Umgebung nur auf der Basis standardisierter Architekturen effizient zu bewältigen ist. Insbesondere kann man zukünftig nicht mehr wie bisher streng zwischen der Administration des Kommunikationsnetzes (Netzmanagement) und der Administration der Endsysteme (Systemmanagement) und Anwendungen (Anwendungsmanagement) unterscheiden. Man muß vielmehr das Management aller Komponenten dieser Umgebungen zu einem **integrierten Management** der DV-Infrastruktur zusammenfassen.

Verteilte Systemumgebungen sowie die in ihnen enthaltenen Komponenten unterscheiden sich stark bezüglich ihres Aufbaus, ihrer Größe oder ihrer Ausrichtung. Es kann folglich auch nicht eine einzige Managementlösung für alle verteilten Systeme geben. Ziel muß es vielmehr sein, einen „Baukasten“ von Modulen zu entwerfen, in dem die Teile, die einzelne Problembereiche bearbeiten, so flexibel wie möglich zusammengesetzt werden können, um für jede Umgebung zu einer optimalen Managementlösung zu kommen.

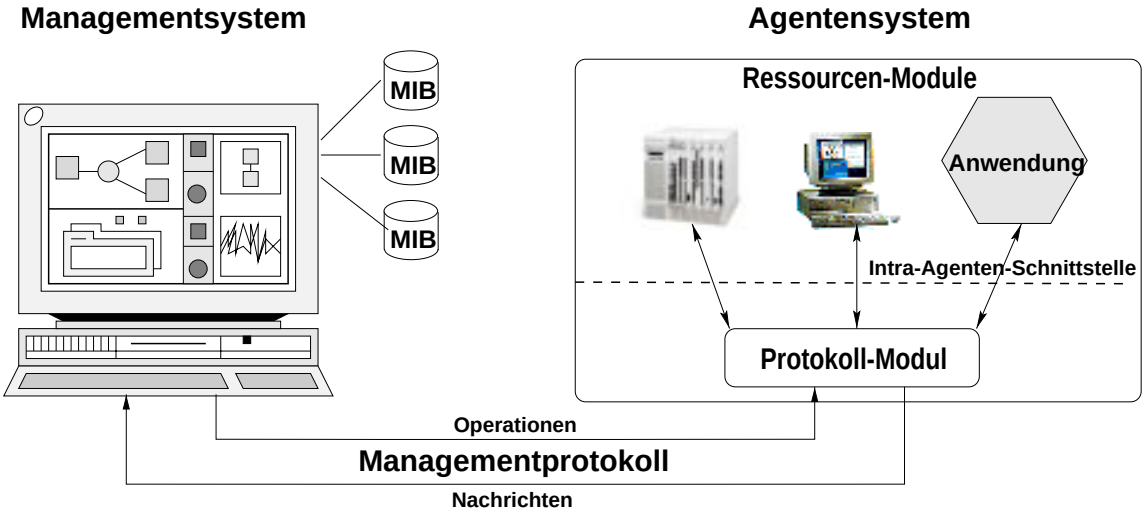


Abbildung 2.1: Management-Gesamtarchitektur

Die *systemübergreifende* Kombination von Modulen verschiedener Hersteller (siehe Abb. 2.1) wird durch standardisierte Managementarchitekturen [Slom 94, HeAN 99, Rose 94] ermöglicht. Sie definieren:

- im **Organisationsmodell** die Rollen der am Managementprozeß beteiligten Systeme und deren jeweilige Zuständigkeitsbereiche (**Domänen**): In der Regel werden Systeme, die aktive Rollen übernehmen, also steuernd auf andere Systeme einwirkend, als

Manager bzw. **Managementsysteme** bezeichnet. Andererseits werden passive Komponenten identifiziert, die ihrerseits von anderen Systemen administriert werden; man bezeichnet diese Systeme als **Agenten** bzw. **Agentensysteme**. Es sei bereits an dieser Stelle erwähnt, daß ein System auch beide Rollen spielen kann, d.h. sowohl als Manager, als auch als Agent agiert. Solche Systeme werden im weiteren als **verteilte kooperative Managementsysteme** bezeichnet.

- die zur Kommunikation zwischen den obigen Systemen erforderlichen Kommunikationskanäle, die durch **Managementprotokolle** realisiert werden. Sie setzen funktionsfähige End-to-End Verbindungen zwischen Managern und Agenten voraus und erlauben die Ausführung von Operationen durch Managementsysteme auf Agentensystemen bzw. die Zustellung von Nachrichten durch Agenten an Manager. Die Spezifikation dieser Aspekte geschieht im **Kommunikationsmodell**.

- ein *einheitliches* Format, in dem Managementinformation, also Information, die zu Managementzwecken auszutauschen ist, beschrieben wird. In einer **Management Information Base (MIB)**, deren Struktur durch das **Informationsmodell** determiniert wird, ist die Gesamtheit der Managementschnittstellen abgelegt, die ein Agent einem Manager zur Verfügung stellt. Dies beinhaltet nicht nur alle Operationen, Nachrichten und Attribute, sondern u.U. auch Angaben über die Struktur des vom Agenten verwalteten Systems, wie z.B. Enthaltenseinsbeziehungen zwischen einzelnen Objektklassen. Aufgrund der Tatsache, daß die Systemarchitekturen der eingesetzen Manager- und Agentensysteme völlig verschieden sein können, sind die Datenstrukturen einer herkömmlichen Programmiersprache kein geeignetes Format zur Beschreibung von Managementinformation. Es werden daher oft eigenständige, managementspezifische Notationen zur Beschreibung von Managementinformation verwendet.

- die für das Management von Systemen erforderliche **Managementfunktionalität**, die sich aus **Managementdiensten** zusammensetzt. Typische Beispiele, die im **Funktionsmodell** einer Managementarchitektur festgelegt werden, sind Dienste für die Verwaltung der Netztopologie und zur Überwachung von Schwellwerten sowie zur Zustellung, Vorverarbeitung und Filterung von Ereignismeldungen. In objektorientierten Managementarchitekturen werden diese Dienste in einer zur Managementinformation äquivalenten Weise spezifiziert, d.h. in Form von Objektklassen. Da die Beschreibung dieser Dienste plattformübergreifend geschehen muß, finden auch hier managementspezifische Notationen Anwendung. Generell gibt es zwei Varianten, wie Managementdienste ausgeführt sind: Erstere hat zum Ziel, die Semantik von Managementobjekten zu verfeinern, indem diese mittels Vererbung generische Managementinstrumentierung erhalten. Beispiele hierfür sind Dienste zur Berechnung statistischer Größen und zur Bildung von Durchschnittswerten. Die zweite Alternative besteht aus abgesetzt implementierbaren Diensten, die ihrerseits Objektklassen für eigenständige Komponenten enthalten, welche mit den im Informationsmodell spe-

zifizierten Managementobjektklassen interagieren. Typische Vertreter dieser Gattung sind Objekte zur Filterung und Zustellung asynchroner Ereignismeldungen.

Für den Austausch von Managementdaten existieren zwei verschiedene Konzepte: Das sogenannte *Pull*-Modell beruht auf dem Abfragen von Agenten durch das Managementsystem, d.h. die Agenten liefern Managementinformation nur auf explizite Anfragen des Managers. Geschehen diese Anfragen in wiederkehrenden zeitlichen Intervallen, spricht man von *Polling*. Demgegenüber liegt ein *Push*-Modell vor, wenn Agenten ohne vorherige Aufforderung durch das Managementsystem selbsttätig Managementinformation an den Manager senden. Dies ist der Fall bei asynchronen Ereignismeldungen, in denen ein Agent einen (oder mehrere) Manager über wichtige Zustandsänderungen informiert. Zwischen diesen beiden Modellen zum Austausch von Managementdaten sind ebenfalls Mischformen möglich, wie beispielsweise das im Internet-Management (siehe 3.1.3) verwendete „trap-directed polling“. Das Ziel ist dabei, die Reaktionszeit auf Seiten des Managers im Falle von Zustandsänderungen bei den Agenten zu minimieren, ohne zusätzlichen Netzverkehr durch das Setzen verkleinerter Pollingintervalle zu erzeugen: Ein Agent sendet eine asynchrone Ereignismeldung an den Manager, deren Aufgabe weniger darin besteht, Managementdaten zu übertragen, als vielmehr den Manager über das Auftreten eines außergewöhlichen Ereignisses zu benachrichtigen, der seinerseits umgehend (d.h. früher, als eigentlich geplant) einen neuen Pollingzyklus, jedoch mit gleichbleibenden Intervallen, einleitet (vgl. hierzu die in [PeMc 97] gemachten Ausführungen).

Sollen Managementlösungen flexibel an verschiedene Einsatzumgebungen angepaßt werden können, müssen sowohl innerhalb der Managementsysteme als auch der Agenten selbst die Module möglichst frei kombinierbar sein. Dies ist notwendig, da bereits auf einem einzigen System häufig eine Vielzahl von Anwendungskomponenten unterschiedlicher Hersteller in ein integriertes Management einzubinden sind. Die Module, die den Anschluß einer Komponente an das Management liefern (sog. **Ressourcen-Module**), werden i.a. mit der zu administrierenden Komponente mitgeliefert und stammen in der Regel jeweils von verschiedenen Herstellern. Damit sind auch innerhalb der Agenten Schnittstellen (sog. **Intra-Agenten-Schnittstellen**) offenzulegen, um die notwendige herstellerübergreifende Koexistenz und Kooperation der Module eines Agenten zu erlauben. Das reibungslose Zusammenwirken unterschiedlicher Agentenmodule innerhalb eines Agentensystems ist eine wesentliche Grundlage für flexibel konfigurierbare Managementlösungen. In der Literatur [WhGu 98] werden Agenten, die diesen Designprinzipien folgen, als **erweiterbare Agenten** (extensible agents) bezeichnet; wir werden in Abschnitt 3.1.3 darauf genauer eingehen.

Auf der Seite der Managementsysteme erreicht man die geforderte Modularität durch Offenlegung der Architektur und der Programmierschnittstellen sogenannter **Managementplattformen**, die wesentliche Integrationsaufgaben abdecken: Sie stellen eine gemeinsame Infrastruktur und Ablaufumgebung für Managementapplikationen verschiedener Hersteller bereit und unterstützen dies durch einheitliche Darstellung, Speicherung und Verwaltung sämtlicher Managementobjekte eines Rechnernetzes, auf deren Information mit

plattformeigenen graphischen Werkzeugen wie z.B. MIB-Browsern oder kommandozeilenorientierten Hilfsmitteln zugegriffen werden kann. Typischerweise verfügen Managementplattformen über mehrere Kommunikationsmodule, die den Austausch von Managementinformation über standardisierte Managementprotokolle gestatten; die Erweiterung der Plattformfunktionalität um ressourcenspezifische Dienste (wie z.B. das Management von Routern) durch sogenannte **Managementapplikationen** ist zwar prinzipiell möglich, da die entsprechenden Programmierschnittstellen (z.B. zur Ereignisverwaltung oder zur Topologiedatenbank) offengelegt (jedoch nicht standardisiert) und Entwicklungswerkzeuge ebenfalls im Lieferumfang enthalten sind bzw. im Rahmen einer Entwicklerlizenz erworben werden können. Die Menge an nutzbaren Schnittstellen und der Komfort der Entwicklungswerkzeuge variieren jedoch erheblich und bieten den Herstellern Möglichkeiten, um sich gegenüber Mitbewerbern abzugrenzen. Folglich basieren Managementapplikationen immer auf konkreten Plattformimplementierungen, was nicht nur den Verlust von Portabilität mit sich bringt, sondern auch das Prinzip offenen Managements konterkariert. So ist beispielsweise die Router-Managementapplikation *CiscoWorks* auf den Plattformen *HP OpenView* und *IBM Tivoli TME10 NetView* ablauffähig, jedoch nicht auf *Cabletron SPECTRUM*, obwohl der Austausch von Managementdaten in allen Fällen über dasselbe standardisierte Managementprotokoll erfolgt.

Das Netzmanagement als erste Ausprägungsform integrierten Managements, das auf die Überwachung und Steuerung von Netzkomponenten mit dem Ziel der Bereitstellung von Konnektivität abstellt, hat den breiten Einsatz von Managementplattformen begünstigt, da selbst in großen Rechnernetzen die Anzahl der Netzkomponenten noch relativ überschaubar ist. Der ausgesprochen hohe Kaufpreis, die Komplexität der Bedienung und die beträchtlichen Anforderungen an die Hardware des Systems, auf dem die Plattform läuft, haben zusätzlich dazu geführt, daß der Einsatz von Plattformen häufig eine zentralistische Organisation des Managements impliziert: Wenige, speziell geschulte Netzadministratoren administrieren und überwachen das gesamte Kommunikationssystem von einer zentralen Stelle aus. Eng damit verbunden ist auch der Designgrundsatz, daß sämtliche „Managementintelligenz“, d.h. das Problemlösungswissen und die dazugehörige Verarbeitungsfunktionalität an zentraler Stelle vorgehalten werden und Ressourcen lediglich managementrelevante Daten bereitstellen sollten. Dies beruht auf der Tatsache, daß auf Netzkomponenten wie Modems, Multiplexern, Hubs oder Bridges häufig nicht genügend Kapazität zur Vorverarbeitung von Managementinformation vorhanden ist. Ein charakteristisches Beispiel für eine Managementarchitektur, der eine zentralistische Organisation des Managements zugrunde liegt, ist das in Abschnitt 3.1.3 besprochene Internet-Management.

2.1.2 Hierarchisches Management

Der zunehmende Grad an Vernetzung und die steigende Zahl administrierter Komponenten haben dazu geführt, daß es in der Regel nicht mehr ausreicht, von einer zentralen Managementplattform aus *sämtliche* Komponenten eines Rechnernetzes zu administrie-

ren. Vielmehr ist es wünschenswert, Teile der verteilten Anwendung „Management“ aus Gründen des Lastausgleichs oder aus organisatorischen Erwägungen heraus möglichst flexibel auf verschiedene Systeme verlagern zu können. Dies geschieht zum Teil durch die Verlagerung bestimmter Dienste auf Agentensysteme, was jedoch aufgrund mangelnder Ressourcen häufig nur bedingt machbar ist (vgl. [Wald 93]).

Demgegenüber bietet sich die Einführung von **Managementhierarchien** mit einem **Toplevel Manager** sowie mehreren, jeweils für einen abgegrenzten Bereich zuständigen Managementsystemen (**Mid-level Managern**) an. Hierarchisches Management ist eine Organisationsform des Managements, in der Managementsysteme in logisch geschichteter Struktur organisiert sind. Systeme in höheren logischen Schichten haben eine Sicht auf das gesamte Kommunikationssystem und werden von unnötigen Details durch Systeme abgeschirmt, welche die unteren logischen Schichten eines Rechnernetzes administrieren. Erfahrungen aus dem praktischen Betrieb von Managementplattformen haben gezeigt, daß die Qualität des Managements so spürbar verbessert wird: Man erreicht eine bessere Skalierbarkeit des Managements insgesamt durch Reduzierung der Last auf dem bisher zentralen Managementsystem sowie der Netzlast auf den Backbone-Netzen. Der Preis dafür ist eine Erhöhung der Last auf den lokalen Netzen zwischen Agenten und Mid-level Managern. Dies ist jedoch unproblematisch, da die Bereitstellung akzeptabler Bandbreite in LANs lediglich geringe Kosten verursacht: Bandbreiten von 10 Mbit/s (Ethernet) bzw. 16 Mbit/s (Token Ring) sind heutzutage Standard und werden durch übliche Applikationen in der Regel noch nicht ausgereizt. Eine einfache Rechnung illustriert dies: Bei einem mittelgroßen LAN mit 50 handelsüblichen Ethernet Switches, die jeweils über 24 Anschlüsse verfügen soll die maximale Zeitspanne zur Entdeckung eines ausgefallenen Ports 5 Minuten nicht überschreiten. Dies erfordert bei einem Polling-basierten Ansatz durchschnittlich $(50 * 24)/(5 * 60) = 4$ Zugriffe pro Sekunde, um den Zustand aller Ports zu überwachen, was sich jedoch in der Praxis als durchschnittlich 1/6 Zugriffe pro Sekunde auswirkt, da Zugriffe auf dasselbe Gerät üblicherweise mit einer Anfrage abgewickelt werden. Will man nun 1200 Endgeräte[1], die an die Ports angeschlossen sein können, in 10-minütigen Abständen überwachen, sind hierfür weitere $1200/(5*60) = 4$ Zugriffe pro Sekunde vonnöten. Insgesamt werden knapp 5 Polling-Zugriffe pro Sekunde ausgelöst, was unter der Annahme, daß SNMP (mit einer durchschnittlichen Paketgröße von 500 Byte) verwendet wird, zu einer benötigten Bandbreite von 0,05 Mbit/s führt, was einem halben Prozent der insgesamt verfügbaren Bandbreite entspricht. Dieser verhältnismäßig geringe Bandbreitenverbrauch ist also durchaus tolerierbar. Ferner erlaubt der Einsatz von filternden Bridges und Ethernet-Switches in Verbindung mit strukturierter Verkabelung kontinuierliche Verbesserungen zur Erhöhung der effektiv nutzbaren Bandbreite. Sollte dies nicht ausreichen, bietet sich der Einsatz von Fast Ethernet (100 Mbit/s) für den LAN-Bereich ebenfalls zu vergleichsweise geringen Kosten an: Gegenwärtig erhältliche Endsysteme verfügen über Netzadapter mit *Autonegotiation*-Mechanismen, die sowohl mit 10 Mbit/s als auch mit 100 Mbit/s betrieben werden können.

[1] Die weiteren 50 Ports dienen zur Verbindung der Switches mit anderen Netzelementen.

Die Hauptaufgabe von Mid-level Managern ist also die Vorverarbeitung der Managementdaten von Agentensystemen, um nur in außergewöhnlichen Situationen das zentrale Managementsystem zu benachrichtigen. Sie nehmen einerseits die Filterung von Ereignismeldungen vor und gewinnen Management*information* aus Rohdaten, die von Agenten bereitgestellt werden. Somit können auch Netzsegmente, die nur über Verbindungen mit geringer Bandbreite an den Backbone angeschlossen sind, ebenfalls umfassend administriert werden, da ein dafür zuständiger lokaler Mid-level Manager ausschließlich bereits vorverarbeitete und daher kondensierte Managementinformation an den Top-level Manager weiterleitet. Ferner bringt der Einsatz von Mid-level Managern die Erhöhung der Fehlertoleranz bei Netzausfällen, da beim Ausfall einer Verbindung vom Backbone zu einem entfernten Teilnetz die dort befindlichen Systeme durch den Mid-level Manager auch weiterhin vollständig überwacht werden können.

Der Einsatz von Mid-level Managern impliziert die Notwendigkeit, diese Systeme geeignet zu konfigurieren und deren korrekte Funktionsweise zu überwachen. Hierfür gibt es jedoch bisher keinerlei offengelegte Managementinformation, wie in Abschnitt 3.3.1 aufgezeigt wird. Ebenfalls sind bisher keine Festlegungen getroffen, welche Arten von Managementinformation zwischen Mid-level Managern und Top-level Managern auszutauschen sind. Wir werden hierfür in Kapitel 5 einige Vorschläge herausarbeiten.

Hierarchisches Management ist dann angebracht, wenn bereits die organisatorische Struktur des Netzes hierarchisch aufgebaut ist, was insbesondere bei Telekommunikationsnetzen der Fall ist: Eine Managementarchitektur, die hierarchisches Managements realisiert, ist folglich das in Abschnitt 3.1.1 beschriebene *Telecommunications Management Network (TMN)*, das Kommunikationssysteme in fünf logische Schichten aufteilt und mit der Verwendung der OSI *Systems Management Functions* sowie des Managementprotokolls *Common Management Information Protocol (CMIP)* einige leistungsfähige Mechanismen zur statischen Delegierung von Managementfunktionalität (Schwellwertüberwachung, Filterung von Ereignismeldungen) sowie zur Ermittlung von Managementobjekten, auf die Operationen angewandt werden sollen (*Scoping and Filtering*) bietet. Da es in der Datenkommunikationswelt bisher keine standardisierten Konzepte für Mid-level Manager gibt, werden wir in Abschnitt 3.3.1 einen kommerziellen Mid-level Manager vorstellen, der dies mit den Mitteln des Internet-Managements leistet.

2.1.3 Verteiltes kooperatives Management

Aus dem erfolgreichen Einsatz von Managementplattformen zur Überwachung und Steuerung von Netzkomponenten folgt die Überlegung, dieselben Verfahren ebenfalls zum Management von Endsystemen (PCs, Workstations, Server) und den darauf laufenden Anwendungen einzusetzen. Hierbei kommen zwei Aspekte ins Spiel, die maßgeblichen Einfluß auf die Organisation des Managements haben:

1. Während die Zahl an Netzkomponenten in großen Netzen wenige tausend Stück kaum überschreitet, liegt die Zahl der angeschlossenen Endsysteme häufig im fünf-

und, unter Umständen, im sechsstelligen Bereich. Die jeweils darauf laufende Anwendungsanzahl ergibt, multipliziert mit der Zahl der Endsysteme, eine in der Regel siebenstellige Menge an Managementobjekten. Folglich scheidet ein zentralistischer, plattformbasierter Ansatz aus Skalierbarkeitsgründen von vornherein aus.

2. Die für das Management nutzbare Menge an Betriebsmitteln ist bei Endsystemen naturgemäß höher als bei einfachen Netzkomponenten. Demzufolge kann das Management eine wesentlich höhere Menge an Verarbeitungsleistung beanspruchen, was die Entwicklung komplexer Agentensysteme zuläßt.

Die aufgrund Punkt 1 zwingend notwendige Vorverarbeitung bzw. Verdichtung von Managementinformation durch Agenten und die damit verbundene Erhöhung der Verarbeitungslast kann aufgrund der vorhandenen Systemleistung toleriert werden, was einerseits die Managementsysteme entlastet und andererseits eine Verringerung des Kommunikationsaufwandes zwischen Manager und Agenten bewirkt. Außerdem bieten Endsysteme, im Gegensatz zu Netzkomponenten, aufgrund des Vorhandenseins von Betriebssystemen eine vollständige Laufzeitumgebung, die Managementprozessen die Zuweisung von Ablaufprioritäten und adäquates Scheduling ermöglicht. Desweiteren sind auf diesen Systemen häufig virtuelle Maschinen und sogar Skriptsprachen vorhanden, was die Portabilität der Agentenbestandteile begünstigt, da diese nun als gewöhnliche Softwaremodule vorliegen. Ferner bietet das Vorhandensein hoher Verarbeitungsleistung die Möglichkeit, softwaretechnische Konzepte wie Objektorientierung und Modularisierung anzuwenden, die bisher aus Effizienzgründen noch nicht auf Agentenseite realisiert werden konnten. Beispielweise kann der aus dem Software-Engineering bekannte „Componentware“-Ansatz (siehe [Sims 94], [Hump 95]) unmittelbar für die Implementierung modularer Managementagenten genutzt werden. Die Offenlegung der Schnittstellen von Ressourcenmodulen – also der o.a. Intra-Agenten-Schnittstellen – bildet wiederum die Basis für Multi-Vendor-Umgebungen: Drittherstellern eröffnen sich somit Perspektiven, von ihnen entwickelte Managementkomponenten nahtlos in bestehende Agentensysteme einfügen zu können. Beispiele hierfür sind die Erhöhung des Funktionsumfangs von Agentensystemen, einerseits um Module zur Administration neuer Ressourcenklassen (wie z.B. die Erweiterung eines Agenten zur Überwachung von Betriebssystemparametern um Managementdienste für Datenbanksysteme oder SAP R/3) und andererseits zur Bereitstellung erweiterter, generischer Managementfunktionalität (z.B. allgemeine Dienste zur Auswertung von Meßergebnissen und zur Bildung von Zeitreihenanalysen). Wünschenswert ist in diesem Zusammenhang, die Erweiterung der Agentensysteme um neue Managementfunktionalität zur Laufzeit vornehmen zu können, also neue Managementdienste an einen Agenten zu delegieren. Ein Beispiel hierfür ist die Delegierung von umfassenden Diagnosediensten an ein Agentensystem, nachdem dieses einen Fehlerzustand an den Manager gemeldet hat. Wir werden in Abschnitt 3.2.1 einen Ansatz diskutieren, der dies leistet und diesen in Kapitel 6 für unsere Belange erweitern.

Die Möglichkeit, Managementfunktionalität in verteilten Umgebungen delegieren zu können, bedingt eine Kooperation zwischen den am Managementprozeß beteiligten Sy-

stemen. Hierbei sind mehrere Kooperationsmodelle denkbar: Neben der oben angesprochenen Kooperation von Managern und Agenten ist ebenfalls die Kooperation zwischen Agentensystemen möglich; die Behandlung dieser Art von Kooperationsbeziehungen liegt jedoch außerhalb des Untersuchungsgegenstandes der vorliegenden Arbeit.

Ab einer gewissen Größe eines Rechnernetzes ist auch das koordinierte Zusammenspiel zwischen einzelnen Managementsystemen notwendig: Jeweils für einen Teil des Rechnernetzes zuständige Managementsysteme tauschen Informationen mit Partnersystemen aus, um sicherzustellen, daß die eigenen (d.h. lokalen) Betriebsziele nicht im Gegensatz zu globalen Zielvorgaben stehen. Hierfür kann es nicht nur aus Gründen der Ausfallsicherheit notwendig sein, Managementfunktionalität eines Managementsystems temporär an ein Partnersystem zu delegieren. Wir werden im weiteren Verlauf dieser Arbeit Systeme, die diese Eigenschaften haben, als **verteilte kooperative Managementsysteme** bezeichnen und in Abschnitt 2.2 einige Managementszenarien schildern, die die Wichtigkeit der Kooperation von Managementsystemen illustrieren. Ebenfalls wird dort aufgezeigt, daß die im vorigen Abschnitt 2.1.2 gemachten Ausführungen bezüglich fehlender Managementinformation zur Administration von Managementsystemen natürlich auch in verteilten kooperativen Umgebungen gelten. Das wesentliche Unterscheidungsmerkmal zwischen verteilten kooperativen Managementsystemen und Mid-level Managern liegt darin, daß letztere in einer festen, vordefinierten hierarchischen Struktur zum Einsatz kommen. Erstere hingegen unterliegen rein technisch keinerlei Hierarchien, obwohl es jederzeit durch geeignete Konfiguration möglich ist, diese in einen hierarchischen Kontext einzuordnen.

Ein betriebliches Argument für verteiltes koopratives Management ergibt sich aus einer besseren Anpassung des Managements an die realen organisatorischen Gegebenheiten einer Unternehmung, in denen die früher starren Hierarchien oftmals einer flexibleren Matrixorganisation gewichen sind. Mit der Modularisierung und der daraus folgenden Flexibilität gelingt eine bessere Abbildung des Managements auf betriebliche Abläufe („Workflows“) und Geschäftsprozesse.

Die in Abschnitt 3.1.2 vorgestellte *Common Object Request Broker Architecture (CORBA)* ist eine für Managementzwecke nutzbare Softwarearchitektur, die die Realisierung verteilten kooperativen Managements begünstigt.

2.1.4 Umbrella Management

Mit der modularen Architektur von Managementsystemen und Agenten sind prinzipiell die notwendigen Voraussetzungen für die Entwicklung integrierter Managementlösungen vorhanden, wenn man lediglich *genau eine* Managementarchitektur zu berücksichtigen hat, d.h. alle Managementsysteme und Agenten dieselbe Beschreibungssprache und dasselbe Managementprotokoll verwenden. Dies ist jedoch gegenwärtig kaum der Fall, da neben diversen proprietären Ansätzen *mehrere* Managementarchitekturen standardisiert wurden, die hinsichtlich ihrer Teilmodelle verschiedene Lösungsansätze zeigen und sich deshalb in Abhängigkeit der jeweils zugrundegelegten Netzszenarien unterschied-

lich am Markt durchgesetzt haben. Neben der im Bereich des LAN- und Endsystemmanagements weit verbreiteten Internet-Managementarchitektur (*SNMP-Management*, vgl. Abschnitt 3.1.3) und der in öffentlichen Datennetzen bzw. Telekommunikationsnetzen angewandten *OSI/TMN-Managementarchitektur* (vgl. Abschnitt 3.1.1) der ISO/ITU werden auch die Arbeiten der OMG (*Common Object Request Broker Architecture (CORBA)*, vgl. Abschnitt 3.1.2) zunehmend beachtet. Letzterer werden heute gute Chancen auf große Verbreitung für die Entwicklung verteilter Anwendungen und deren Management eingeräumt.

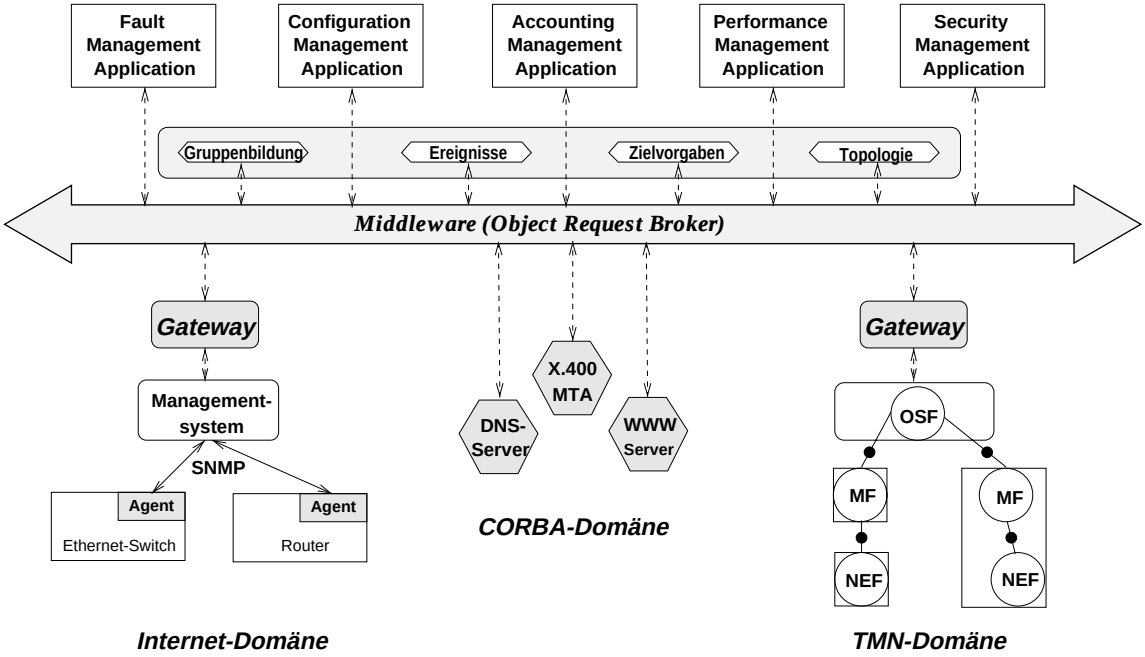


Abbildung 2.2: Umbrella Management: Verschattung heterogener Architekturen

Insgesamt kann also davon ausgegangen werden, daß in großen Rechnernetzen zahlreiche Komponenten interagieren, die aus der Sicht des Managements unterschiedlichen Architekturen angehören. Diese Heterogenität führt zu einer Bildung abgeschlossener Architekturdomänen, in denen die jeweils in einer solchen Domäne enthaltenen Systeme über einen einheitlichen Managementkontext (hinsichtlich aller vier Teilmodelle von Managementarchitekturen) verfügen und miteinander interagieren können. Aufgrund der Heterogenität der zugrundegelegten Architekturen ist es jedoch nicht möglich, daß Systeme einer Architekturdomäne mit davon außerhalb liegenden Systemen kommunizieren. Dies steht jedoch im Widerspruch zu einer zentralen Forderung integrierten Managements wonach der Zugriff auf die *gesamte* Menge an vorhandener Managementinformation notwendig ist, und zwar unabhängig davon, wie diese beschaffen ist, d.h. in welchen Formaten diese vorliegt und mit Hilfe welcher Mechanismen der Zugriff erfolgt. Es ist daher von erheblicher Wichtigkeit, Übergänge zwischen heterogenen Managementarchitekturen zu schaffen.

In Abbildung 2.2 ist das Ziel des Umbrella Managements dargestellt: Ein (potentiell verteiltes) Managementsystem auf der Basis einer Managementarchitektur (hier: CORBA) wird durch das Umbrella Management in die Lage versetzt, nicht nur die Systeme mit gleichem Managementkontext zu administrieren, sondern auch Systeme fremder Architekturdomänen (hier: Internet-Management, TMN). Hieraus ergibt sich zwangsläufig die Frage, mit welchen Mitteln diese Integration erfolgen kann. Aufgrund der hohen Bedeutung dieses Problems für integriertes Management werden wir uns in Kapitel 4 dediziert mit denjenigen drei Varianten auseinandersetzen, die Umbrella Management realisieren. Eine Variante besteht im Einsatz der in Abschnitt 4.4 näher vorgestellten **Management-Gateways**, die sich jeweils an den Grenzen architektureller Domänen befinden und die einzelnen Teilmodelle der unterschiedlichen Managementarchitekturen aufeinander abbilden, um alle Systeme einer fremden Architektur in einen gemeinsamen Managementkontext einzubinden.

Die detaillierte Kenntnis aller in der zu integrierenden Architekturdomäne liegenden Systeme und ihre zentrale Position als Bindeglied zwischen unterschiedlichen Architekturen machen Management-Gateways zu prädestinierten Kandidaten für verteilte kooperative Managementsysteme. Dies resultiert ferner auch aus der Tatsache, daß solche Systeme zwangsläufig eine Managerrolle gegenüber den administrierten Ressourcen der zu integrierenden Architekturdomäne übernehmen.

2.1.5 Enterprise Management

Der Begriff „Enterprise Management“ wird in der Literatur unterschiedlich definiert: Eine frühe, sehr allgemeine Definition beschreibt Enterprise Management als „whatever it takes for a manager to ensure that the environment serves the needs of its users“ [StSy 94]. Nach [HeAN 99] faßt Enterprise Management „die Aufgaben des Finanz-, Personal-, Technologie- und Produktionsmanagement unter unternehmensweiten Gesichtspunkten (Geschäftsfelder, Geschäftsprozesse) zusammen und leitet daraus Zielvorgaben (Policies) für die IT-Infrastruktur, die Betriebsprozesse, die zugehörigen Dienste und Datenbestände ab“. Folglich kennt auch das Enterprise Management einen Domänenbegriff, der hier jedoch aufgrund seiner Abstraktion von technischen Spezifika rein organisatorischer Art ist und die Definition betrieblicher und geographischer Verantwortungsbereiche sowie entsprechende Zuständigkeitsprofile erlaubt.

Wir werden für die vorliegende Arbeit diese Definition ein wenig einschränken, da wir primär an den **technischen Fundamenten des Enterprise Managements** interessiert sind, auf denen dann von den betrieblichen Geschäftsprozessen abgeleitete unternehmensweite Zielvorgaben definiert und überwacht werden können. Folglich umfaßt Enterprise Management in unserem Kontext *alle Mechanismen, die für das umfassende Management unternehmenskritischer Dienste und Anwendungen sowie der gesamten IT-Infrastruktur in einem heterogenen Umfeld erforderlich sind.* Diese Definition stellt aufgrund des darin beschriebenen weiten Aufgabenumfangs naturgemäß sehr hohe Anforderungen an die

zugrundeliegende Managementarchitektur. Wir werden daher in Kapitel 3 gegenwärtige Managementarchitekturen und -systeme kritisch auf ihre Eignung für Enterprise Management überprüfen und bewerten sowie im weiteren Vorschläge für wichtige Erweiterungen erarbeiten.

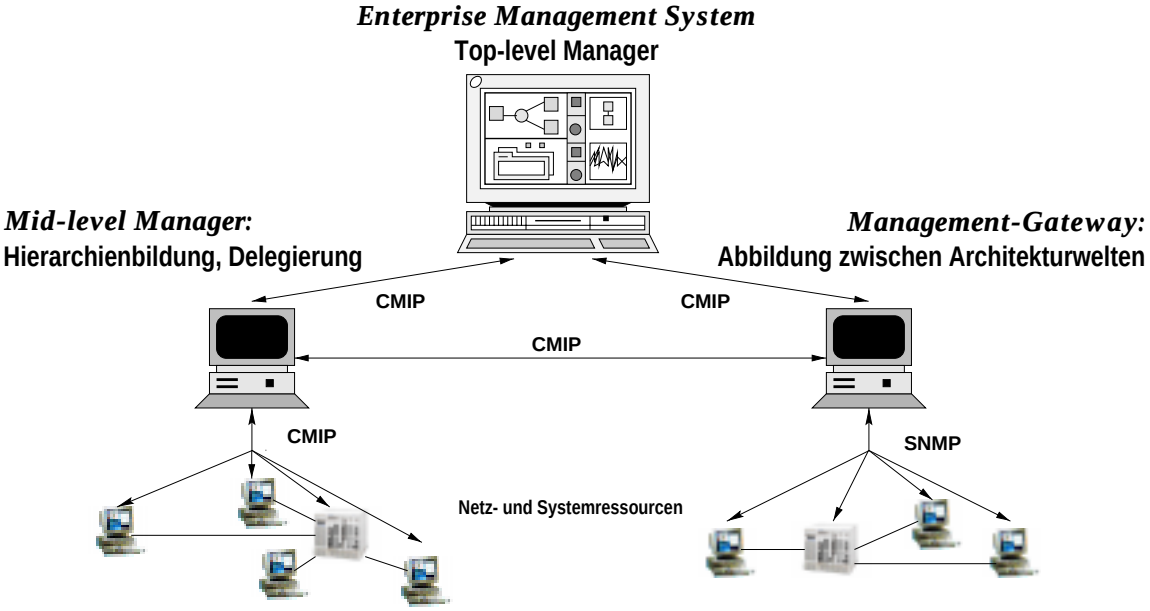


Abbildung 2.3: Ausprägungsformen verteilter kooperativer Managementsysteme

Grundsätzlich sind Enterprise Management Systeme sehr leistungsfähige Managementplattformen, die sowohl hinsichtlich ihrer Verarbeitungskapazität als auch ihres Dienstumfangs den zentralen Interaktionspunkt für Netz- und Systemadministratoren mit dem gesamten Kommunikationssystem darstellen. Sie sind demzufolge in der obersten Ebene einer Managementhierarchie angesiedelt und kommunizieren nur in Ausnahmefällen unmittelbar mit den Netzressourcen, da dies bei großen Kommunikationsnetzen, wie in Abschnitt 2.1.2 ausgefürt wurde, zu Skalierbarkeitsproblemen führt. Vielmehr sind Enterprise Management Systeme zu einem hohen Grad von einer Vorverarbeitung der anfallenden Managementdaten durch vorgelagerte Mid-level Manager abhängig. Die Integration von Systemen aus heterogenen Architekturdomänen geschieht beispielsweise durch die oben angesprochenen Management-Gateways, die jedoch ebenfalls eine Verdichtung von Rohdaten in Managementinformation vornehmen können. Wie in Abbildung 2.3 dargestellt, sind Mid-level Manager und Management-Gateways die beiden möglichen Ausprägungsformen verteilter kooperativer Managementsysteme, die von fundamentaler Bedeutung für die Effektivität und die Effizienz des gesamten Enterprise Management Prozesses sind. Um ein koordiniertes Zusammenwirken dieser Systeme zu erreichen, ist es daher sehr wichtig, daß diese ihrerseits eine offengelegte Managementschnittstelle besitzen, über die sie geeignet überwacht und gesteuert werden können. Abbildung 2.3 veranschaulicht ebenfalls die hierbei möglichen Kommunikationsbeziehungen: Neben dem Management durch

das Enterprise Management System können diese verteilten kooperativen Managementsysteme ebenfalls miteinander interagieren. Für ein integriertes Management dieser Systeme ist es daher unabdingbar, beide Anwendungsfälle verteilter kooperativer Managementsysteme einer genaueren Betrachtung hinsichtlich ihrer spezifischen Anforderungen zu unterziehen. Dies geschieht in den Kapiteln 4 und 5.

2.2 Szenarien des Enterprise Managements

Dieser Abschnitt beschreibt einige Szenarien aus der Praxis großer Netzbetreiber, die aktuelle Probleme bei der Umsetzung integrierten Enterprise Managements verdeutlichen. Anhand dieser Managementszenarien wird aufgezeigt, daß der überwiegende Teil der gegenwärtig auftretenden Probleme auf das oft mangelhafte Zusammenspiel der verteilten Managementsysteme zurückzuführen ist. Die Notwendigkeit, offengelegte Schnittstellen zum Management von Managementsystemen bereitzustellen, wird durch mehrere Szenarien aus dem Daten- und Telekommunikationsumfeld illustriert.

2.2.1 Managementinseln beim Betrieb mehrerer Managementsysteme

Die in Abschnitt 2.1.2 motivierte Notwendigkeit der Verteilung der Managementlast auf mehrere Managementsysteme führt dazu, daß in großen unternehmensweiten Netzen mehrere Managementsysteme eingesetzt werden , die von unterschiedlichen Herstellern stammen und jeweils für eine Domäne zuständig sind. Natürliche Managementdomänen sind, wie in Abbildung 2.4 dargestellt, die Zentrale eines Unternehmens sowie die einzelnen Zweigwerke, Niederlassungen und z.T. Händler. Diese geographische Domänenbildung ähnelt oft der Organisationsform der Unternehmung, da diese Domänen meist eigenverantwortliche Bereiche darstellen. Erfahrungen im Rahmen diverser Forschungskooperationen mit großen Netzbetreibern (u.a. mit der *Deutschen Telekom AG* und den *Bayerischen Motorenwerken (BMW) AG*) haben gezeigt, daß die heutzutage dort zahlreich vorhandenen Managementsysteme meist nur isoliert betrieben werden können, da es sowohl technisch als auch organisatorisch kaum machbar ist, diese Systeme kooperativ zu betreiben. Ursachen hierfür bestehen einerseits darin, daß die Umsetzung hierarchischen bzw. verteilten kooperativen Managements mit gegenwärtigen Managementplattformen nur sehr schwierig machbar ist, da diese häufig von Seiten der Hersteller als Top-level Manager ausgelegt sind. Dies ist aus Herstellersicht verständlich, da man so sowohl die Leistungsfähigkeit der eigenen Produkte unterstreicht, als auch diese gegenüber Mitbewerbern abschottet, indem die Hemmschwelle für den Einsatz konkurrierender Systeme mangels geeigneter Kooperationsmöglichkeiten mit den bisher eingesetzten relativ hoch gehalten wird. Ebenfalls sind bislang noch keine Bestrebungen bekannt geworden, herstellerübergreifende Standards für

Managementschnittstellen für diese Systeme festzulegen. Dies führt dazu, daß Funktionalität zur Überwachung und Steuerung von Managementsystemen, soweit überhaupt vorhanden, zumeist auf mehrere Programmierschnittstellen verteilt ist und keinerlei Agenten existieren, die die gewünschten Dienste auf offene Weise anbieten. Werden, wie es in unternehmensweiten Kommunikationsnetzen häufig der Fall ist, Managementsysteme unterschiedlicher Hersteller eingesetzt, sind natürlich die Möglichkeiten zur Kooperation dieser Systeme aufgrund fehlender Standards äußerst eingeschränkt. Stammen die in einem Corporate Network eingesetzten Systeme von demselben Hersteller, ist die Situation etwas besser: Fortschrittliche Systeme verfügen dann beispielsweise über sogenannte *Manager Overtake* Funktionen, welche die Verantwortung für eine Managementdomäne beim Ausfall eines Managementsystems (der u.U. über Managementagenten mit offengelegten Schnittstellen bemerkt wird) auf ein anderes System übertragen. Die technischen Details dieser Mechanismen werden in Abschnitt 3.3.1 vorgestellt; es sei an dieser Stelle lediglich erwähnt, daß diese Verfahren gegenwärtig relativ einfach gehalten sind und keinesfalls die redundante Haltung von Managementdaten durch Datenintegration implizieren.

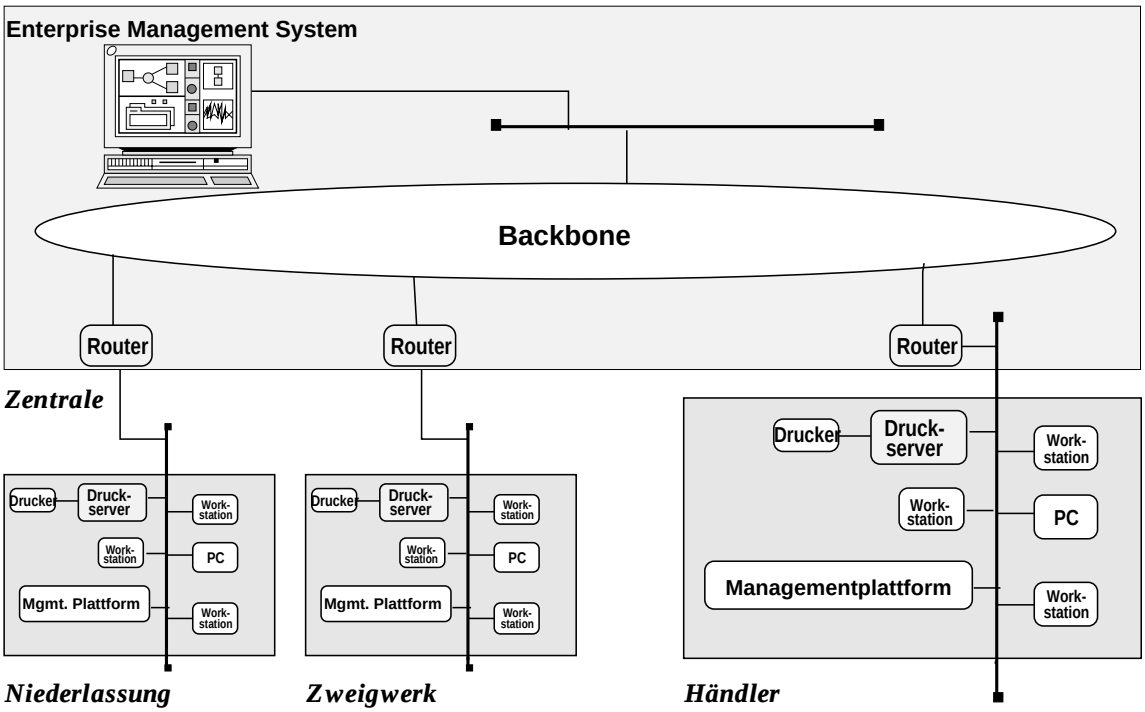


Abbildung 2.4: Unabhängiger Betrieb von Managementsystemen

Andererseits erweist es sich bei der praktischen Umsetzung von Betriebskonzepten für Managementsysteme oftmals als ausgesprochen schwierig, eigenständige bzw. angegliederte Bereiche des Unternehmens (wie Zweigwerke, Niederlassungen oder Händler) in eine ganzheitliche, unternehmensweite Managementstrategie einzubinden: Es ist beispielweise einer unabhängigen organisatorischen Einheit wie z.B. einem Zweigwerk nicht oh-

ne weiteres vermittelbar, daß zur Erreichung eines einheitlichen Enterprise Managements über lange Zeit erfolgreich eingesetzte Managementsysteme (im konkreten Fall: Cabletron Spectrum) durch neue Systeme (hier: IBM/Tivoli) ersetzt werden sollen, die in der Zentrale des Netzbetreibers beschafft wurden. Die Durchsetzbarkeit an zentraler Stelle getroffener Entscheidungen in Unternehmen, die in eigenständige Einheiten aufgeteilt sind, ist zumeist problematisch, da dies oftmals als Eingriff in die Autonomie eines unabhängigen Bereichs angesehen wird.

Beide Ursachen führen letztendlich zur Bildung autonomer Management-„Inseln“, was zwar heutzutage die Situation in zahlreichen Unternehmen darstellt, jedoch im Widerspruch zu den Prinzipien des Enterprise Managements steht. Da die Managementsysteme über Netzkonnektivität verfügen, ergibt sich die paradoxe Situation, daß die Systeme zwar nicht miteinander interagieren, jedoch auf die jeweils vom Partnersystem überwachten Ressourcen einwirken können. Bis jetzt haben für das Enterprise Management System zuständige Administratoren keinerlei Möglichkeiten, Konfigurationsprofile für Partner-Managementsysteme zu erstellen und deren Durchsetzung auf technischem Wege zu forcieren. Solche Profile können sich einerseits auf Polling-Intervalle und die Weiterleitung asynchroner Ereignismeldungen mit dem Ziel der Verringerung der Gesamtnetzlast beziehen; andererseits sollte unbefugten Systemen die Ausführung von Managementaktionen bzw. Discovery-Aktivitäten in bestimmten Bereichen untersagt werden können. Gesucht sind also Verfahren, die die Bildung von Verantwortungsbereichen und darauf abgestimmter Konfigurationsprofile für verteilte kooperative Managementsysteme zulassen.

2.2.2 Fehlkonfiguration von Managementsystemen

Ein anderes Szenario, das die Notwendigkeit des Managements von Managementsystemen illustriert, stammt aus dem universitären Umfeld: Neben dem Hochschulrechenzentrum verfügen z.T. auch Institute über eigene Managementplattformen, die bei fehlerhafter Konfiguration den Gesamtbetrieb der Hochschulnetze beeinträchtigen können.

Während der Großteil der Forschungseinrichtungen über einen unmittelbaren Netzzugang zum Universitätsbackbone verfügt, werden Institute mit geringem Datenaufkommen aus Kostengründen häufig über Wählleitungen an das Backbone-Netz angeschlossen. Hierbei wird lediglich dann eine Verbindung zum Backbone hergestellt, wenn Daten vom bzw. zum Institut übertragen werden sollen. Die Modems, die die Verbindung zum Backbone über Wählleitungen herstellen, sind in der Regel so konfiguriert, daß ca. 5 Minuten nach Beendigung einer Übertragung jeweils die Wählverbindung abgebaut wird. Sollten in dieser Zeit wieder Übertragungen erfolgen, bleibt die Verbindung bestehen und der Timeout-Mechanismus wird nach Beendigung der Übertragung neu gestartet.

Bei mancher Modemsoftware besteht keine Möglichkeit, bestimmte Paketarten (wie z.B. zum Testen der IP-Konnektivität mittels ICMP) von der Übertragung auszuschließen, was sich negativ bemerkbar macht, wenn innerhalb des Hochschulnetzes eine neue Managementplattform installiert wird und an dieser – mit einer Ausnahme – keine Änderungen an

der Default-Konfiguration vorgenommen werden: Damit die Plattform per Autodiscovery alle Systeme in den Subnetzen des Hochschulnetzes in die Topologiedarstellung mit aufnimmt, wird die Funktion abgeschaltet, die lediglich die Geräte bis zum nächsten Router erfaßt. Wenn an der Plattform ein Polling-Zyklus von 5 Minuten eingestellt ist, erhalten somit sämtliche entdeckten Systeme (darunter auch diejenigen, deren Wählverbindung zu diesem Zeitpunkt gerade aufgebaut ist) im Abstand von jeweils 5 Minuten ein ICMP-Paket zur Ermittlung ihres Systemstatus.

Die unglückliche Kombination von Timeoutwert der Wählverbindung und Polling-Zyklus der Managementplattform führt unter diesen Umständen dazu, daß die (nach Verbindungszeit abgerechnete) Wählverbindung tagelang aufrechterhalten wird, ohne daß in nennenswertem Umfang Nutzdaten übertragen werden, da jedes ICMP-Paket der Plattform den Timeout-Wert der Wählverbindung wieder zurücksetzt. Das Ergebnis sind unnötig hohe Kommunikationskosten, die nur dadurch vermieden werden können, daß der Verbindungsrechner, über den neben diesen Wählverbindungen auch der Internet-Zugang des Hochschulnetzes abgewickelt wird, durch den Plattformverwalter explizit vom Polling ausgenommen wird. Es besteht also keine Möglichkeit für die Backbone-Administratoren im Rechenzentrum, von ihrem Managementsystem aus anderen Managementplattformen das Überwachen derjenigen Systeme und Netzbereiche zu untersagen, für die sie nicht zuständig sind.

2.2.3 Zusammenwirken unterschiedlicher Telekom-Carrier

Die Umsetzung des *Telecommunications Reform Act of 1996 (TA96)* [TAct 96] der Vereinigten Staaten stellt die Telekommunikationsindustrie vor neue Herausforderungen, mit denen im Zuge der Deregulierung der europäischen und asiatischen Märkte [IsNi 98] für Telekommunikationsdienstleistungen der überwiegende Teil der TK-Dienstanbieter konfrontiert ist: Bisher war ein Kunde mit der Vergabe des Auftrags zur Einrichtung eines Anschlusses auf einen bestimmten Carrier festgelegt, der damit auch die Bereitstellung aller weiteren TK-Dienste übernahm. Dies hat sich mit der Wirksamkeit des Telekommunikationsgesetzes von 1996 geändert: Man unterscheidet in diesem Zusammenhang etablierte Telekommunikationsgesellschaften (*Incumbent Local Exchange Carrier, ILEC*) und neue Wettbewerber (*Competitive Local Exchange Carrier, CLEC*), die beide vor beträchtliche Herausforderungen gestellt werden (siehe auch [McPo 98]), da ihre jeweiligen Managementsysteme (*Operations Systems, OS*) nun miteinander interagieren müssen, um den Anforderungen von TA96 gerecht zu werden. Im folgenden sind zwei Beispiele solcher Anforderungen wiedergegeben [Wate 98]:

- „An incumbent LEC shall provide nondiscriminatory access [...] on an unbundled basis to any requesting telecommunications carrier for the provision of a telecommunications service [...]“.
- „An incumbent LEC must provide electronic access [...]. Providing access to OS functions [...] is a critical requirement for complying with section 251 of the act“.

Mögliche Szenarien umfassen sowohl die Bereitstellung von IN-Diensten durch den CLEC (speziell tarifierte Rufnummern, Konferenzschaltung) bei gleichzeitiger Nutzung der ILEC-Infrastruktur (Switches, Transportnetz) sowie die Verwendung der ILEC-Verkabelung zu den jeweiligen Teilnehmern bei Gesprächsvermittlung durch den CLEC oder die Beibehaltung der Rufnummer beim Wechsel des Carriers.

Hieraus ergeben sich folgende Interaktionen zwischen den jeweiligen Dienstleistern, für die zudem maximale Antwortzeiten festgelegt sind:

- Erfragen und Reservieren von ILEC-Diensten durch den CLEC (maximale Antwortzeit bei 98% der Anfragen: 2 Sekunden).
- Bereitstellung von Information bzgl. der Nutzung von Diensten.
- Übermittlung von Abrechnungsdaten.
- Entgegennahme und Verarbeitung von Störungsmeldungen; Durchführung von Tests.
- Austausch von Kundendaten, falls ein Endkunde den Dienstanbieter wechselt (maximale Dauer: 5 Stunden).

Diese Interaktionen bedingen die Kooperation der CLEC- und ILEC-Managementsysteme, woraus sich wiederum die Notwendigkeit einer Offenlegung der Managementschnittstellen ergibt: Schließlich müssen nunmehr nicht nur die Managementsysteme eines Anbieters interagieren, sondern sämtliche Systeme aller CLECs und ILECs. Um dies zu ermöglichen, müssen nicht nur sowohl die Syntax und Semantik der auszutauschenden Informationen festgelegt werden, sondern auch die Mittel zum Zugriff darauf (d.h. die Managementprotokolle). Es stellt sich daher die Frage, welche Managementarchitektur am geeignetsten für diese Zwecke ist und wie unterschiedliche Managementarchitekturen aufeinander abgebildet werden können. Wir werden hierauf in Kapitel 4 detailliert eingehen.

2.2.4 Dienstgüte bei Service Provider Hierarchien

Während früher durch die Bindung an einen einzigen Provider von Telekommunikationsdiensten die Überwachungs– und Eingriffsmöglichkeiten in die durch den Provider zur Verfügung gestellte Netzinfrastruktur in der Regel nicht vorhanden waren und damit auch die Managementsysteme nicht interoperabel zu sein brauchten, ist dies nach dem Wegfall der Telekommunikationsmonopole nicht mehr der Fall: Ein IT-Betreiber geht mit den Anwendern Dienstgütevereinbarungen ein, deren Erfüllung unmittelbar von der Dienstgüte der durch den Provider zur Verfügung gestellten Netzinfrastruktur abhängt. Aufgrund der Liberalisierung auf dem Telekommunikationssektor und der damit einhergehenden Konkurrenzsituation zwischen den Anbietern von Netzinfrastrukturen ist es dem IV-Betreiber nunmehr möglich, seinerseits Dienstgütevereinbarungen mit dem bzw. den Providern abzuschließen. Zu deren Überwachung ist es erforderlich, daß der Provider dem IV-Betreiber Einblick in Kenngrößen seiner Systeme gewährt. Dies impliziert:

- die Sicherstellung der Interoperabilität von Systemen, die unterschiedliche Managementarchitekturen verwenden, sowie
- die Festlegung, welche Informationen über ein Managementsystem eines Diensterbringers dem Dienstnutzer zur Verfügung gestellt werden sollen.

Schließlich sollte einerseits ein Dienstnutzer wissen können, welche Diagnosemaßnahmen des Managementsystems eines Diensterbringers ihm zur Verfügung stehen bzw. welche Beschränkungen ihm dabei auferlegt sind. Andererseits muß ein Diensterbringer auch in der Lage sein, das Managementsystem des Dienstnutzers so beeinflussen zu können, daß die von dort ausgehenden Maßnahmen nicht seinen eigenen Betriebszielen zuwiderlaufen. Eine grundlegende Voraussetzung hierfür ist, daß die Managementsysteme der unterschiedlichen Diensterbringer und Dienstnutzer miteinander kommunizieren können und über eine einheitliche Management-Begriffswelt verfügen: Wie Abbildung 2.5 verdeutlicht, ist es keinesfalls selbstverständlich, daß die Managementsysteme aller beteiligten Partner zu einer einheitlichen Managementarchitektur konform sind. Vielmehr zeigt der bevorzugte Einsatz von SNMP im LAN-Bereich sowie der hohe Anteil an OSI/TMN-basierten Managementwerkzeugen im Bereich der Telekommunikation, daß zumindest an der Schnittstelle zwischen Carrier und IP-Provider einiger Aufwand erforderlich sein wird, damit diese Systeme überhaupt Managementinformation austauschen können. Die (in Kapitel 3 ausführlich dargelegte) Vielfalt an Managementarchitekturen läßt darauf schließen, daß die Interoperabilitätsproblematik auch an weiteren Ebenen der Service Provider Hierarchie auftritt. Eine detaillierte Darstellung und Bewertung dieser Lösungen findet sich in Kapitel 4. Die Interoperabilität von Managementsystemen stellt die Grundlage dar, auf der die Managementsysteme der jeweiligen Partner nicht nur passiv kommunizieren, sondern ebenfalls *kooperativ* tätig werden können. Entsprechende Maßnahmen zur Sicherstellung der Interoperabilität sind im Rahmen dieser Arbeit intensiv erforscht worden.

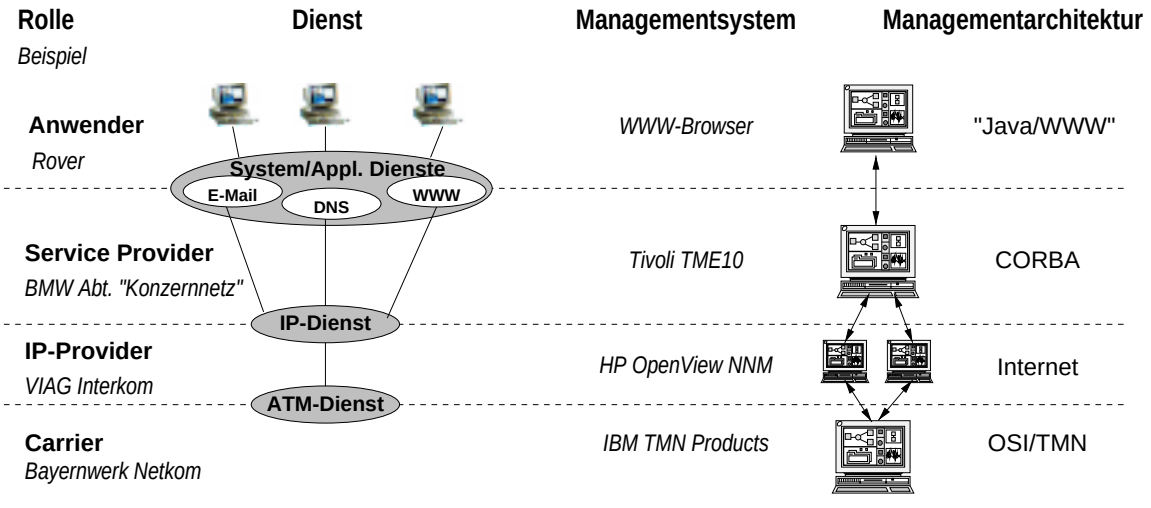


Abbildung 2.5: Geschichtete Dienstnutzer- / Diensterbringer-Hierarchien

Abbildung 2.5 zeigt ebenfalls die Abgrenzung dieser Arbeit gegenüber einer anderen, gegenwärtig offenen Fragestellung, die u.a. im Rahmen eines aktuellen DFN-Projekts[2] untersucht wird: Während die vorliegende Arbeit auf die Überwachung und Steuerung von Managementsystemen in heterogener Umgebung fokussiert (d.h. den rechten Teil von Abbildung 2.5), befaßt sich **Customer Network Management** mit dem Problem, welche Art von Managementinformation einem Dienstnutzer durch den Diensterbringer zur Verfügung gestellt werden sollte (siehe hierzu [ADKL 97] und [LaLN 98]). Letzteres umfaßt *sämtliche* Dienste und Systeme des Diensterbringers (also den linken Teil der Grafik) und ist hinsichtlich der Überwachung von Dienstgütevereinbarungen durch den Dienstnutzer von fundamentaler Bedeutung, da er bei unzureichender Dienstgüte entscheiden kann, ob ein Fehler in seinem Verantwortungsbereich vorliegt oder ob das Problem in einer Hierarchieebene tiefer angesiedelt ist. Für eine effektive Fehlerdiagnose in geschichteten Systemen ist dies unabdingbar. Dem steht eine oftmals restriktive Informationspolitik auf Seiten des Diensterbringers entgegen, da dieser die ihm zur Verfügung stehende Managementinformation oft als vertraulich einstuft. Die Informationsmenge, die sowohl die Informationsbedürfnisse des Dienstnutzers als auch die Vertraulichkeitsanforderungen des Diensterbringers erfüllt, ist oft nur schwer bestimmbar.

Um Managementsysteme in heterogener Umgebung überwachen und steuern zu können, ist es daher unabdingbar, einen Mindestumfang an Managementinformation festzulegen und eine Architektur auszuwählen, die der verteilten Natur des Managements gerecht wird. Wir werden daher in Kapitel 3 gegenwärtige Architekturen auf ihre Eignung für das Enterprise Management überprüfen und anhand der im folgenden Abschnitt 2.3 zusammengefaßten Anforderungen eine Auswahl treffen. Auf dieser Architektur werden wir dann unsere weiteren Untersuchungen aufbauen und Möglichkeiten aufzeigen, mit welchen Mitteln die Unterschiede zwischen Managementarchitekturen am geeignetsten überbrückt werden können.

2.2.5 Verteiltes Management von LEO/MEO-Satellitennetzen

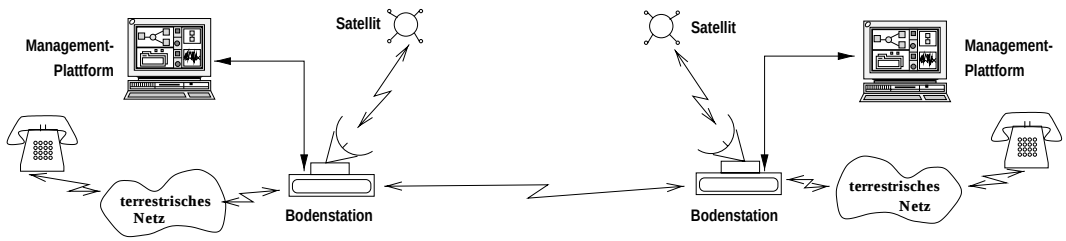
Wir werden nun einen Problembereich vorstellen, der im Vergleich zu den bisher vorgestellten Szenarien über eine ausgesprochen hohe Dynamik verfügt und daher spezielle Anforderungen an das Management stellt: Die Inbetriebnahme des ersten globalen Satellitennetzes für Mobilkommunikation im Herbst 1998 unterstreicht auch die hohe Aktualität und Praxisrelevanz dieses Themenkomplexes. Das Szenario verdeutlicht ebenfalls den Wechsel von zentralisiertem Management hin zu verteiltem kooperativem Management.

Zwar wurden bereits seit 1965 diverse Kommunikationsdienste (Abwicklung interkontinentaler Telefonie, Fernsehausstrahlungen) mit Hilfe geostationärer Satelliten (*geosynchronous earth orbit (GEO)*) abgewickelt, die sich in einer Höhe von 36000 Kilometern über dem Äquator befinden. Aufgrund der großen Entfernung zur Erde deckt ein geostationärer Satellit ungefähr ein Drittel der Erdoberfläche ab; demzufolge reichen bereits drei

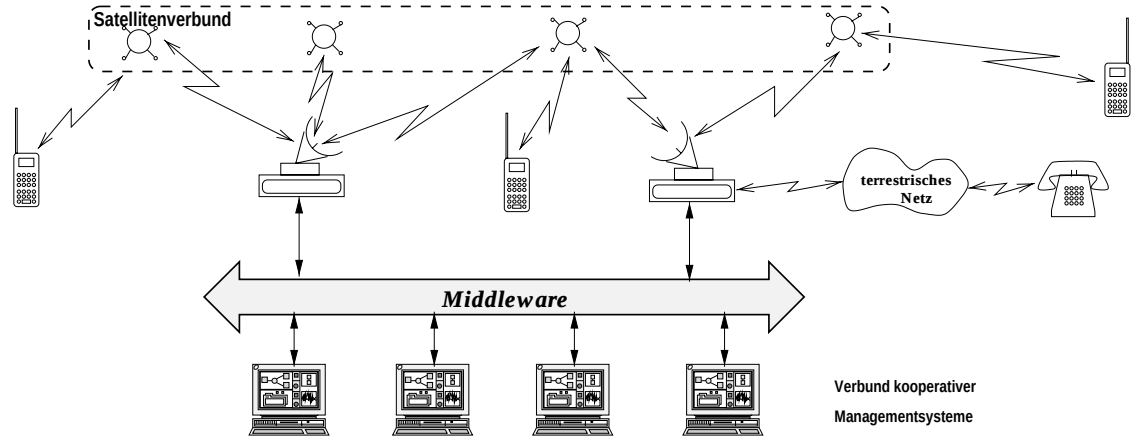
[2] Customer Network Management für das Breitband-Wissenschaftsnetz (B-WIN).

solcher Raumsonden aus, um globale satellitengestützte Kommunikation zu ermöglichen. Desweiteren folgen geostationäre Satelliten (wie es der Name bereits impliziert) der Erdrotation, weshalb lediglich eine geringe Anzahl von Bodenstationen benötigt wird. Um diese Systeme zu überwachen und zu steuern, werden daher ebenfalls wenige Managementsysteme benötigt, die sich in den Bodenstationen befinden und jeweils für das Management eines geostationären Satelliten zuständig sind. Dieses zentralistische Managementkonzept (dargestellt in Abbildung 2.6) ist für die vollständige Überwachung der Kommunikation ausreichend, da innerhalb jeder Bodenstation lediglich eine überschaubare Zahl von Managementobjekten administriert wird.

Neben ihren hohen Kosten haben geostationäre Satelliten jedoch noch weitere Nachteile, die sich insbesondere bei der Telefonie negativ bemerkbar machen: Aufgrund der hohen Entfernungen ergeben sich signifikante Signallaufzeiten (Round-trip delays von 260 ms sind hierbei üblich), die sich negativ auf die Gesprächsqualität auswirken. Ferner verbietet die Notwendigkeit einer sehr hohen Sendeleistung die direkte Verbindung von Mobiltelefonen mit den Satelliten, so daß GEO-basierte Satellitennetze ausschließlich als Transitnetze zum Einsatz kommen [Mill 98].



(a) Zentralistisches Management von GEO-Satelliten



(b) Verteiltes kooperatives Management von LEO/MEO-Satellitennetzen

Abbildung 2.6: Zentralistisches vs. verteiltes kooperatives Management

Zur Umgehung dieser Restrikitionen wurden in den letzten Jahren zwei Ansätze entwickelt, die darauf beruhen, die Satelliten näher in Richtung Erdoberfläche zu positionieren, sodaß sowohl die Verzögerungszeiten verkleinert werden, als auch bereits die geringe Sendeleistung mobiler Telefone (ca. 1 Watt) ausreicht, um unmittelbar mit den Satelliten in Verbindung zu treten. In Abhängigkeit von ihrer Entfernung zur Erde werden solche Satellitensysteme als *Low earth-orbit (LEO)*-Satelliten (500 bis 1500 km Entfernung; Round-trip delays zwischen 10 und 30 ms) oder *Middle earth-orbit (MEO)*-Satelliten (5000 bis 12000 km Entfernung; Round-trip delays ca. 100 ms) bezeichnet. Beispiele für LEO-Satellitensysteme sind Iridium (780 km Entfernung, 66 Satelliten) und Globalstar (1400 km, 48 Satelliten); bei ICO (10354 km, 12 Satelliten) handelt es sich um einen typischen Vertreter eines MEO-Satellitennetzes.

Wie bereits aus den angegebenen Werten hervorgeht, erfordert die relativ geringe Entfernung von LEO/MEO-Satelliten zur Erdoberfläche eine entsprechend hohe Zahl von kleinen Satelliten (und damit prinzipiell auch Bodenstationen), die sich überdies in einer von der Erdrotation unterschiedlichen Geschwindigkeit bewegen. Dies führt dazu, daß die Anzahl von Satelliten, die sich jeweils in Reichweite einer Bodenstation (oder eines Endgerätes) befinden, variabel ist und insbesondere zu einem gegebenen Zeitpunkt mehrere Raumsonden Kommunikationsverbindungen mit einer Bodenstation haben. Eine Konsequenz daraus ist, daß eine Verbindung während ihres Bestehens über unterschiedliche Satelliten geschaltet wird, was wiederum eine dynamische Topologie impliziert. Anforderungen an die Topologiedarstellung von LEO/MEO-Satellitenverbunden wurden in [DNWI 95] untersucht.

Die hohe Dynamik der Kommunikationsbeziehungen und die beträchtliche Anzahl an kleinen Satelliten und Bodenstationen bedingt ebenfalls eine Neuausrichtung des Managements: Anstelle des in GEO-Satellitennetzen verwendeten zentralistischen Managements mit wenigen leistungsfähigen Managementplattformen muß in LEO/MEO-Satelitennetzen ein Management eingeführt werden, bei dem die einzelnen „leichtgewichtigen“ Managementsysteme zusammenwirken. Gefordert ist also eine Managementarchitektur, die der verteilten Natur von Satellitensystemen und Bodenstationen gerecht wird und kooperatives Management (wie es in Abschnitt 2.1.3 definiert wurde) unterstützt. Dies ist nicht zuletzt deswegen vonnöten, als die hier beschriebenen Satellitensysteme sehr hohe Anforderungen an die Verfügbarkeit der (Management-) Systeme haben, was das Vorhandensein geeigneter Replikationsmechanismen erfordert.

Hierbei ist es wichtig, sich auf objektorientierte Architekturen für verteilte Verarbeitung abzustützen, wie sie in Abschnitt 3.1 vorgestellt werden. Diese bilden die Grundlage, um wiederverwendbare Softwarekomponenten zu konstruieren und in eine verteilte Umgebung einzubringen. Außerdem stellen sie nicht nur eine Ablaufumgebung für diese Komponenten bereit, sondern auch geeignete Zugriffsmechanismen, die von den plattformspezifischen Gegebenheiten (Betriebssystem, Systemarchitektur) abstrahieren. Bei der Verwendung portabler Programmiersprachen und virtueller Maschinen ist es ebenfalls möglich, Managementdienste an Agentensysteme zu delegieren, d.h. diese zur Laufzeit um

neue Funktionalität zu erweitern. Insbesondere in der hier vorgestellten Umgebung ist dies von ausgesprochen hoher Wichtigkeit, da einerseits die Größe der auf einzelnen Satelliten realisierbaren Agentensysteme naturgemäß beschränkt ist und – im Gegensatz zu den vorher beschriebenen Szenarien – nach Inbetriebnahme der Systeme (d.h. nach ihrer Positionierung im Orbit) keine manuellen Eingriffsmöglichkeiten mehr bestehen[3]. Das Einspielen einer neuen Version eines Agenten kann daher nur über softwaretechnisch realisierte Kommunikationsmechanismen erfolgen. Hierbei sollten aus Effizienzgründen ebenfalls diejenigen Kommunikationsmechanismen für das Management genutzt werden, die bereits zur Erbringung der Nutzfunktionalität vorhanden sind. Die Tatsache, daß Management „in-band“ erfolgen sollte, impliziert die Abbildung der vier Teilmodelle einer Managementarchitektur auf eine Softwarearchitektur und stellt, wie in Abschnitt 3.1 genauer ausgeführt wird, eine signifikante Einschränkung der Wahlfreiheit dar: Gegenwärtige Managementarchitekturen arbeiten überwiegend nach dem Prinzip des „Outband-Managements“, d.h. sie verwenden eigenständige Protokolle speziell für die Zwecke des Managements.

2.3 Anforderungen an das Management verteilter kooperativer Managementsysteme

Die im vorigen Abschnitt vorgestellten Managementszenarien, die sich aus dem praktischen Betrieb großer Daten- und Telekommunikationsnetze ergeben, führen zu den nachfolgend aufgeführten Feststellungen:

1. In verteilten (Management-) Umgebungen bestehen grundsätzliche Anforderungen an die Darstellung und Übermittlung von Information sowie an die Mechanismen zum Zugriff auf Managementdaten und -dienste. Diese sind oft nicht managementspezifisch, sondern gelten vielmehr für alle Ausprägungsformen verteilter Anwendungen. Sie stellen somit grundlegende Anforderungen an *Implementierungsumgebungen* bzw. Rahmenwerke für verteilte Verarbeitung im allgemeinen dar, auch wenn das Management als spezielle Art einer verteilten Anwendung besonders davon abhängig ist. Wir werden diese Anforderungen in Abschnitt 2.3.1 darlegen.

2. Darauf aufbauend, existieren generelle Anforderungen an das zugrundegelegte architekturelle Management-Rahmenwerk, um verteiltes kooperatives Management in heterogener Umgebung überhaupt realisieren zu können. Wir werden daher in Abschnitt 2.3.2 Kriterien für Enterprise Management Architekturen aufstellen, die uns in Kapitel 3 helfen werden, die gegenwärtigen Managementarchitekturen hinsicht-

[3] Eine ähnliche Situation liegt bei Tiefseekabeln für die Telekommunikation vor, bei denen es notwendig ist, die Funktionsfähigkeit der ebenfalls unter Wasser verlegten Repeater und Verbindungselemente (*branching units*) durch das Management zu überwachen (vgl. hierzu [TrMR 97]).

lich ihrer Eignung für die dieser Arbeit zugrundeliegende Fragestellung kritisch zu überprüfen.

3. Desweiteren muß eine garantierte Mindestmenge an Managementinformation von verteilten kooperativen Managementsystemen bereitgestellt werden, um diese in heterogenen Umgebungen geeignet zu instrumentieren (siehe Abschnitt 2.3.3).

4. Das Vorhandensein verschiedenartiger Dienste zur Überwachung und Steuerung dieser Systeme muß gewährleistet sein. Hierzu gehören auch Dienste zur Delegierung von Managementfunktionalität an Partnersysteme oder Agenten (siehe Abschnitt 2.3.4).

Wir werden im folgenden die aus den Managementszenarien gewonnenen Anforderungen hinsichtlich dieser vier Teilbereiche klassifizieren, um im folgenden Kapitel die bereits bestehenden Rahmenwerke, Ansätze und Produkte unter Bezugnahme auf die dieser Arbeit zugrundeliegenden Fragestellungen kritisch zu untersuchen und zu bewerten. Ferner gewinnen wir durch die Analyse der Anforderungen Anhaltspunkte, welche Managementinformation und -dienste gebraucht werden, die wir in den Kapiteln 5 und 6 spezifizieren, modellieren und implementieren werden.

Zur Vervollständigung der Managementanforderungen haben wir ferner bestehende Objektkataloge verwandter Problembereiche (z.B. für das Anwendungsmanagement) betrachtet und im täglichen Betrieb aufgetretene Fehlersituationen analysiert (Top-down Analyse) sowie den Umfang an Managementinformation realer (Management-) Systeme durch Auswertung der Programmierschnittstellen ermittelt (Bottom-up Analyse).

2.3.1 Anforderungen an die Middleware

Für den Begriff „Middleware“ haben sich unterschiedliche Definitionen etabliert: So definiert die Gartner Group Middleware als Systemsoftware zwischen Anwendungsprogramm und Betriebssystem während das Beratungshaus Ovum unter diesem Begriff jede Art von Connectivity-Software zur Steuerung des Datenflusses zwischen Clients und Servern versteht. Wir werden in Anlehnung an diese beiden Definitionen Middleware als eine verteilte Ablaufumgebung ansehen, die den Rahmen für sämtliche Arten von Interaktion bildet und somit das implementierungstechnische Grundgerüst für Managementarchitekturen bildet.

Management als verteilte Anwendung

In den Abschnitten 2.1.2 und 2.1.3 hatten wir bereits motiviert, weshalb in jüngster Zeit Tendenzen erkennbar werden, Management als eine spezielle verteilte Anwendung zu betrachten. Die in diesem Zusammenhang gestellte Frage „What’s so special about Management?“ [Mazu 98a] bringt diesen Sachverhalt auf den Punkt. Aus dieser Betrachtungsweise ergeben sich folgende Aussagen:

1. Zur Implementierung von Managementsystemen können prinzipiell diejenigen Technologien eingesetzt werden, die bereits für die Entwicklung „gewöhnlicher“ verteilter Anwendungen benutzt werden (vgl. hierzu [ChKo 97], [Bapa 98]). Dies wirft wiederum die Fragestellung auf, ob der heutige Stand dieser Technologien dies zuläßt bzw. in welcher Form Erweiterungen vonnöten sind. Wir werden uns in Kapitel 3 eingehend mit dieser Thematik befassen und eine für unsere Belange geeignete Architektur auswählen.

2. Die Notwendigkeit des Managements verteilter Anwendungen ist heute allgemein anerkannt und ist in die Literatur unter dem Themenkomplex „Anwendungsmanagement“ eingegangen. Neu und bislang von der Fachwelt nicht hinreichend wahrgenommen ist jedoch die Erkenntnis, daß aus dem oben Gesagten insbesondere folgt, daß auch die verteilte Anwendung „**Management**“ selbst überwacht und gesteuert werden muß. Das Managementszenario in Abschnitt 2.2.2 illustriert diesen Sachverhalt unter dem Gesichtspunkt des Fehlermanagements. Die Definition einer Instrumentierung, die dies in heterogener Umgebung leistet, ist das Ziel der vorliegenden Arbeit.

Grundlegende Forderungen

Wir werden nun kurz einige grundsätzliche Forderungen darstellen, die für alle Ausprägungen verteilter Anwendungen (und somit auch für das Management) gelten. Wir orientieren uns dabei an den Festlegungen des ODP-Referenzmodells [ISO 10746], welches wir in Abschnitt 3.1.4 einer genaueren Betrachtung unterziehen werden.

1. **Offenheit** bezeichnet diejenige Kerneigenschaft verteilter Anwendungen, die sowohl die **Portabilität** (d.h. die Ausführbarkeit auf verschiedenen Systemen ohne Vornahme von Modifikationen) von Komponenten einer verteilten Anwendung als auch deren **Kooperation** (d.h. die Ermöglichung von Interaktionen zwischen Komponenten, welche sich unter Umständen auf unterschiedlichen Systemen befinden). Wesentliche Voraussetzungen für dieses Ziel sind das Vorhandensein einer standardisierten Notation zur Definition der Komponentenschnittstellen sowie (ebenfalls standardisierte) Abbildungen dieser Notation auf gängige Programmiersprachen. Die Portabilität der Komponenten erfordert das Vorhandensein einheitlicher Middleware durch die bereits in Abschnitt 2.1.3 angesprochenen virtuellen Maschinen. Die Dokumentenverarbeitung bietet mit der Seitenbeschreibungssprache Postscript sowie den auf zahlreichen Druckern vorhandenen Postscript-Interpretern ein alltägliches Beispiel für den hohen Nutzen virtueller Maschinen, da diese Sprache einerseits von nahezu sämtlichen Textverarbeitungsprogrammen erzeugt und andererseits die virtuellen Maschinen (in Form von Postscript-Interpretern) von den technischen Spezifika der einzelnen Drucker abstrahieren. Aufgrund dieser positiven Eigenschaften spielt Postscript als Datenaustauschformat für komplexe Dokumente – nicht zuletzt im Internet – eine dominierende Rolle.

2. **Integration** definiert die Fähigkeit, verschiedenartige (verteilte) Systeme und Ressourcen unabhängig von ihrer architekturellen Herkunft oder ihrer Leistungsfähigkeit zu einem funktionsfähigen Ganzen zusammenzufassen.

3. **Flexibilität** steht für die Unterstützung der Evolution eines Systems und impliziert die dynamische Anpaßbarkeit an Veränderungen in seinen Außenbeziehungen. Dies schließt ebenfalls die Existenz und den Betrieb bereits existierender älterer Systeme mit ein.

4. **Modularität** beschreibt die strukturelle Aufteilung eines Systems in autonome Komponenten, die jedoch logisch zusammengehören. Sie ist damit eine Schlüsselgröße für die Erweiterbarkeit verteilter Anwendungen.

5. **Föderation** ist die Kombination von Systemen aus unterschiedlichen technischen oder administrativen Domänen zur Erreichung eines gemeinsamen Ziels. Dies schließt insbesondere die Kooperation bestehender Dienste mit dem Ziel der Erbringung eines neuen, höherwertigen Dienstes ein.

6. **Managebarkeit** umfaßt die Möglichkeiten, die Ressourcen eines Systems zu überwachen und zu steuern, mit dem Ziel, vereinbarte Konfigurations-, Dienstgüte- und Abrechnungspolitiken durchzusetzen.

7. **Bereitstellung von Dienstgüte** gestattet die Ermittlung des qualitätsbezogenen Zielerreichungsgrades in Bezug auf das Verhalten eines Systems und betrachtet dabei folgende Kenngrößen: Antwortzeiten, Verfügbarkeit, Zuverlässigkeit, Fehlertoleranz.

8. **Sicherheit** bezeichnet die Gesamtheit aller Mechanismen zum Schutz von Systemressourcen und Daten vor unberechtigtem Zugriff; diese umfassen unter anderem: Zugangskontrolle, Protokollierung, Authentifizierung, Integrität, Schlüsselverwaltung.

9. **Transparenz** schließlich zählt ebenfalls zu den zentralen Eigenschaften verteilter Systeme. Die unterschiedlichen Arten von Transparenz werden nachfolgend erläutert.

Verteilungstransparenz

Verteilungstransparenz beschreibt die Fähigkeit, sowohl Entwickler und Benutzer als auch die Komponenten einer verteilten Anwendung selbst von der Komplexität und den technischen Details verteilter Systeme abzuschirmen. Man verfolgt hierbei das Prinzip des „Information hiding“, um eine homogene Sicht auf das verteilte System zu gewährleisten, d.h. die Verteiltheit des Systems wird nicht sichtbar. Für Entwickler bedeutet Verteilungstransparenz, daß die Implementierung verteilter Anwendungen auf dieselbe Art erfolgen kann, wie die gewöhnlicher Applikationen. Das ODP-Referenzmodell definiert in [ISO 10746-1] insgesamt acht verschiedene Arten von Transparenz, die wir aufgrund ihrer Relevanz für die verteilte Anwendung „Management“ bereits an dieser Stelle einführen. Hierbei wird

für ODP-konforme verteilte Systeme keinesfalls die vollständige Umsetzung aller Transparenzen gefordert, vielmehr sollte eine selektive Auswahl einzelner Transparenzen in Abhängigkeit des Anwendungsbereichs erfolgen. Die beiden erstgenannten Ausprägungsformen von Transparenz bilden hierbei den Grundstock, auf dem die sechs anderen Arten aufbauen.

1. **Zugriffstransparenz** (*Access transparency*) abstrahiert von den Arten der Datenrepräsentation sowie der Aufrufmechanismen von Prozeduren, um eine Zusammenarbeit von Applikationen, die auf heterogenen Systemen ablaufen, überhaupt zu ermöglichen.
2. **Ortstransparenz** (*Location transparency*) zählt – zusammen mit der erstgenannten – zu den wohl wichtigsten Ausprägungsformen von Transparenz. Sie verdeckt den Ort, d.h. das konkrete System, an dem eine Komponente einer verteilten Anwendung sich befindet und gestattet so die Verwendung logischer anstelle von physikalischer Adressierung.
3. **Migrationstransparenz** (*Migration transparency*) baut auf dem Konzept der Ortstransparenz auf und bedeutet, daß eine Komponente von einem System zu einem anderen transferiert werden kann. Dieses Prinzip ist die Grundvoraussetzung für das in Abschnitt 3.2.1 beschriebene *Management by Delegation*.
4. **Relokationstransparenz** (*Relocation transparency*) ist erforderlich, wenn die Migration einer Komponente die zu diesem Zeitpunkt offenen Kommunikationsbeziehungen zu anderen Komponenten erhalten soll.
5. **Fehlertransparenz** (*Failure transparency*) verschattet das Auftreten sowie die Mittel zur Behebung eines Fehlers mit dem Ziel, fehlertolerante Systeme zu realisieren.
6. **Replikationstransparenz** (*Replication transparency*) impliziert die Möglichkeit, eine Komponente auf identische Art zu vervielfachen sowie die Existenz geeigneter Mechanismen zur Konsistenzsicherung.
7. **Persistenztransparenz** (*Persistence transparency*) abstrahiert vom Aktivierungszustand einer Komponente indem geeignete Aktivierungs- bzw. Deaktivierungsmechanismen zur Vefügung gestellt werden.
8. **Transaktionstransparenz** (*Transaction transparency*) verdeckt die Koordinationsmechanismen zur Erreichung der ACID-Eigenschaften[4] für Komponenten verteilter Anwendungen.

Designbezogene Anforderungen

Allgemeine Anforderungen an das Design verteilter Systeme umfassen neben den oben ganannten Kriterien auch Eigenschaften wie einfache Modellierbarkeit und Implementierbarkeit. Als zweckmäßig hat sich in diesem Zusammenhang die Objektorientierung mit

[4] **A**tomicity, **C**onsistency, **I**solation, **D**urability.

den Kerneigenschaften Datenabstraktion, Kapselung, Polymorphie und Vererbung erwiesen. Diese Eigenschaften sind insbesondere hilfreich bei der Definition geeigneter Basisklassen der Vererbungshierarchie, da diese einerseits möglichst generisch sein, andererseits auch eine hinreichende Informationsmenge bereitstellen sollten.

2.3.2 Kriterien für Enterprise Management Architekturen

Zusätzlich zu den im vorigen Abschnitt beschriebenen grundsätzlichen Aspekten ergeben sich aufgrund der hohen Anforderungen an das Management weitere Bedingungen, denen eine Managementarchitektur genügen muß, um als Basis für das Enterprise Management zu dienen. Diese sind nachstehend aufgeführt:

1. Der **zuverlässige Datentransfer zwischen Managementsystemen**: Die zwischen Managerinstanzen ausgetauschten Nachrichten sind, wie das Szenario in Abschnitt 2.2.1 aufzeigt, aufgrund ihrer hohen Informationsdichte von großer Wichtigkeit. Es muß daher gewährleistet sein, daß nicht nur Mechanismen zur Manager-to-Manager Kommunikation bestehen, sondern asynchrone Ereignismeldungen von einem Managementsystem zu einer Partnerinstanz auch zuverlässig ihr Ziel erreichen; alternativ zur Anwendung von (meist relativ aufwendigen) Verfahren, die eine „exactly-once"-Semantik besitzen, kann auch auf Bestätigungsmechanismen zurückgegriffen werden.

2. Die **effiziente Übertragung großer Datenmengen**: Oft müssen sowohl zwischen unterschiedlichen Managern als auch zwischen Managern und Agenten größere Datenmengen ausgetauscht werden. Beispiele hierfür sind das Auslesen der auf einem System eingetragenen Benutzer oder die zum gegenwärtigen Zeitpunkt aktiven Prozesse sowie deren Kenndaten. Letzteres kann, wie Leistungsmessungen an einem von uns entwickelten Managementagenten für UNIX-Systeme[5] ergeben haben, für 100 bis 150 Prozesse zwischen 30 und 45 Sekunden in Anspruch nehmen. Hiervon benötigte die Ermittlung der Managementdaten durch den Agenten sowie deren Übergabe an die Protokollmaschine knapp eine Sekunde; der Engpaßfaktor war hier eindeutig das verwendete Managementprotokoll (im vorliegenden Fall: SNMP). Folglich müssen die Kommunikationsmechanismen darauf ausgerichtet sein, einen zügigen Transfer auch großer Datenmengen zu gewährleisten. Das Szenario in Abschnitt 2.2.5 liefert hierfür ein weiteres praxisnahes Beispiel.

3. Die **Skalierbarkeit** hinsichtlich sehr großer Mengen an Managementobjekten: Das in Abschnitt 2.2.3 beschriebene Szenario verdeutlicht, daß beispielsweise für den Austausch von Abrechnungsdaten zwischen Telekom-Carriern hinsichtlich der Nutzungsdauer geschalteter Verbindungen zahlreiche, sehr feingranulare Managementobjekte, berücksichtigt werden müssen. Überdies sind diese Managementobjekte, die jeweils eine Verbindung repräsentieren, äußerst dynamischer Natur.

[5] Wir werden diesen Agenten in Abschnitt 5.1.2 vorstellen.

4. Das Bestehen von **Sicherheitsmechanismen** gegen potentielle Angriffe: Das in Abschnitt 2.2.2 aufgeführte Managementszenario hat demonstriert, wie durch unbeabsichtigte Fehlkonfiguration von Managementsystemen in Verbindung mit fehlerhafter Protokollsoftware hohe Kosten entstehen können. Ein bösartiges Ausnutzen dieser Sicherheitsproblematik ist verhältnismäßig unkompliziert und kann beträchtliche Schäden verursachen. Es ist daher notwendig, daß bereits in der Managementarchitektur selbst Mechanismen vorgesehen sind, die die üblichen Bedrohungen (Maskerade, Mitlesen, Modifikation, Verzögerung und Vervielfältigung) verhindern können. Dienste, die diese Mechanismen realisieren, können – je nach Bedarf – bereits von der zugrundeliegenden verteilten Umgebung übernommen, gegebenenfalls verfeinert und an die Bedürfnisse des Managements angepaßt werden.

5. Das Vorhandensein von **Logging-Mechanismen** zur Protokollierung außergewöhnlicher Ereignisse ist nicht nur aus Sicherheitsgründen zwingend erforderlich, sondern ist auch im Fehlerfalle von großer Wichtigkeit, um Rückschlüsse auf eventuelle Fehlerursachen ziehen zu können. Nach der Beseitigung des Fehlers sollte es möglich sein, den Fehlerbehebungsvorgang zu dokumentieren und mit den protokollierten Fehlersymptomen zu verknüpfen, um beim Auftreten ähnlicher Fehler bereits über Ansätze zur Behebung des Problems zu verfügen.

6. Die **Verteilung von Managementaufgaben** auf mehrere Managementsysteme bzw. die Delegierung von Aufgaben an Agentensysteme sind, wie es bereits in Abschnitt 2.1.3 angesprochen wurde, notwendig, um die Skalierbarkeit des Managements auch für sehr große Kommunikationssysteme zu gewährleisten. Eine Architektur für das Enterprise Management sollte über geeignete Mechanismen zum Transfer von Managementfunktionalität verfügen, der sowohl von den Managementsystemen als auch von den Agenten initiierbar sein sollte. Letzteres impliziert eine hohe Autonomie der Agentensysteme, die situationsbezogen von sich aus entscheiden, welche Art von Managementdiensten erforderlich ist und diese aus einem Pool von Diensten beziehen können. Solche Systeme werden als **Selbststeuernde Systeme** (*Self-managed Systems*) bezeichnet; sie befinden sich jedoch überwiegend noch im Stadium der Forschung [WeAu 96].

7. Die Möglichkeit der **Abfrage von Metainformationen** (sog. Management Knowledge oder Kontext-Wissen) über vorhandene Managementsysteme bzw. Agenten zählt ebenfalls zu den wichtigen Managementdiensten, da so beispielweise zur Laufzeit ermittelt werden kann, welche MIBs von einem gegebenen Agenten implementiert werden. Dies ist besonders hilfreich bei der Initialkonfiguration eines Managementsystems, das sich so bereits bei der Autodiscovery des Netzes Detailinformationen bezüglich der darin befindlichen Ressourcen, Systeme und Anwendungen beschaffen kann. So könnten aufgrund der Tatsache, daß ein Managementsystem Router erkennen kann, deren Routing-Tabellen über das Management ausgelesen werden um somit präzise Informationen bezüglich der Netzstruktur zu gewinnen. Bisher müssen

die MIBs aller zu administrierenden Ressourcen durch den Administrator in die Plattformen eingespielt werden, damit diese die über den Minimalumfang hinausgehende Ressourceninformation überhaupt nutzen können. Eine andere Anwendung besteht in der Ermittlung derjenigen Agenten, welche gegenwärtig von einem bestimmten Managementsystem überwacht werden oder welche Managementdienste überhaupt zur Vefügung stehen.

8. Der **Reifegrad** einer Managementarchitektur ist oftmals das ausschlaggebende Auswahlkriterium. Dies kann sich einerseits auf den Status des Standardisierungsprozesses beziehen, der ein Maß für die Vollständigkeit der technischen Vision und die universelle Einsetzbarkeit ist. Andererseits ist ein hoher Grad an Marktdurchdringung – der oft als Synonym für Investitionssicherheit gilt – ein wichtiger Indikator für den Reifegrad einer Architektur.

9. Während die **Verfügbarkeit leistungsfähiger Entwicklungswerkzeuge** primär kein architekturbezogenes Kriterium zu sein scheint, so hat dies doch großen Einfluß auf die Auswahl einer Architektur: Die Einfachheit der Implementierung, die überwiegend an die Verfügbarkeit von Entwicklungswerkzeugen gekoppelt ist, schlägt sich im Vorhandensein einer hohen Zahl implementierter architekturspezifischer Managementdienste nieder. Im Zweifelsfall wird immer diejenige Architektur bevorzugt, deren eingeschränkter Dienstumfang durch eine vollständige Implementierung kompensiert wird, gegenüber einer Architektur mit einer großen Menge spezifizierter Dienste, die ihrerseits jedoch nur unvollständig am Markt erhältlich sind. So hatte das OSI-Management lange Zeit den Ruf, eine nur unter großen Komplikationen implementierbare Architektur zu sein, weil einerseits lediglich eine sehr komplexe Programmierschnittstelle für das verwendete Managementprotokoll zur Verfügung stand und andererseits nur eine geringe Zahl an spezifizierten Diensten auch tatsächlich implementiert wurde. Nicht zuletzt deswegen konnte es sich im Bereich lokaler Netze nie durchsetzen.

10. Die **einfache Abbildbarkeit der Notationen der Informationsmodelle auf Implementierungssprachen** zählt ebenfalls zu den wichtigen Faktoren, von denen die flexible Einsetzbarkeit einer Managementarchitektur maßgeblich abhängt. Wünschenswert ist das Vorhandensein von Abbildungsvorschriften auf eine möglichst hohe Zahl von Implementierungssprachen, um eine Migration proprietärer Managementlösungen hin zu offenen Managementarchitekturen zu vereinfachen.

2.3.3 Erforderliche Managementinformation

Nachdem wir prinzipielle Kriterien für verteilte Umgebungen sowie Enterprise Management Architekturen aufgestellt haben, werden wir uns nun der Gewinnung von Anforderungen zuwenden, die auf einen speziellen Problembereich des Managements abzielen, nämlich den der Überwachung und Steuerung verteilter kooperativer Managementsyste-

me. Die hierbei gewonnenen Parameter werden uns bei der Definition geeigneter Objektmodelle für das Management verteilter kooperativer Managementsysteme behilflich sein. Dies ist notwendig, weil in manchen Managementarchitekturen zwar Mechanismen zur Kommunikation zwischen Managementsystemen vorhanden sind, jedoch bisher keine genauen Vortellungen bestehen, welche Arten von Managementinformation übermittelt bzw. welche Aktionen auf einem Partnersystem ausgeführt werden sollten. Wir werden daher in diesem Abschnitt zunächst ermitteln, welche Managementinformation ein verteiltes kooperatives Managementsystem bereitstellen sollte und im folgenden Abschnitt die benötigten Managementdienste identifizieren[6].

Grundlegende Annahmen

Managementsysteme sind heutzutage auf gewöhnlichen Mehrbenutzer-Betriebssystemen wie UNIX oder Windows NT ablauffähig und ihre Funktionsweise ist daher unmittelbar von Fehlern abhängig, die sich im Betriebssystem selbst ereignen oder in der von Netzdiensten und -protokollen bereitgestellten Kommunikationsinfrastruktur auftreten. Beispiele aus der Internet-Welt für letztere sind das *Domain Name System (DNS)*, das *Network File System (NFS)* oder das *Network Information System (NIS)*. Wie wir in Abschnitt 3.3.1 sehen werden, besteht der Großteil an Managementinstrumentierung kommerzieller Managementsysteme darin, die Grundfunktionalität der Betriebssystem- und Kommunikationsinfrastruktur zu überwachen. Während dies zweifellos wichtig für die Eingrenzung aufgetretener Fehler ist, so sind derlei Informationen nicht unmittelbare Bestandteile einer Managementinformationsbasis für verteilte kooperative Managementsysteme. Wir werden daher im folgenden lediglich Managementinformation aufführen, die unmittelbare Relevanz für die Instrumentierung von Managementsystemen hat. Managementinformation wie Parameter zur Instrumentierung von Betriebssystemen (CPU-Belastung, Quotenzuteilungen für Betriebsmittel, Plattenplatz usw.), Kommunikationsinfrastruktur (Feststellung der Konnektivität, Timeoutwerte, Puffergrößen, Paketfehler usw.) und Netzdiensten (wie die oben genannten Beispiele) werden wir aus diesem Grunde hier nicht behandeln. Es geht uns in erster Linie darum, *Managementinformation verteilter kooperativer Managementsysteme* zu identifizieren, also diejenigen Ressourcenparameter, die wichtige Informationen bezüglich der überwachten Managementsysteme liefern und an denen gegebenenfalls Modifikationen vorgenommen werden sollen.

Sichtweisen auf Managementinformation

Auch für verteilte kooperative Managementsysteme muß ein Grundumfang an generischer Managementinformation bereitgestellt werden, wie beispielsweise der Name des Herstellers, die Produktbezeichnung und die Versionsnummern, der Zeitpunkt der Installation, der Mindestbedarf an Betriebsmitteln (Plattenplatz, Hauptspeicher) und Angaben

[6] Aus Platzgründen werden wir uns hierbei auf charakteristische Beispiele beschränken; Detailinformationen werden in Kapitel 5 vorgestellt.

zur (Un-)Verträglichkeit mit Betriebssystemversionen. Dies spiegelt die Sichtweise auf ein Managementsystem unter dem Aspekt der Grundkonfiguration, d.h. die **funktionale** Betrachtung des Systems als installierbares Softwarepaket wider, welches seinerseits aus einzelnen Komponenten besteht. Eine andere, **strukturelle** Sichtweise besteht darin, das Managementsystem als ausführbares Programmsystem zu betrachten, also als Menge von Dateien, die entweder die Managementsoftware selbst oder Konfigurationseinstellungen, Hilfefunktionen und Benutzerdaten enthalten. Hierfür sind Informationen wie Installationspfade, Dateiattribute (geschützt, lesbar, schreibbar) und die jeweiligen Zugriffsrechte sowie die Eigentümer mitsamt ihrer Gruppenzugehörigkeit relevant. Eine dritte, **operationelle** Betrachtungsweise behandelt das System zur Laufzeit, hier als Menge von Prozessen (bzw. Tasks), die jeweils Rechenzeit und Speicherplatz in Anspruch nehmen. Adäquate Managementinformation enthält in diesem Fall neben den beiden vorher angegebenen Parametern die Prozeßeigentümer sowie Statusangaben (aktiv, wartend, suspendiert). Unter diese Sichtweise fallen ebenso Zähler für entgegengenommene, bearbeitete und zurückgewiesene Anfragen (möglichst unter Angabe damit verbundener Gründe). Hierbei wird deutlich, daß jede der drei Betrachtungsweisen für unsere Belange relevant ist, da diese jeweils bestimmte eigene Arten von Managementinformation liefern.

Informationen für das Konfigurationsmanagement

Neben den vorher beschriebenen Parametern muß ein verteiltes kooperatives Managementsystem anderen Partnersystemen eine umfassende Menge an dynamischen Konfigurationsparametern zur Verfügung stellen. Diese umfassen beispielsweise das Auslesen und Setzen von Pollingintervallen und das Kreieren und Löschen von Managementobjekten in der Datenbank des Systems; solche weitreichenden Eingriffsmöglichkeiten sind notwendig, um einerseits hohen Netzbelastungen durch zu kurze Pollingintervalle vorzubeugen und andererseits die Datenbestände der einzelnen am Managementprozeß beteiligten Systeme konsistent zu halten. Das Setzen von Parametern zur Konfiguration der Ereignisweiterleitung und -filterung ist ebenfalls relevant, da es Partnersystemen möglich sein sollte, sich für bestimmte an diesem System auftretende Ereignisse zu registrieren.

Informationen für das Fehlermanagement

Wichtige Parameter für das Fehlermanagement umfassen Angaben über aufgetretene Fehlertypen sowie deren Häufigkeit, die Zeit seit der letzten Initialisierung des Managementsystems sowie der gegenwärtige Zustand der einzelnen Systemkomponenten. Fehlerlogs, die vom Managementsystem angelegt wurden, sollten von anderen Systemen auslesbar (und gegebenenfalls löschbar) sein, um die Erstellung einer globalen Netzhistorie zu vereinfachen. Zur Wiederherstellung des Managementsystems nach einem Ausfall sollten geeignete Backup- und Recoverymechanismen verfügbar sein. Das Managementszenario aus Abschnitt 2.2.1 illustriert die Wichtigkeit solcher Maßnahmen.

Informationen für das Leistungsmanagement

Zusätzlich zu den weiter oben definierten Parametern wie die Anzahl entgegengenommener, bearbeiteter und zurückgewiesener Anfragen sollte ebenfalls die aktuelle Auslastung des Systems abgerufen werden können. Zähler zur Ermittlung der Bearbeitungszeiten von Aufträgen oder die Länge der Auftragswarteschlangen geben ebenfalls Aufschluß über das Gesamtverhalten des Systems und liefern Anhaltspunkte hinsichtlich des Nutzens der Aufstellung weiterer Managementsysteme.

Aus diesen Grunddaten kann anschließend aussagekräftigere Managementinformation abgeleitet werden, die oft von den betreiberspezifischen Zielvorgaben (sog. *Policies*) abhängt (s.u.). Beispiele hierfür sind die Korrelation der Basiswerte mit Zeitintervallen oder die Bildung von Durchschnittswerten.

Informationen für das Abrechnungsmanagement

Die Tatsache, daß die Lizenzüberwachung seit geraumer Zeit auch auf Managementsysteme ausgedehnt wurde, bedingt die Bereitstellung entsprechender Information hinsichtlich der Art der Lizenz (nodelocked, floating), die maximale sowie die gegenwärtige Zahl von Benutzern des Managementsystems, die (reale und maximal zulässige) Menge administrierter Ressourcen. Bei Überschreitung bestimmter Schwellwerte sollten Ereignismeldungen ausgelöst werden, die den Bedarf an weiteren Lizenzen signalisieren.

Für die verbrauchsbezogene Abrechnung von Managementdiensten, wie es das in Abschnitt 2.2.3 beschriebene Szenario fordert, ist es notwendig, festzuhalten, welche Aktionen von welchen Partnersystemen initiiert wurden und wie lange diese zur Ausführung benötigt haben, bzw. welche Datenmenge jeweils dabei angefallen ist. Die Vorgaben, welches Abrechnungsmodell (volumenorientiert, subjektbezogen, objektbezogen, pauschal) letztendlich angewandt wird, ist primär von den unternehmensweiten Zielvorgaben abhängig; jedoch sollten alle möglichen Modelle durch die Bereitstellung einer hinreichend großen Menge an Abrechnungsdaten unterstützt werden können.

Informationen für das Sicherheitsmanagement

In diese Kategorie fallen bei verteilten Managementsystemen insbesondere Informationen zur Speicherung von domänenbezogenen Daten (z.B. welchen Domänen ein System zugeordnet ist und welche Partnersysteme jeweils darin enthalten sind) und Definitionen von Zuständigkeitsbereichen (z.B. aus welchen Domänen Informationen von Agentensystemen ermittelt bzw. modifiziert werden dürfen). Das in Abschnitt 2.2.2 beschriebene Managementszenario liefert hierfür ein anschauliches Beispiel. Im einzelnen muß desweiteren definiert werden können, welche Aktionen das betroffene Managementsystem auf Partnersystemen initiieren darf (*Capabilities* bzw. *Access Control*). Andere Sicherheitsanforderungen betreffen Middleware und Managementarchitekturen als Ganzes und werden daher an dieser Stelle nicht explizit behandelt.

Die genaue Spezifikation und Modellierung dieser aus den Szenarien in Abschnitt 2.2 gewonnenen Managementinformation erfolgt in Kapitel 5.

Zustandsbehaftete und abgeleitete Managementinformation

Bisher liegt unseren Untersuchungen eine zustandslose Betrachtungsweise zugrunde, d.h. zu jedem Abfragezeitpunkt wird die jeweils aktuelle Wertebelegung der Managementinformation zurückgeliefert, ohne die Ergebnisse vorhergehender Abfragen mit einzubeziehen. Gerade für das Leistungsmanagement ist es jedoch wichtig, Zeitreihenanalysen und Trendvorhersagen zu erstellen und dabei auf über (unter Umständen dynamisch) festgelegte Zeiträume gemessene Durchschnittswerte zuzugreifen, wie die Zahl bearbeiteter Anfragen pro Stunde. Es ist daher ebenso notwendig, neben zustandsloser auch **zustandsbehaftete Managementinformation** in die Betrachtung mit einzubeziehen: Die aus Sicherheitsgründen notwendige Prüfung einer maximalen Anzahl zurückgewiesener Anfragen muß ebenso ermittelbar sein, wie davon **abgeleitete Managementinformation** : Die Zahl zurückgewiesener Anfragen pro Zeiteinheit (Minuten, Stunden, Tage) liefert ebenfalls wichtige Aussagen hinsichtlich potentieller Manipulationsversuche oder einer möglichen Fehlkonfiguration eines Partnersystems.

Diese Beispiele zeigen jedoch ebenfalls auf, daß zustandsbehaftete und abgeleitete Managementinformation extrem von den Anforderungen und den Zielvorgaben des jeweiligen Systembetreibers abhängen und a priori nicht ermittelbar sind: So wird sich ein Betreiber möglicherweise mit stündlich ermittelten Durchschnittswerten für zurückgewiesene Anfragen zufrieden geben, während andere aufgrund ihrer Zielvorgaben auf Auswertungen im Minutentakt angewiesen sind. Bei Verknüpfung mehrerer (u.U. abgeleiteter) Parameter wird deutlich, daß die Menge an möglichen Kombinationen nicht mehr unmittelbar durch die Instrumentierung eines Systems geliefert werden kann, sondern hierfür geeignete Managementdienste gefordert sind, die individuell konfigurierbare Verknüpfungen von Rohdaten anbieten. Es ist daher sinnvoll, diese Auswertungsmöglichkeiten von der eigentlichen Instrumentierung zu trennen und in Form *abgesetzt implementierter Managementdienste* zu realisieren. Wir werden im folgenden einige Beispiele solcher Managementdienste vorstellen.

2.3.4 Benötigte Managementfunktionalität

Die im vorigen Abschnitt identifizierte Managementinformation gestattet zwar grundsätzlich das Management verteilter kooperativer Managementsysteme, jedoch bezieht sich dieses im wesentlichen auf das Auslesen bzw. Modifizieren einzelner Parameter. Demgegenüber muß es möglich sein, auch Aktionen auf den Ressourcen zu definieren. Dies geschieht durch die Festlegung von **Managementfunktionalität**, die entweder unmittelbar bei der Ressource selbst angesiedelt ist, oder, wenn diese Dienste für ein breites Ressourcenspektrum relevant sind, auch in Form eigenständig implementierter **Managementdienste** realisiert werden kann.

Dienste für das Konfigurationsmanagement

Wie die in den Abschnitten 2.2.1 und 2.2.3 beschriebenen Managementszenarien verdeutlichen, ist es von fundamentaler Bedeutung, Dienste zur Bildung von Domänen in Verbindung mit geeigneten Zugriffsrechten vorzusehen, um die Zuständigkeitsbereiche von Managementsystemen geeignet abgrenzen zu können. Um innerhalb einer Domäne Aktionen auf mehreren Managementobjekten simultan auszuführen, benötigt man ebenfalls Dienste zur Gruppenbildung, deren Kriterien flexibel definierbar sein sollten und unter Umständen von den Zielvorgaben des Betreibers abhängen. Die Zuweisung von Konfigurationsprofilen für Managementsysteme beispielsweise hinsichtlich der Pollingintervalle und chronisch auszuführender Managementaktivitäten (*Scheduling*) bedingen ebenfalls geeignete Konfigurationsdienste. Ferner ist es wichtig, Dienste zur Ermittlung der gegenwärtig von einem Managementsystem administrierten Ressourcen bereitzustellen.

Dienste für das Fehlermanagement

Für Partnersysteme muß es möglich sein festzustellen, ob das gesamte Managementsystem bzw. wichtige Komponenten (Topologieverwaltung, Monitoring-Komponente) ausgefallen sind. Die Ermittlung eines Ausfalls erfolgt durch Diagnosemechanismen zur Durchführung von Tests und Traces auf dem betroffenen System und muß von diesem angeboten werden. Ist ein Ausfall aufgetreten, sollte ein Partnersystem in der Lage sein, das Management der durch das ausgefallene Managementsystem administrierten Ressourcen zu übernehmen (vgl. hierzu die *Manager Overtake* Funktion aus Szenario 2.2.1); die im vorigen Abschnitt besprochenen Dienste zur Ermittlung dieser Ressourcenmenge bilden hierfür die Grundlage. Ferner sollte aus den obengenannten Gründen ein Partnersystem in der Lage sein, ein gesamtes Managementsystem bzw. dessen einzelne Teilkomponenten zu initialisieren oder zu stoppen oder einen Restart zu veranlassen.

Im praktischen Betrieb von Managementsystemen zeigt sich, daß ein häufiger Grund für den Ausfall eines Managementsystems im Vorliegen von Inkonsistenzen in seiner Datenbank liegt. Solche Inkonsistenzen breiten sich über verhältnismäßig lange Zeiträume (oft mehrere Tage) aus und führen schließlich zum Kollaps des Gesamtsystems. Das regelmäßige Prüfen der Datenbank auf Konsistenz ist hierbei ein Schlüsselfaktor zur Erreichung einer hohen Systemstabilität, kann jedoch bei gegenwärtigen Managementsystemen lediglich offline, d.h. dann durchgeführt werden, wenn das System selbst inaktiv ist. Auch aus diesem Grund ist der oben angesprochene Dienst zum Stoppen bzw. Starten eines Managementsystems notwendig.

Dienste zur Ereignisfilterung, -verarbeitung und -weiterleitung sind ebenfalls wesentlich, um lokalen Netzfehlern mit globalen Auswirkungen wirksam vorbeugen zu können. Ein Enterprise Management System kann sich so gezielt über an einem verteilten kooperativen Managementsystem auftretende Ereignisse informieren und gegebenenfalls Test- und Fehlerbehebungsmaßnahmen anstoßen. Hierzu zählt auch das Verwalten, Auswerten und

Bereinigen der Ereignis- und Fehlerlogs eines Managementsystems, das ebenfalls durch Partnersysteme möglich sein sollte.

Dienste für das Leistungsmanagement

Im letzten Teil des Abschnitts 2.3.3 wurde einerseits motiviert, daß die Ermittlung zustandsbehafteter sowie abgeleiteter Managementinformation (wie z.B. die Erstellung von Zeitreihenanalysen und die Bildung evtl. gewichteter Durchschnittswerte) einerseits äußerst wichtig ist, da diese wesentlichen Anteil an einer betreibergerechten Aufbereitung von Managementinformation hat. Andererseits verbietet die Vielfalt der Ausprägungsformen eine unmittelbare Bereitstellung solcher betreiberorientierter Managementinformation durch die Ressourcen selbst. Folglich muß diese wichtige Managementinformation durch flexibel konfigurierbare Managementdienste angeboten werden, die geeignete Parameter sowohl für das Leistungs- als auch für das Fehlermanagement implementieren.

Dienste für das Abrechnungsmanagement

Die Abrechenbarkeit von Managementdienstleistungen ist eine grundlegende Voraussetzung für das Outsourcing von Managementaktivitäten an externe Dienstleister. Dies erfordert die Definition und Durchsetzung betreiberspezifischer Abrechnungspolicies auf Managementsystemen. Es muß daher möglich sein, die im vorigen Abschnitt identifizierte Managementinformation auf Dienstkataloge abzubilden, die eine Bewertung hinsichtlich betriebswirtschaftlicher Kriterien zulassen. Naturgemäß spielt auch hier das Vorhandensein abgeleiteter Managemeninformation eine tragende Rolle.

Dienste für das Sicherheitsmanagement

Hierunter fallen Dienste zum Eintragen bzw. Löschen von Partnersystemen sowie deren Rechte in die Zugriffskontrollisten eines verteilten kooperativen Managementsystems. Diese Funktionalität sowie die damit zusammenhängenden Dienste zur Prüfung von Berechtigungen sind universeller Natur, so daß sie in der Regel entweder bereits von der Middleware oder von der Managementarchitektur bereitgestellt werden müssen. Die Notwendigkeit der Implementierung sicherheitsspezifischer Analysedienste, deren Ziel die Gewinnung (nach unterschiedlichen Kriterien) abgeleiteter Managementinformation aus den von der Instrumentierung bereitgestellten Managementdaten ist, wurde bereits weiter oben begründet.

Metadienste: Auffinden, Vermitteln und Delegieren von Diensten

Die in den vorangehenden Teilabschnitten identifizierten Managementdienste laufen grundsätzlich in verteilten Umgebungen ab und sind damit prinzipiell an keinen festen Ausführungsort gebunden. Dies wirft zwangsläufig mehrere fundamentale Probleme auf, die zum überwiegenden Teil bereits von der Middleware gelöst werden müssen:

- An erster Stelle ist hier das Problem einer **eindeutigen Adressierung und Namensgebung** zu nennen, das in verteilten Managementumgebungen überdies systemübergreifend und ortstransparent gelöst sein muß. Ferner ist der Umfang eines verteilten Managementsystems (d.h. die in ihm enthaltenen Dienste und Ressourcen) weder a priori ermittelbar, noch bleibt dieser über längere Zeiträume konstant: Zu jedem Zeitpunkt können sowohl neue Ressourcen in das System eingefügt (bzw. daraus entfernt) als auch neue Dienste angeboten (bzw. gelöscht) werden. Dies impliziert insbesondere die Anforderung einheitlicher Formate für Namen und Adressen.
- Um der Dynamik eines verteilten Managementsystems gerecht zu werden, sind **Such- und Verzeichnisdienste** erforderlich, die es gestatten, die vorhandenen Dienste hinsichtlich *syntaktischer* Kriterien (wie z.B. Eigenschaften ihrer Signatur) zu ermitteln, um diese anschließend zu adressieren.
- Während die vorgenannten Dienste grundsätzlich das Auffinden und Adressieren sowie die Nutzung von Diensten in verteilten Umgebungen erlauben, bietet die Auswertung lediglich syntaktischer Eigenschaften von Diensten oftmals nicht das gewünschte Ergebnis: Schließlich ist man weniger an Diensten interessiert, deren Schnittstelle zu einem bestimmten Suchmuster paßt, sondern möchte diese vielmehr nach ihren *Eigenschaften* auswählen. Dies leisten auf den obengenannten Diensten aufbauende **Vermittlungsdienste** (sogenannte *Trader*), die die Dienstauswahl nach *semantischen* Kriterien gestatten.

Eine signifikante Erweiterung der Funktionalität verteilter Managementsysteme besteht darin, die auf wenigen, ausgezeichneten Systemen befindlichen Managementdienste nicht nur global nutzbar zu machen, sondern diese nach Bedarf auch *zur Laufzeit* auf andere Systeme zu verlagern. Man spricht in diesem Zusammenhang von der *Delegierung* von Managementaufgaben an andere Systeme. Natürliche Kandidaten für solche delegierbaren Dienste sind die oben besprochenen Dienste zur Ermittlung abgeleiteter, betreiberspezifischer Managementinformation, die auf einem System geeignet konfiguriert und auf einem anderen System zur Ausführung gebracht werden.

Naheliegende Delegierungsmodelle bestehen in der von Managementsystemen ausgehenden Erweiterung von Agenten sowie im Transfer von Managementfunktionalität zwischen Managementsystemen nach dem *Push-Modell*. Denkbar sind jedoch auch Ansätze, die es individuellen (Manager- oder Agenten-)Systemen ermöglichen, selbständig das Laden geeigneter Dienste nach dem *Pull-Modell* zu initiieren. Die hierfür notwendigen technischen Voraussetzungen in Bezug auf die dieser Arbeit zugrundeliegende Fragestellung werden wir in Kapitel 6 diskutieren.

2.4 Zusammenfassung

Während die im vorigen Abschnitt anhand einer Top-down-Vorgehensweise identifizierte Managementinformation und -funktionalität naturgemäß **technischer** Art ist, fließen in der Praxis bei der Auswahl geeigneter Managementsysteme weitere Kriterien in den Analyseprozeß ein, die im wesentlichen auf die Betreiberanforderungen abzielen. Wir haben dies in einem von uns entwickelten Kriterienkatalog dokumentiert, dessen Ziel darin besteht, Netzbetreiber bei der Evaluierung existierender Managementsysteme zu unterstützen. Im einzelnen haben wir folgende vier Hauptkategorien identifiziert, die in insgesamt 23 Unterkategorien verfeinert wurden. Wir werden im folgenden anhand einiger Beispiele die vier Hauptkategorien skizzieren; der vollständige Kriterienkatalog ist in [Domb 97] aufgeführt:

- **Unternehmensbezogene** Kriterien umfassen Anforderungen an betriebswirtschaftliche Aspekte eines Managementsystems wie z.B. Preise und Lizenzgebühren, Support und Zuverlässigkeit des Herstellers, Möglichkeiten einer Teststellung, Schulungsaufwand usw.
- **Informationsbezogene** Kriterien fokussieren auf die Anforderungen des Informationsaustauschs sowie die Arten, auf Managementinformation zuzugreifen, also Aspekte wie die Art der Speicherung (Dateien, relational, objektorientiert), die Schnittstellen und Abfragesprachen zum externen Zugriff auf Managementinformation und die unterstützten Objektkataloge.
- **Funktionsbezogene** Kriterien beziehen sich auf den von einem System bereitgestellten Funktionsumfang, d.h. sie beurteilen aus Benutzersicht die Qualität von MIB-Browsern sowie Autodiscovery- und Autotopologyfunktionen oder die Möglichkeiten zur automatisierten Generierung von Statusberichten.
- **Systembezogene** Kriterien zielen auf die Anforderungen eines Managementsystems an die Ablaufumgebung ab wie z.B. die Möglichkeit der Kopplung von Managementsystemen mit Trouble-Ticket-Systemen oder Inventardatenbanken des Betreibers oder die Mechanismen, um im Fehlerfalle Meldungen über E-Mail, Telefon oder Pager an das Wartungspersonal abzusetzen.

Dieser insgesamt über 250 Einzelparameter umfassende Kriterienkatalog konnte bereits mehrfach erfolgreich in der Praxis eingesetzt werden[7]. Während eine solche Aufstellung für den Betreiber von Managementsystemen ein wichtiges Hilfsmittel darstellt, sind wir im Rahmen der vorliegenden Arbeit jedoch an der technischen Instrumentierung verteilter kooperativer Managementsysteme interessiert. Wir werden uns daher ausschließlich auf die in Abschnitt 2.3 identifizierten technischen Anforderungen stützen.

[7] Unter anderem von der Kantonalverwaltung in Luzern (Schweiz), die den Kriterienkatalog zur Evaluierung neu zu beschaffender Managementsysteme verwendet hat.

Diese verdeutlichen die logische Schichtung verteilter Ablaufumgebungen und Managementarchitekturen sowie der darauf aufbauenden spezifischen Managementinformation und -dienste (siehe Abbildung 2.7): Anforderungen wie z.B. Offenheit, Flexibilität und Verteilungstransparenz sind nicht nur notwendige Charakteristiken einer Managementarchitektur, sondern insbesondere Eigenschaften der Middleware, also des Rahmenwerkes für verteilte Verarbeitung, mit dem diese implementiert wird.

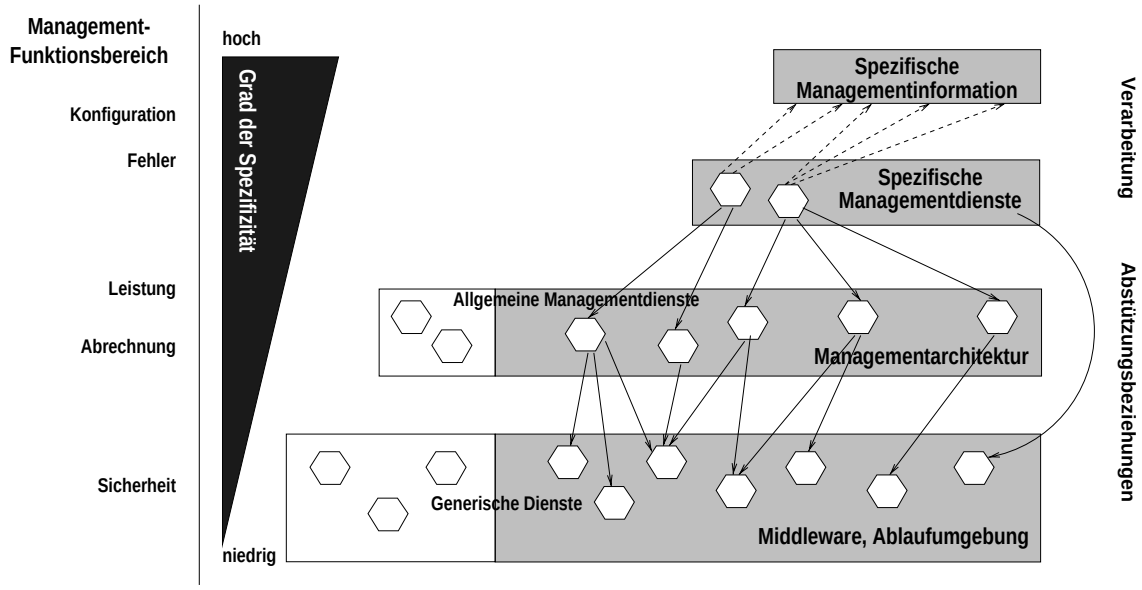


Abbildung 2.7: Wiederverwendbarkeit und Erweiterung von (Management-) Diensten

Wie in Abbildung 2.7 dargestellt, impliziert die logische Schichtung der Middleware- und Managementarchitekturen, daß eine so implementierte Managementarchitektur eine Vielzahl wichtiger Dienste bereits von der Middleware übernehmen kann und nicht mehr neu definieren muß: Beispiele hierfür sind Dienste zur Zustellung asynchroner Ereignismeldungen und zur Ermittlung von Metainformation (wie die Signatur einer Komponente oder die Zahl der Instanzen). Ist ein Ereignisdienst bereits Bestandteil der Middleware, kann ein davon abgeleiteter Management-Ereignisdienst wesentliche Teile des Funktionsumfangs erben und gegebenenfalls anpassen. Dies ist jedoch für den überwiegenden Teil der in Abschnitt 3.1 vorgestellten Managementarchitekturen nicht gegeben; somit müssen die dort notwendigen Managementdienste von Grund auf neu implementiert werden. Im Sinne der Einsparung von Entwicklungskosten ist jedoch ein hoher Grad an Wiederverwendbarkeit durch geeignete Abstützung des Managements auf vorhandene Middleware wünschenswert.

Auch in den darüberliegenden logischen Ebenen ist die Wiederverwendbarkeit von Diensten gegeben: Managementinformation und -dienste für ein spezifisches Szenario (hier: das Management von Managementsystemen) stützen sich auf bereits in den darunterliegenden Schichten vorhandene Dienste ab und verfeinern diese gegebenenfalls.

Voraussetzung ist hierfür allerdings, daß entweder die Notationen zur Definition von Information und Diensten sowie die Mechanismen zum Zugriff auf Komponenten einheitlich sind oder geeignete Mittel zur Sicherstellung der Interoperabilität vorhanden sind. Kapitel 4 stellt einige Ansätze vor, die dieses leisten.

Folglich reicht es nicht aus, unsere Betrachtungen auf reine Managementarchitekturen zu beschränken; vielmehr ist es notwendig, im weiteren Verlauf auch Rahmenwerke für verteilte Umgebungen in unsere Überlegungen mit einzubeziehen.

Die in Abschnitt 2.2 analysierten Managementszenarien sowie die daraus (in Abschnitt 2.3 abgeleiteten Anforderungen haben ebenfalls verdeutlicht, daß nicht nur ein enger Zusammenhang zwischen Managementinformation und Managementdiensten besteht, sondern der Übergang zwischen diesen beiden Ausprägungsformen von Managementinstrumentierung fließend ist. Ein Beleg hierfür sind die oben besprochenen Möglichkeiten zur Realisierung zustandsbehafteter und abgeleiteter Managementinformation: Diese können entweder in Form von Managementdiensten implementiert sein, die aus der von den Ressourcen zur Verfügung gestellten Basisinformation betreibergerechte Managementdaten generieren oder bereits Teil der Basisinformation sein. Letzteres ist, wie wir in Abschnitt 2.3.3 begründet haben, aufgrund der vielfältigen Ausprägungsformen nicht sinnvoll.

Eine weitere Implikation ist die Notwendigkeit einer *einheitlichen Notation* zur Definition sowohl von Managementinformation als auch von Managementdiensten. Sie wird uns im weiteren Verlauf dieser Arbeit gestatten, dieselben Verfahren zur Modellierung, Spezifikation und Implementierung sowohl von Managementinformation als auch von Managementdiensten anzuwenden.

Wir haben in diesem Kapitel ebenfalls festgestellt, daß **Sicherheit**sdienste aufgrund ihrer essentiellen Bedeutung bereits auf unterster Ebene eines verteilten Managementsystems vorhanden sein müssen und folglich oft bereits durch die verteilte Ablaufumgebung erbracht und durch die darauf aufbauende Managementarchitektur verfeinert werden. Die anwendungsspezifische Ausgestaltung dieser Dienste des Sicherheitsmanagements besteht daher oft nur in einer geeigneten Wertebelegung der Signatur elementarer Sicherheitsfunktionen.

Dienste, die den Management-Funktionsbereichen **Abrechnung** und **Leistung** zuzuordnen sind, sind zu großen Teilen ebenfalls generisch, wie beispielweise die Berechnungsvorschriften für gewichtete Durchschnittswerte oder die Bildung von Analysen über begrenzte Zeiträume. Auch hier können bereits in den Schichten der Ablaufumgebung sowie der Managementarchitektur einige solcher Dienste realisiert werden, die von höheren, problemspezifischeren Diensten geeignet instrumentiert werden. Die Qualität solcher Managementdienste in Bezug auf Betreiberanforderungen hängt jedoch maßgeblich von denjenigen Managementdaten ab, die durch die individuelle, problemspezifische Instrumentierung erbracht werden.

Auf einen speziellen Anwendungsfall individuell zugeschnittene Managementinformation findet man insbesondere in den Bereichen des **Konfigurations-** und **Fehlermanagements**, da hier die Spezifizität einer Anwendung am deutlichsten sichtbar wird. Sie bietet

uns somit die vielversprechendsten Möglichkeiten zur Abgrenzung dieser Arbeit von verwandten Problemklassen. Daher werden wir uns in Kapitel 5 dieser Arbeit hauptsächlich auf die Modellierung, Spezifikation und Implementierung von Managementinformation und Managementdiensten für diese beiden Management-Funktionsbereiche abstützen.

KAPITEL 3

Verteilte kooperative Managementsysteme: Status Quo

Die im vorigen Kapitel aus den Managementszenarien abgeleiteten Anforderungen an das Management verteilter kooperativer Managementsysteme beziehen sich zu einem nicht unbeträchtlichen Teil auf Rahmenwerke für verteilte Anwendungen sowie auf Managementarchitekturen. Um zu einem tragfähigen Lösungskonzept zu gelangen, ist es zunächst notwendig, die zahlreich vorhandenen Architekturen sowohl unter konzeptionellen als auch unter technischen Gesichtspunkten auf ihre prinzipielle Eignung für das Enterprise Management zu untersuchen.

Wir werden daher in Abschnitt 3.1 auf bereits existierende, standardisierte Rahmenwerke für das Management ebenso eingehen wie auf neuartige Architekturen, die möglicherweise in der Zukunft eine bedeutende Rolle spielen werden. Neben den Managementarchitekturen sind für die Thematik der vorliegenden Arbeit auch einige meist jüngere Forschungsansätze relevant, die wir in Abschnitt 3.2 vorstellen und hinsichtlich ihrer Einsetzbarkeit für die Zwecke des Enterprise Managements und damit bezüglich ihrer Eignung für die vorliegende Arbeit bewerten. Nicht zuletzt aufgrund der hohen Bedeutung integrierten Managements sind zahlreiche kommerzielle Systeme am Markt erhältlich, die im Kontext dieser Arbeit ebenfalls eine wichtige Rolle spielen, da es sich bei ihnen um potentielle Kandidaten für verteilte kooperative Managementsysteme handelt. Folglich werden wir in Abschnitt 3.3 einige charakteristische Vertreter solcher Systeme vorstellen, die einen repräsentativen Querschnitt der am Markt erhältlichen Produkte bilden. Zum Abschluß dieses Kapitels werden in Abschnitt 3.4 diejenigen Architekturen und Ansätze ausgewählt, auf denen in den folgenden Kapiteln unsere Methodik aufsetzt. Ferner wird begründet, weshalb sich die genutzten Frameworks besonders gut für die vorliegende Arbeit eignen und wie sie kombiniert werden können, um den größten Nutzen zu erbringen.

3.1 Gegenwärtige und zukünftige Managementarchitekturen

Im folgenden Abschnitt werden bereits standardisierte sowie zur Zeit in der Standardisierungsphase befindliche Architekturen vorgestellt und hinsichtlich ihrer Eignung für das Management verteilter kooperativer Managementsysteme bewertet. Die im OSI-Management standardisierte Gliederung des Managements in vier Teilmodelle (beschrieben in Abschnitt 2.1.1) erweist sich auch bei der Darstellung anderer Managementarchitekturen als zweckmäßig und ist insbesondere hinsichtlich deren Klassifizierung und Bewertung vorteilhaft. Ferner wird aufgezeigt, daß sich die Heterogenität der Architekturen nicht auf alle Teilmodelle in gleicher Weise auswirkt.

3.1.1 OSI/TMN-Management

Organisationsmodell

Das OSI-Organisationsmodell unterteilt die Rollen der am Managementprozeß beteiligten Systeme in Manager und Agenten. Ein System kann zu einem Zeitpunkt auch beide Rollen wahrnehmen, was unserer Definition eines verteilten kooperativen Managementsystems (vgl. Abschnitt 2.1.1) entspricht. Neben diesen asymmetrischen Manager/Agent-Beziehungen wird ebenfalls ein weitreichendes Domänenkonzept festgelegt, das die Gruppierung von Managementobjekten nach organisatorischen oder verwaltungsbezogenen Gesichtspunkten gestattet. Die Tatsache, daß keine technologiespezifischen Kriterien zur Domänenbildung vorgesehen sind, reflektiert sowohl die aus den Anforderungen abgeleitete („Top-down“) Sichtweise eines Netzbetreibers als auch die Prämisse, daß man sich in bezug auf die zugrundeliegende Managementarchitektur in einem homogenen Umfeld (hier: OSI) befindet. Die Tatsache, daß in heutigen Kommunikationsnetzen diese Annahme nur in Ausnahmefällen gilt, wurde unter anderem in dem in Abschnitt 2.2.4 aufgeführten Managementszenario dargelegt.

Informationsmodell

Die OSI-Managementarchitektur definiert für das Informationsmodell einen objektorientierten Ansatz [ISO 10165-1]. **Managementobjekte** (*Managed Objects, MOs*) beschreiben reale Ressourcen und sind Instanzen von **Management-Objektklassen** *Managed Object Classes (MOCs),* welche die Eigenschaften einer bestimmten Klasse von Ressourcen in einer standardisierten Notation, den sogenannten *Guidelines for the Definition of Managed Objects (GDMO)* [ISO 10165-4] definieren. Eine MOC wird in der Regel als Unterklasse von einer oder mehreren Vaterklasse(n) definiert und erbt dadurch alle Eigenschaften dieser Klasse(n). Der Bildung einer **Vererbungshierarchie** (*Inheritance Hierarchy*) liegt das Vorhandensein von Mehrfachvererbung zugrunde. Erstere ist insbesondere

wichtig, um das Vorhandensein einer garantierten minimalen Grundmenge an Managementinformation für alle MOCs zu gewährleisten: Neue MOCs besitzen durch die Vererbungshierarchie bereits allgemeine Attribute und enthalten daher lediglich für diese MOC spezifische Informationen. Mehrere für das Management einer Ressourcenklasse erforderliche MOCs werden in der **Management Information Base (MIB)** zusammengefaßt.

Die kleinste wiederverwendbare Einheit im OSI-Management ist nicht etwa eine einzelne Managementobjektklasse, sondern zu einem sog. *Package* zusammengefaßte Bestandteile eines Objekts. Hierunter versteht man eine Ansammlung von Attributen, Aktionen, asynchronen Ereignismeldungen und Semantik (sog. *Behaviour*), also eine sehr feingranulare Form von Managementinformation. Die Definition asynchroner Ereignismeldungen ist ebenfalls expliziter Bestandteil einer Objektklassenspezifikation. Objektklassen bestehen ihrerseits mindestens aus einem oder mehreren obligatorischen (*mandatory*) Packages und können weitere optionale (*conditional*) Packages enthalten. Dies bedeutet, daß bei Instanzen von Objektklassen sämtliche in den obligatorischen Packages spezifizierten Parameter vorhanden sein müssen, während die in den zusätzlichen optionalen Packages vorhandene Information zwischen einzelnen Instanzen derselben MOC variiert. Letzteres hängt von den in diesen Packages spezifierten Bedingungen ab. Hierdurch wird sogenanntes *late binding* realisiert, d.h. die von einem Managementobjekt bereitgestellte Informationsmenge wird erst zum Zeitpunkt der Instantiierung (und nicht schon bei der Übersetzungszeit) festgelegt und variiert somit zwischen Objekten, denen dieselbe Managementobjektklasse zugrundeliegt. Diese Detaileigenschaft sorgt einerseits für eine ausgesprochen hohe Flexibilität, trägt jedoch andererseits zur Komplexität des OSI-Managements bei (vgl. hierzu [Rama 98]).

Von den unterschiedlichen Beziehungen zwischen Objektklassen ist die Enthaltenseinsbeziehung die wichtigste, da sie die Basis für die eindeutige Namensgebung ist und zum Aufbau des **Management Information Tree (MIT)** benötigt wird, der in der Literatur oft auch als **Enthaltenseinshierarchie** (*Containment Hierarchy*) bezeichnet wird. Um MOCs instantiieren zu können, spielt ihre Anordnung bezüglich der Enthaltenseinseigenschaften eine wichtige Rolle: Sie bildet die Grundlage für ein Managementsystem, um den logischen Aufbau einer Ressource, d.h. ihre Darstellung in bezug zu anderen Komponenten an der Operatorkonsole zu bestimmen. Die hierzu erforderlichen Informationen (also die Einschränkungen bezüglich der Enthaltenseinsbeziehung von Objektklassen) sind durch sogenannte **Name Bindings** festgelegt, die ebenfalls in GDMO spezifiziert, jedoch nicht Bestandteil der Objektklassen selbst sind. Innerhalb der Objektklassenspezifikation muß jeweils ein sog. **Naming Attribute** vorhanden sein, dessen Bezeichnung zusammen mit seiner Wertebelegung den **Relative Distinguished Name (RDN)** bildet. Die Konkatenation aller Tupel von Naming Attribut und Wertebelegung ausgehend von der Wurzel des MIT wird als **Distinguished Name (DN)** bezeichnet und dient der eindeutigen Referenzierung von Objektinstanzen, die im sog. **Naming Tree** registriert sind.

Kommunikationsmodell

Das Kommunikationsmodell für das schichtübergreifende Management der OSI-Architektur beruht auf dem CMIS-Dienst [ISO 9595], der sich hierzu des CMIP-Managementprotokolls [ISO 9596-1] bedient. Als protokollbasierte Architektur bietet das OSI-Management optimierte Zugriffsmechanismen auf Managementinformationen: Der Zugriff ist hierbei nicht nur auf einzelne Managementobjekte beschränkt, sondern gestattet die Auswahl baumartig angeordneter Objektgruppen (sog. **Scoping**), die auf das Vorhandensein von Attributwertebelegungen geprüft werden können (**Filtering**). Für die Wahl des Scopes einer Operation gibt es vier verschiedenen Arten: das Basisobjekt selbst, das Basisobjekt sowie sämtliche Objekte bis zur n-ten Ebene unterhalb des Basisobjekts, die Objekte innerhalb der n-ten Ebene unterhalb eines gegebenen Basisobjekts oder der gesamte Teilbaum unterhalb eines Basisobjekts. Scoping und Filtering sind sehr leistungsfähige Merkmale und erweitern OSI-Managementagenten um Fähigkeiten, wie sie aus objektorientierten Datenbanken bekannt sind. Die Auswertung der vom Managementsystem aufgestellten Filterkriterien geschieht vollständig auf Seiten des Agentensystems, was durch die implizite Verteilung einerseits Managementsysteme entlastet und andererseits zu einer signifikanten Verminderung des durch das Management ausgelösten Netzverkehrs führt. Die Ergebnisse von Operationen, die auf Objektgruppen ausgefürt werden, führen pro Objektinstanz zu einer Antwort, d.h. auf eine Anfrage können mehrere zusammenhängende Antworten (*Linked Replies*) eingehen.

Einem OSI-Managementsystem stehen für die Kommunikation mit einem OSI-Agenten folgende Protokolldateneinheiten *Protocol Data Units (PDU)* zur Verfügung: m-Get, m-Set, m-Create, m-Delete, m-Action und m-CancelGet. Eine Instanz einer MOC eines OSI-Agenten kann Ereignisse mit Hilfe von m-EventReport PDUs an das OSI-Managementsystem schicken. Die Kommunikation kann dabei sowohl bestätigt als auch unbestätigt erfolgen. Einige dieser PDUs erlauben zusätzlich das Übertragen von Scopes und Filtern (m-Get, m-Set, m-Action, m-Delete). Außerdem kann in diesen PDUs eine Synchronisationsbedingung übermittelt werden (z.B. *atomic, best effort*). CMIP ist ein sog. *remote execution* Protokoll, das auf asynchronem Message Passing beruht und nicht auf (synchronen) entfernten Prozeduraufrufen, wie sie bei anderen Architekturen (s.u.) verwendet werden.

Funktionsmodell

Mit gegenwärtig über 22 **Systems Management Functions (SMFs)** [ISO 10164-x] verfügt das OSI-Management über ein ausgesprochen breites Spektrum an Basisdiensten, die die effiziente Nutzung der Kommunikationsinfrastruktur gewährleisten und generische, häufig benötigte Managementfunktionalität bereitstellen. Wir werden uns an dieser Stelle auf charakteristische Managementdienste beschränken, an denen das Designziel des OSI-Managements einer hohen Skalierbarkeit durch die Nutzung von verteilter Managementfunktionalität besonders deutlich wird.

Mit der **Event Report Management Function** [ISO 10164-5] bietet OSI eine feingranulare dezentrale Ereignisverwaltung, die auf **Event Forwarding Discriminators (EFD)** basiert. EFDs sind spezielle Filterobjekte, die von einem Managementsystem innerhalb von Agenten kreiert und geeignet konfiguriert werden können. In ihnen sind Informationen über diejenigen Managementsysteme gespeichert, an die Ereignisse gesandt werden sollen sowie Filterangaben, die sich auf Ereignistypen, Objektklassen und -instanzen, Zeitstempel usw. beziehen. Ereignismeldungen können durch Verwendung der **Log Control Function** [ISO 10164-6] zusätzlich persistent bei den Agenten in **Log Records** gespeichert werden, die vorab vom Manager eingerichtet wurden. Dies bedeutet, daß ein Manager ebenfalls auf Agentenseite Filter konfigurieren kann, die darüber entscheiden, ob eine Ereignismeldung gespeichert werden soll. Beim Auftreten eines Ereignisses laufen die Filter- und Speichervorgänge selbständig auf den Agentensystemen ab.

Ein charakteristisches Hilfsmittel zur Bereitstellung der in Abschnitt 2.3.3 erwähnten abgeleiteten Managementinformation sind die **Metric Objects and Attributes** [ISO 10164-11] sowie die **Summarization Function** [ISO 10164-13], deren Hauptaufgabe in der Definition von Diensten zur Erkennung potentieller Überlastsituationen als Mittel zur präventiven Fehlerbehebung besteht. Neben Mechanismen zur Ermittlung der Betriebsmittelauslastung (gegenwärtige Auslastung, gestellte Nutzungsanfragen, abgelehnte Nutzungsanfragen aufgrund von Überlast) werden auch diverse Zähler- (*Counter*) und Pegelobjekte (*Gauges*) definiert. Für diese Basisobjekte können dann sogenannte Monitor-Objekte gebildet werden, die aus den Rohdaten der Basisobjekte statistische Informationen (wie z.B. die Auslastung eines Betriebsmittels über frei definierbare Zeitintervalle, gewichtete und ungewichtete Durchschnittswerte beliebiger Zählerobjekte, Bildung von Standardabweichungen, Varianzen usw.) ableiten. Auch in diesem Falle werden die Objekte zur Bildung abgeleiteter und somit zustandsbehafteter Managementinformation vom Managementsystem konfiguriert und anschließend dezentral, d.h. auf den Agentensystemen ausgeführt. Naheliegenderweise werden mit diesen abgeleiteten Managementinformationen ebenfalls Ereignisfilter definiert (s.o.), die die Benachrichtigung eines Managementsystems beim Überschreiten von Maximal- bzw. Minimalwerten veranlassen.

Der dynamischen Ermittlung von Agenteneigenschaften dient die **Management Knowledge Management Function (MKMF)** [ISO 10164-16]. Diese umfassen Wissen über Managementobjektklassen (welche MOCs verwaltet ein Agentensystem, welche MOCs können instantiiert und überwacht werden) und -instanzen (welche Agentensysteme besitzen wieviele Instanzen eines bestimmten Typs, welche Conditional Packages sind aktiviert) sowie deren Beziehungen zueinander. Das Vorhandensein von Mechanismen zur Abfrage von Metainformation zur Laufzeit ist zur Bestimmung der Dienste, die von einem Agentensystem angeboten werden, unentbehrlich und bildet die Grundlage, um neue Agentensysteme dynamisch in einen Managementkontext einzubinden (sog. *Discovery*-Funktionalität).

Neben diesen angesprochenen Diensten bietet das OSI-Management eine Fülle wichtiger und nützlicher Managementdienste, die flexibel konfigurierbar sind und einen großen Teil

des Anforderungsspektrums abdecken. Hierzu zählen insbesondere Dienste zur Administration von Managementobjekten, deren Zuständen und Beziehungen zu anderen Objekten, aber auch Sicherheits- und Abrechnungsdienste.

Telecommunications Management Network

Die umfassenden Möglichkeiten des OSI-Managements zur effizienten Realisierung verteilten Managements haben dazu geführt, daß die Betreiber öffentlicher Kommunikationsnetze im Rahmen der Standardisierungsaktivitäten eines **Telecommunications Management Network (TMN)** auf der OSI-Managementarchitektur aufgebaut und diese um Belange des Telekommunikationsmanagements erweitert haben. Diese Erweiterungen umfassen insbesondere ein architekturelles Rahmenwerk [ITU M.3010], das ebenfalls die Integration von nicht OSI-konformen Managementsystemen vorsieht. Bei TMN handelt es sich um ein eigenständiges Managementnetz, das physisch von den Carriernetzen getrennt ist, auf denen ausschließlich Nutzdaten transportiert werden. Um der bereits in Abschnitt 2.2.3 angesprochenen Problematik des Zusammenwirkens unterschiedlicher (Telekommunikations-)Dienstleister gerecht zu werden, werden in TMN insgesamt fünf Schichten definiert, auf die jedoch an dieser Stelle aus Platzgründen nicht näher eigegangen werden soll. Detaillierte Informationen hierzu findet man in [Cohe 94] und [Sahi 94].

Das TMN-Referenzmodell verfeinert mit den **Function Blocks** das OSI-Organisationsmodell dahingehend, daß anstelle der Manager- und Agentenrollen detailliertere Ausprägungsformen der am Managementprozeß teilnehmenden Systeme definiert werden: **Operations Systems Functions (OSF)** sind diejenigen Managementsysteme, in denen die Verarbeitung von Managementinformation geschieht, um die TMN-Komponenten zu administrieren. Die Benutzerschnittstelle zum TMN wird durch **Work Station Functions (WSF)** bereitgestellt. Eine **Network Element Function (NEF)** entspricht im wesentlichen dem OSI-Agentenbegriff und gestattet das Management der Netzressourcen, um die von NEFs bereitgestellten Informationen an die Erfordernisse der OSFs anzupassen: Hierunter fallen Aufgaben wie z.B. das Sammeln, Anpassen, Filtern und Verdichten von Managementinformation. Die Einbeziehung von nicht TMN-konformen Ressourcen in ein globales (TMN-basiertes) Management bedingt das Vorhandensein von **Q-Adapter Functions (QAF)**, die entsprechende Konvertierungsfunktionen bereitstellen. Zwischen diesen Funktionsblöcken definiert TMN Referenzpunkte, die in Abbildung 3.1 eingezeichnet sind. Für die vorliegende Arbeit sind insbesondere folgende Referenzpunkte von Bedeutung: OSF-zu-OSF Kommunikation wird über das am q3-Referenzpunkt definierte CMIP-Interface (Q3) abgewickelt, der x-Referenzpunkt verbindet unterschiedliche TMNs miteinander und der m-Referenzpunkt stellt den Abschluß des TMN gegenüber nicht-TMN konformen Systemen dar.

Hinsichtlich des Informationsmodells bietet TMN mit dem *Generic Network Information Model* [ITU M.3100] eine Erweiterung des OSI-Informationsmodells um einen Objektkatalog, der speziell auf die Belange des Telekommunikationsmanagements abstellt. Sämtli-

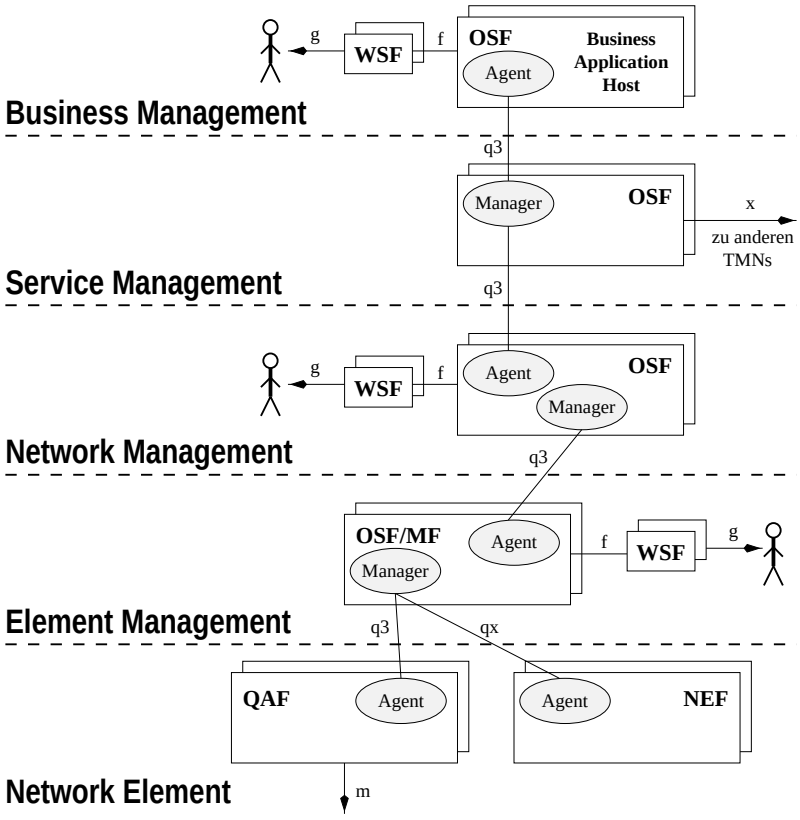


Abbildung 3.1: Bestandteile des Telecommunications Management Network

che Konzepte und Prinzipien des OSI-Informationsmodells werden unverändert übernommen [Sido 98].

Dies gilt ebenso für die verbleibenden Teilmodelle des OSI-Managements: Wie bereits oben angedeutet, ist CMIS derjenige Dienst, mit dem Informationen zwischen den TMN-Bestandteilen ausgetauscht werden. Ferner werden die OSI Systems Management Functions in TMN angewendet.

Bewertung

Es ist deutlich erkennbar, daß das OSI-Management sehr leistungsfähige Mechanismen bietet, die vollständig ereignisgesteuertes Management bei einem hohen Verteilungsgrad ermöglichen. Nicht zuletzt deswegen hat es sich in Umgebungen, die eine sehr große Anzahl von Ressourcen umfassen, durchgesetzt. Dies wird durch die Verwendung des OSI-Managements in TMN unterstrichen, da man es gerade im Bereich der Telekommunikation mit sehr großen und komplexen Systemen zu tun hat, die häufig von unterschiedlichen Betreibern überwacht und gesteuert werden müssen. Die Notwendigkeit der Kommunikation zwischen Managementsystemen (evtl. unterschiedlicher Betreiberorganisationen) spiegelt sich daher explizit im Vorhandensein des x-Referenzpunkts wider. Ebenfalls sind

mit den Q-Adapter Functions Übergänge von TMN in andere Architekturwelten vorgesehen. TMN kann somit als eine zeitgemäße Fortführung des OSI-Managements aufgefaßt werden und wird durch einige Hersteller von Managementsystemen unterstützt. Der hohen Zahl an standardisierten Managementdiensten steht in der Praxis jedoch lediglich eine verhältnismäßig geringe Menge an tatsächlich implementierter Managementfunktionalität entgegen: Gegenwärtige OSI-basierte Managementsysteme wie die in Kapitel 4 besprochene *IBM NetView TMN Support Facility* oder *OpenView Telecom* von Hewlett-Packard implementieren meist nur die unbedingt notwendigen (oft weniger als ein halbes Dutzend, d.h. knapp ein Viertel der tatsächlich vorhandenen) Managementdienste. Nichtsdestoweniger dient die OSI-Managementarchitektur häufig als Vergleichsmaßstab für andere Managementarchitekturen.

Die hohe Flexibilität und die zahlreichen Konfigurationsmöglichkeiten der generischen Managementdienste, die sich bereits aus der obigen Beschreibung weniger Managementdienste deutlich abzeichnet, bedingen im praktischen Einsatz nicht nur eine tiefgreifende Analyse, welche Managementdaten man erhalten möchte, sondern erfordern insbesondere eine aufwendige Anpassung (*Customizing*) an die spezifischen Gegebenheiten des Betreibers. Dieser Aufwand erscheint für das Management kleinerer Kommunikationssysteme in Unternehmen oft zu hoch, weshalb sich das OSI-Management im Bereich unternehmensweiter Datennetze nicht durchgesetzt hat.

Ein weiteres Problem für die Entwickler von OSI/TMN-Systemen war bis vor kurzem, daß die in OSI gebräuchliche GDMO-Notation lediglich auf Konstrukte der Programmiersprache C abgebildet werden konnte: Diese Abbildungsvorschriften waren Bestandteil der *X/Open OSI-Abstract-Data Manipulation API (XOM)* Spezifikation [XOM 91]. Auch für das Managementprotokoll CMIP stand mit dem *X/Open Management Protocol (XMP)* [XMP 92] ausschließlich eine C-Programmierschnittstelle zur Verfügung, die aufgrund ihrer Komplexität einen zweifelhaften Ruf hatte. Dies hat sich in jüngster Zeit mit dem Vorhandensein der TMN/C++ API [CCSH 97] geändert, die eine komfortable Entwicklungsumgebung für C++-basierte Managementapplikationen bereitstellt und im Rahmen der *OMNIPoint*-Initiative des Network Management Forums ([NMF 93], [Murr 93]) entwickelt wurde. Wir werden darauf in Abschnitt 4.4 detailliert eingehen.

3.1.2 Object Management Architecture

Ein wesentliches Merkmal der im vorigen Abschnitt vorgestellten OSI/TMN-Managementarchitektur besteht in der Verwendung von „Outband Management“, d.h. der Nutzung getrennter Architekturen, Protokolle und Kommunikationswege (bezogen auf die höheren Schichten eines verteilten Systems) für Nutz- und Managementdaten. Ein aus den Unterschieden dieser Systeme resultierendes Problem besteht darin, daß ein Entwickler einer verteilten Anwendung in der Regel kaum über Kenntnisse bezüglich existierender Managementarchitekturen verfügt und somit wohl nur in den seltensten Fällen eine zu diesen Architekturen konforme Managementschnittstelle für die verteilte Anwendung vom Her-

steller dieser Anwendung bereitgestellt wird.

Demgegenüber verläuft CORBA[1]-basiertes Management[2] nach dem Prinzip des „Inband Management“, d.h. Nutz- und Managementdaten haben nicht nur dieselbe Kommunikationsarchitektur, sondern können auch auf einheitliche Weise modelliert und implementiert werden. Somit kann der Entwickler einer verteilten Anwendung dieselben Methoden und Werkzeuge (wie zum Beispiel CASE-Tools, Codegeneratoren, Debugger) zur Implementierung der Managementinstrumentierung benutzen, die er bereits für die eigentliche Nutzfunktionalität verwendet. Eine derart entwickelte verteilte Anwendung verfügt somit über zwei in einer einheitlichen Syntax spezifizierte Schnittstellen: Die eigentliche Nutzfunktionalität steht über eine Dienstschnittstelle (*Service Interface*) anderen Modulen zur Verfügung; die zu ihrem Management erforderlichen Dienste sind für die Managementapplikation über eine *Managementschnittstelle* zugreifbar. Management wird somit zu einem integralen Bestandteil einer verteilten Anwendung.

Die folgenden Abschnitte beschreiben anhand der vier in Abbildung 3.2 dargestellten Management-Teilmodelle die Möglichkeiten, die CORBA für integriertes Management bietet. Im weiteren Verlauf werden insbesondere die Abschnitte 4.4.2 bis 4.4.5 unter Rückgriff auf die nun folgenden Ausführungen aufzeigen, wie die einzelnen Teilmodelle unterschiedlicher Managementarchitekturen aufeinander abgebildet werden können.

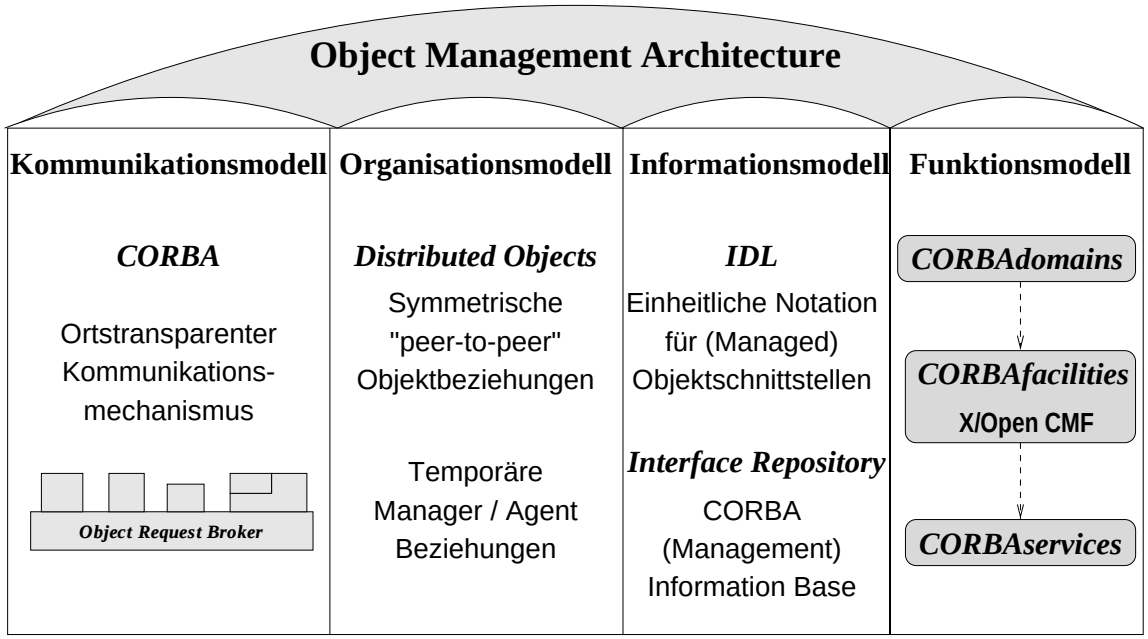


Abbildung 3.2: OMA als Managementarchitektur

[1]Eine ausführliche Beschreibung von CORBA befindet sich in Anhang A.

[2]Wie es die Überschrift dieses Abschnitts andeutet, handelt es sich hierbei – strenggenommen – um **OMA**-basiertes Management. Nichtsdestoweniger hat sich in der Fachwelt der Begriff „CORBA-basiertes Management“ etabliert, weshalb auch wir diesen im Zuge dieser Arbeit verwenden.

Organisationsmodell

Im Gegensatz zu klassischen Managementarchitekturen setzt CORBA keinerlei Hierarchie zwischen Objekten voraus, sondern geht von symmetrischen Peer-to-Peer (Client/Server) Beziehungen zwischen Objekten aus; das aus dem OSI-Management bekannte Organisationsmodell kann dahingehend erweitert werden, daß nicht nur Manager/Agent- und Manager/Manager-Beziehungen möglich sind, sondern auch Agenten/Agenten-Beziehungen. Dies erlaubt nicht nur die Verteilung von Managementfunktionalität an hierarchisch gleichwertige Managementsysteme sowie die Delegierung von Managementdiensten an hierarchisch niedrigere Systeme, sondern gestattet prinzipiell auch die Kooperation zwischen Agentensystemen. Im Hinblick auf die in Abschnitt 2.1 festgelegten Randbedingungen werden jedoch derartige Kooperationsbeziehungen autonomer Agentensysteme nicht in dieser Arbeit behandelt, da die Definition von Maßnahmen zur Sicherstellung eines geordneten Betriebs dieser Systeme sehr umfangreich ist und den Rahmen dieser Arbeit bei weitem sprengen würde: Verfahren, die gewährleisten, daß das durch die Tätigkeit der Agenten hergestellte lokale Optimum auch tatsächlich dem globalen Optimum entspricht, das für das Gesamtsystem allein relevant ist, sind überdies bisher noch nicht bekannt geworden.

Informationsmodell

CORBA bietet neben den Vorteilen objektorientierter Programmiersysteme (z.B. Kapselung, Datenabstraktion, Vererbung, Polymorphie) per Definition die Möglichkeit, verteilte (Management-) Anwendungen so zu implementieren, daß deren einzelne Module interagieren können, ohne die Spezifika der darunterliegenden Kommunikations- bzw. Systemarchitekturen beachten zu müssen, da die Modulschnittstellen in einer einheitlichen Schnittstellen-Beschreibungssprache (*Interface Definition Language, IDL*) offengelegt sind. Die in Abschnitt 2.3.1 angesprochenen acht Ausprägungsformen von Verteilungstransparenz werden durch CORBA nahezu vollständig erfüllt: Lediglich die Bedingung der Migrationstransparenz umfaßt Festlegungen, die über den Umfang von CORBA hinausgehen (z.B. an die Programmiersprache und evtl. vorhandene virtuelle Maschinen) und kann folglich nur dann von CORBA erfüllt werden, wenn diese zusätzlich bereitgestellt werden. Dies liegt jedoch außerhalb des Umfangs der Festlegungen zu CORBA. Wir werden diesen Punkt in Kapitel 6 eingehender behandeln. An dieser Stelle sei daher lediglich darauf hingewiesen, daß die Beschreibung der Modulschnittstellen in IDL nicht die *Portabilität* der Module impliziert, da diese in der Regel die besonderen Eigenschaften einer spezifischen Systemplattform nutzen und daher nicht ohne weiteres auf ein anderes Zielsystem migriert werden können.

Die kleinste Einheit der Verteilung bei CORBA ist ein einzelnes Objekt; durch das Fehlen des aus dem OSI-Management bekannten *Package*-Konzeptes entfällt die Möglichkeit, die Ausgestaltung eines Objekts (und damit sein Verhalten) bei seiner Instantiierung von der Erfüllung frei definierbarer Bedingungen abhängig zu machen. Eine Konsequenz da-

von ist, daß sämtliche Instanzen einer Objektklasse identisches Verhalten aufweisen und kein *late binding* möglich ist. Instanzen derselben Objektklasse werden anhand ihrer abweichenden Objektreferenzen unterschieden; umgekehrt ist es durchaus möglich, daß eine Instanz mehrere Objektreferenzen besitzt. Letzteres verdeutlicht, daß Objektreferenzen keine interne Struktur besitzen und daher nichts über die Anordnung des Objekts in Bezug zu anderen Objekten aussagen (Enthaltensein, Registrierung usw.).

Das CORBA-Analogon zu Managementoperationen besteht im Aufruf einer Methode eines anderen Objekts. Ferner werden Objekte über ihre Schnittstelle und nicht ihren Namen angesprochen. Diese Schnittstellen werden, wie bereits oben erwähnt wurde, in einer programmiersprachenunabhängigen Notation, der sogenannten *OMG Interface Definition Language, (OMG IDL)*[3] beschrieben. Hierdurch wird sichergestellt, daß sämtliche Objekte eines CORBA-Systems auf einheitliche Art angesprochen werden können, ohne beachten zu müssen, auf welcher Systemarchitektur und unter welchem Betriebssystem diese laufen. Die Frage, in welcher Programmiersprache Managementobjekte implementiert sind, ist für die Kommunikation zwischen diesen ebenfalls irrelevant.

Hinsichtlich der Definition der von einem Objekt ausgesandten asynchronen Ereignismeldungen ergeben sich im Vergleich zu OSI weitere Unterschiede: In CORBA sind von einem Objekt ausgehende asynchrone Ereignismeldungen nicht Bestandteil der Objektdefinition, da hier Ereignisse wie „Operationen in umgekehrter Richtung“ ablaufen, d.h. als Methodenaufruf am Empfängerobjekt sichtbar sind. Asynchrone Ereignismeldungen werden somit an der Schnittstelle des empfangenden Objekts durch eine dort definierte Methode entgegengenommen.

Kommunikationsmodell

Im Gegensatz zu anderen Managementarchitekturen erfordert CORBA kein dediziertes Managementprotokoll, da hier Objekte durch das wechselseitige Aufrufen von Methoden interagieren. Ein Managementagent besteht in CORBA grundsätzlich aus jeweils individuellen Objekten, die per se weder über eine Enthaltenseinshierarchie verfügen, noch über hierarchische Namen. Kommunikationsinfrastruktur ist der Object Request Broker, der Client-Aufrufe an Server-Objekte ortstransparent weiterleitet. Somit spielt es aus Client-Sicht keine Rolle, ob das Server-Objekt lokal erreichbar ist oder sich auf einem entfernten System befindet. Voraussetzung ist hierbei natürlich die Konformität der ORBs zum CORBA-Standard [CORBA 2.2], der unter anderem Mechanismen zur Interoperabilität unterschiedlicher ORBs spezifiziert. Der Kommunikationsmechanismus des ORB entspricht einem verbindungslosen Remote Procedure Call, der in der Regel auf einem TCP/IP-Protokollstack aufsetzt und jeweils eine Anfrage (bzw. die Rückantwort) an ein spezifisches Objekt weiterleitet. Folglich ist es bisher nicht möglich, eine Operation auf mehrere Objekte simultan anzuwenden; ein CORBA-Analogon zu den effizienten Scoping- und Filteringmechanismen, wie sie vom OSI-Management bekannt sind, exi-

[3] Wenn wir im weiteren Verlauf abgekürzt von IDL sprechen, ist stets OMG IDL gemeint.

stiert nicht. Dies würde auch nicht zuletzt am Fehlen hierarchischer Namen (siehe oben) scheitern.

Für das Management bedeutet das Vorhandensein des **Dynamic Invocation Interface (DII)**, daß das Interface Repository (die Gesamtheit der MIBs aller Objekte des ORB-Systems) jeweils die aktuellen Schnittstellen der Managed Objects beinhaltet und zur Laufzeit nach bestimmten syntaktischen (nicht nach semantischen) Kriterien durchsucht werden kann, um anschließend Abfragen auf den so ermittelten MOs auszuführen. Es ist daher nicht erforderlich, daß einem Managementsystem explizit eine neue MIB bekanntgegeben werden muß; dies geschieht implizit durch das Vorhandensein der neuen Schnittstellen im Interface Repository. Eine neue Version eines Agenten kann beispielsweise ohne Unterbrechung des Gesamtsystems und ohne Neuübersetzung der entsprechenden Anwendung in das Laufzeitsystem eingefügt werden. Wie wir in Abschnitt 3.1.3 aufzeigen werden, ist beispielsweise bei der Internet-Managementarchitektur eine solche Vorgehensweise oft nur durch manuelle Konfiguration durch den Administrator möglich und erfordert im OSI-Management einen durch die *Management Knowledge Management Function* realisierten Abfragemechanismus. Die Verwendung des DII durch eine Managementapplikation bietet ferner den Vorteil, daß diese nicht modifiziert werden muß, falls sich die Server-Objekte ändern.

Um akzeptable Antwortzeiten des Gesamtsystems zu garantieren, ist es notwendig, daß für einen Client die Möglichkeit besteht, **asynchrone** Aufrufe an ein Server-Objekt zu richten. Da asynchrone, nicht blockierende Aufrufe derzeit nicht Gegenstand der CORBA-Spezifikation sind[4], ist die Nutzung von sogenannten **deferred synchronous** Aufrufen vorgesehen; synchrone Aufrufe sind ebenfalls möglich. Erstere Aufrufart ist für Managementbelange eindeutig die geeignetere Methode, da ein aufrufendes Managementsystem eine sogenannte *Handle* für das Zielobjekt enthält, um zu einem späteren Zeitpunkt den Status der Operation zu erfragen bzw. das Ergebnis anzufordern. Während der Zeitspanne der Bearbeitung durch den Server kann der Client weitere Aufrufe absetzen.

Treten beim Zugriff auf Attribute eines Objekts Fehler auf, bietet CORBA (im Gegensatz zu OSI) hierfür nur Standardfehlermeldungen; benutzerdefinierte, attributbezogene Fehlermeldungen sind – im Gegensatz zum OSI-Management – nicht vorgesehen.

Funktionsmodell

Aufgrund unserer bisherigen Ausführungen zu CORBA kann der Eindruck entstehen, daß CORBA (insbesondere im direkten Vergleich mit dem OSI/TMN-Management) einige für das Management wichtige Eigenschaften fehlen. Hierbei muß jedoch beachtet werden, daß bisher lediglich die Basisinfrastruktur von CORBA im Mittelpunkt unserer Betrachtungen stand, und für eine umfassende Bewertung ein Blick auf die im Rahmen der *Object Management Architecture (OMA)* bereitgestellten (Management-) Dienste erforderlich ist. Ferner besitzt das OSI-Management einen Entwicklungsvorsprung von ungefähr zehn Jah-

[4] Der *Messaging Service* [OSMsg 96] wird CORBA um diese Funktionalität erweitern.

ren, sodaß sein Umfang an Managementdiensten gegenwärtig objektiv größer als der von CORBA ist. Wir werden im folgenden aufzeigen, daß sowohl die Differenz in bezug auf vorhandene Dienste nicht (mehr) besonders groß ist, als auch demonstrieren, inwiefern einige der oben angesprochenen Nachteile von CORBA durch den Einsatz entsprechender Dienste aufgewogen werden. Hierbei ist zu beachten, daß Dienste einer Kategorie[5] jeweils Eigenschaften von anderen Diensten derselben oder einer darunterliegenden Kategorie erben können. Somit ist es beispielsweise möglich, aus bestehenden CORBAservices, CORBAfacilities und CORBAdomains mit vertretbarem Aufwand neue Managementfunktionalität zu realisieren. Die eigentlichen verteilten Management-Anwendungen stützen sich auf die in den drei oben genannten Kategorien definierten Dienste ab und erben somit bereits wichtige Teile ihres Dienstumfangs. Wir werden zunächst einige allgemeine CORBAservices, die für alle Klassen verteilter Anwendungen gelten, vorstellen und in Beziehung zu den entsprechenden Systems Management Functions setzen. Anschließend gehen wir auf spezifische CORBA-Managementdienste ein.

Wie bereits oben erwähnt wurde, bietet CORBA (u.a. mit dem **Lifecycle** Service) die Möglichkeit, zur Laufzeit neue Objekte in das System zu integrieren bzw. nicht mehr benötigte Objekte zu löschen. Hinsichtlich der Instantiierung von Objekten gilt jedoch die in objektorientierten Systemen verbreitete Annahme, daß sowohl die Instantiierung als auch die Löschung von Objekten nur bei Instantiierung bzw. Terminierung des Gesamtsystems geschehen, also verhältnismäßig selten. Angesichts der Tatsache, daß im Management zahlreiche kurzlebige Managementobjekte existieren (z.B. Prozesse im Anwendungsmanagement, Teilnehmerverbindungen im Telekommunikationsmanagement), ist dies für das Management nicht sinnvoll. Ein daraus resultierendes Problem besteht darin, daß in CORBA bisher kein allgemeiner Mechanismus zur Instantiierung von Objekten existiert und pro Objektklasse jeweils eine sogenannte *Factory* benötigt wird[6]. Für das Löschen, Kopieren und Migrieren von Objekten existieren hingegen generische Dienste.

Der oben angesprochenen Problematik, daß CORBA-konforme Objekte über Objektreferenzen adressiert werden und a priori keine (hierarchischen) Namen besitzen, wird durch den **Naming Service** begegnet. Somit können Objekte nicht nur einen Namen zugewiesen bekommen, sondern auch durch die Definition von Namenskontexten auch hierarchisch angeordnet werden. Allerdings kann der Objektname nicht nur während der Laufzeit des Objekts geändert werden, sondern es können zu einem Objekt auch mehrere Namen existieren. Somit entfällt einerseits die Notwendigkeit einer Enthaltenseinshierarchie, andererseits ist gegenwärtig noch nicht abschließend geklärt, inwiefern die flexible Namenszuteilung Nachteile für das Management bringt.

Durch die Verteilung und/oder Replikation wichtiger Managementfunktionalität ist es prinzipiell möglich, CORBA-basierte Managementsysteme skalierbar zu machen, d.h. so zu implementieren, daß die Verarbeitungs- und Antwortzeiten auch für eine sehr große

[5] Eine Übersicht der Dienstekategorien in CORBA befindet sich in Anhang A.

[6] Der gegenwärtig von der OMG angedachte *Meta-Object Service* soll u.a. dies leisten.

Zahl von Systemen in akzeptablen Grenzen bleiben[7]. Mangelnde Skalierbarkeit aufgrund einer Polling-Strategie ist nicht zuletzt einer der Hauptkritikpunkte an SNMP-basierten (siehe Abschnitt 3.1.3) Managementsystemen. Durch den standardisierten **Event Service** zur Zustellung asynchroner Ereignismeldungen bietet CORBA demgegenüber gute Möglichkeiten, ereignisgesteuerte Managementstrategien zu realisieren. Hierbei werden Sender und Empfänger durch *Event Channels* voneinander entkoppelt. Wir werden anhand eines von uns entwickelten Prototypen in Abschnitt 4.2 detailliert auf diese Mechanismen eingehen. An dieser Stelle sei jedoch bemerkt, daß die Filtermöglichkeiten gegenwärtig noch nicht so weit entwickelt sind wie bei der vergleichbaren OSI *Event Report Management Function*.

Der OMG **Security Service** bietet leistungsfähige Mechanismen zur objektbezogenen Authentifizierung, Zugriffskontrolle und Protokollierung, der den OSI-Sicherheitsdiensten gleichwertig ist. Die Modellierung von Beziehungen zwischen einzelnen Objekten kann mit dem **Relationship Service** so effektiv geschehen, wie es im OSI-Management mit der *Relationship Management Function* getan werden kann. Die Dienste **Property** und **Query** in Verbindung mit dem Interface Repository sowie dem DII bieten ausgezeichnete Möglichkeiten zur Bereitstellung und Abfrage von Metainformationen, die über den Umfang der in Abschnitt 3.1.1 angesprochenen OSI *Management Knowledge Management Function* hinausgehen. Als erster Schritt hin zu einem CORBA-basierten Abrechnungsmanagement kann der **Licensing Service** angesehen werden, der objekt- und verursacherbezogene Nutzungsstatistiken führt. Die Möglichkeit, Managementaktivitäten transaktionsorientiert auszuführen, wird durch den **Transaction Service** angeboten. Der Mehrwert eines solchen Dienstes für das Management ist jedoch bisher noch nicht abschließend geklärt. Weitere grundlegende CORBAservices befassen sich mit persistenter Speicherung, Nebenläufigkeit, sowie der Bereitstellung eines einheitlichen Zeitbegriffs in CORBA-basierten Systemen.

Mit den CORBAfacilities und den CORBAdomains existiert der konzeptionelle Rahmen, um Managementdienste in CORBA einzuführen. Die konkrete Ausgestaltung dieser Dienste ist aber heute nur teilweise vorhanden, eine Standardisierung durch die OMG lediglich in wenigen Fällen bereits erfolgt.

Treibende Kraft beim Entwurf von **CORBAfacilities für das Systemmanagement**, die für eine große Zahl CORBA-basierter Systeme gelten, war die *Systems Management Working Group* der OpenGroup. Im Rahmen einer vorläufigen Spezifikation [X/Open 95a], die unter dem Namen **X/Open Common Management Facilities (XCMF)** bekanntgemacht und von der OMG im November 1996 angenommen wurde, sind unter anderem folgende Dienste definiert worden:

- Der **Managed Sets Service** erlaubt es, zu Managementzwecken zusammengehörige

[7] In der Praxis besitzt CORBA mangels gegenwärtig implementierter Verteilungs- und Replikationsdienste den Ruf, nicht besonders performant zu sein. Der Problematik, daß CORBA generell über viele standardisierte, jedoch wenige tatsächlich implementierte Dienste verfügt, werden wir noch mehrmals im Laufe dieser Arbeit begegnen.

Objekte zu gruppieren und diese Gruppen zu administrieren. Darunter fallen Tätigkeiten wie die Aufnahme und Streichung von Mitgliedern oder das Einholen von Informationen über Mitglieder einer bestimmten Gruppe.

- Der **Instance Management Service** unterstützt das Lifecycle Management, also das Kreieren oder Löschen administrierter Ressourcen, sogenannter *Managed Instances*.
- Der **Policy Management Service** soll Administratoren dabei helfen, das Verhalten eines Managementsystems ihrer spezifischen Umgebung anzupassen. Mit ihm können Zielvorgaben (*Policies*) zu Objektgruppen definiert und ihre Einhaltung überwacht werden.
- Mit dem **Customization Management Service** können Objekt(-typen) auf eine Art erweitert werden, die es erlaubt, auf solche Objekte auch noch in ihrer alten Form zuzugreifen. Damit kann dann die Interoperabilität von Managementanwendungen und Objektinstanzen verschiedener Versionen sichergestellt werden. Diese Funktionalität bildet das im OSI-Management vorhandene Allomorphie-Konzept in einer CORBA-Umgebung nach.

Obwohl die Systems Management Working Group neben der abschließenden Standardisierung dieser Dienste noch diverse weitere Systemmanagementdienste in die OMG-Referenzarchitektur einbringen wollte, muß man diese Bemühungen im nachhinein als gescheitert ansehen: Seit dem Erscheinen dieser ersten Spezifikation sind keine weiteren Aktivitäten der Arbeitsgruppe mehr bekanntgeworden. Hierfür gibt es mehrere Gründe: Paradoxerweise hat vermutlich der sehr frühe Erscheinungstermin dieser Dienste ihre endgültige Spezifikation verhindert, da zum damaligen Zeitpunkt von den jetzt vorhandenen 18 CORBAservices lediglich vier Dienste verabschiedet waren. Somit wurden in der Zwischenzeit durch die fortschreitende Normierung entsprechender CORBAservices einige XCMF-Dienste wie der *Managed Sets* oder der *Instance Management* Service überflüssig. Für die anderen XCMF-Dienste (wie z.B. den *Policy Management Service*) ist zum Teil auch heute noch unklar, auf welchen CORBAservices diese Dienste aufsetzen können. Somit wurden diese auf einem sehr hohen Abstraktionsniveau definiert, ohne daß das hierzu nötige Fundament vorhanden war. Die zwischenzeitliche Verschmelzung von X/Open mit der Open Software Foundation mag ebenfalls dazu beigetragen haben, daß eine Neuausrichtung der daraus hervorgegangenen Opengroup nicht mehr das Systemmanagement beinhaltet. Die äußerst restriktive Informationspolitik der X/Open, wonach der Zugang zu relevanten Dokumenten lediglich institutionellen Mitgliedern vorbehalten ist, hat das Übrige zur mangelnden Verbreitung beigetragen. Dies alles hat dazu geführt, daß bis heute der dem Systemmanagement vorbehaltene Teil der CORBAfacilities-Spezifikation [OMGCF 98] nur ein knappes Dutzend überblicksartig formulierte Seiten umfaßt, die zudem aus dem Jahre 1995 stammen.

Im Bereich der CORBAdomains gehen seit der OMG-Restrukturierung im Jahre 1995 die Standardisierungsaktivitäten für Managementdienste ausschließlich von der *Telecom-*

munications Domain Task Force aus. Gegenwärtig befaßt sich diese Gruppe mit der Standardisierung des **Notification Service**, der den Event Service um managementspezifische Filtermöglichkeiten für asynchrone Ereignismeldungen erweitert. Als Maßstab dient hier die OSI *Event Report Management Function*. Der Notification Service bietet neben dem Versehen von Meldungen mit Zeitstempeln flexibel konfigurierbare Ereignisfilter, die denen des OSI-Managements ebenbürtig sind. Bei Bedarf kann darüberhinaus die persistente Speicherung von Ereignismeldungen in Logs veranlaßt werden. Die Definition dieses an der OSI *Log Control Function* angelehnten **Logging Service** befindet sich jedoch noch in einem sehr frühen Stadium.

Weitere angedachte Dienste der OMG Telecom DTF beinhalten die Interoperabilität von CORBA mit Intelligenten Netzen, sowie die Schaffung von Übergängen zwischen CORBA und TMN. Die schon begonnenen Arbeiten zum **Topology Service**, dessen Ziel in der Bereitstellung und Modifikation topologischer Beziehungen zwischen Managementobjekten bestand, wurden im Dezember 1997 zurückgestellt, da nicht abschließend geklärt werden konnte, inwiefern neueste Erweiterungen von CORBA (*Portable Object Adapter*, *Meta-Object Service*) bereits Teile der Funktionalität des Topologiedienstes enthalten.

Bewertung

Wir haben in diesem Abschnitt die technischen Eigenschaften vorgestellt, die es CORBA gestatten, neben ihrem Hauptanwendungsgebiet als Architektur für verteilte objektorientierte Programmierung ebenfalls für das integrierte Management eingesetzt zu werden. CORBA spielt zum gegenwärtigen Zeitpunkt eine führende Rolle als Middleware-Architektur und hat dort das *Distributed Computing Environment (DCE)* weitgehend abgelöst. Die Tatsache, daß die Object Management Group mit über 800 Mitgliedsorganisationen zahlenmäßig das weltweit größte Konsortium in der Computerindustrie ist, verdeutlicht die gute Akzeptanz dieser Architektur am Markt.

Während die ehemals mangelnde Akzeptanz von CORBA sowohl an einem unzureichenden Vorhandensein unbedingt notwendiger Dienste als auch an der geringen Verfügbarkeit tragfähiger Implementierungen lag, so ergibt sich nunmehr ein anderes Bild: Die grundlegenden Standards sind seit einiger Zeit stabil; für die Praxis wesentliche Dienste wie die Interoperabilität zwischen ORB-Implementierungen, Sicherheit, Ereignisverwaltung und Namensgebung liegen ebenfalls seit mehr als einem Jahr in endgültiger Form vor. Der noch vor wenigen Jahren mangelhafte Reifegrad der CORBA-Entwicklungsumgebungen hat inzwischen dank der kontinuierlichen Verbesserungen von Seiten der Hersteller zu Systemen geführt, die selbst für unternehmenskritische Anwendungen eingesetzt werden (vgl. hierzu [LLLM 96], [Gome 98]).

Die Auffassung des Managements als verteilte Anwendung führt zu der Erkenntnis, daß es sinnvoll und aussichtsreich ist, CORBA auch als Managementarchitektur zu verwenden, selbst wenn diese Architektur zur Zeit noch nicht den Funktionsumfang „klassischer“ Managementarchitekturen (wie das OSI-Management) erreicht. Einige managementspe-

zifische Dienste sind jedoch bereits standardisiert und die Aktivitäten der Telecom DTF[8] lassen erkennen, daß in naher Zukunft weitere Dienste hinzukommen. Die Tatsache, daß bisher von den standardisierten Diensten erst sehr wenige Implementierungen kommerziell erhältlich sind, erscheint als ein mit der Zeit lösbares Problem.

3.1.3 Die Internet-Managementarchitektur

Organisationsmodell

Das Organisationsmodell der im Bereich des Managements von LAN-Komponenten und -Systemen weit verbreiteten Internet-Managementarchitektur [Rose 96] unterscheidet strikt zwischen Managern und Agenten; die Erweiterung des Organisationsmodells um Mischformen wie Mid-level-Manager befindet sich noch in einem frühen Entwicklungsstadium und wird weiter unten in Zusammenhang mit den Aktivitäten der *Distributed Management Working Group (DISMAN)* besprochen. Die Bildung von Domänen wurde im Rahmen von SNMPv2 ursprünglich mittels sog. *Parties* angedacht, jedoch aufgrund zu hoher Komplexität wieder verworfen. Die dieses Konzept realisierende *Party MIB* ist daher ebenfalls nicht mehr gültig. Gegenwärtig erfolgt die Domänenbildung (*Communities*) implizit durch die Vergabe identischer Lese- und Schreibkennwörter an bestimmte Ressourcenklassen.

Informationsmodell

Der Internet-Managementarchitektur liegt ein im Vergleich zu den bisher vorgestellten Architekturen verhältnismäßig einfaches Informationsmodell (*Structure of Management Information (SMI)*) [rfc1902] zugrunde: Eine MIB-Definition besteht grundsätzlich aus zahlreichen *Object-Type* Makros, die einfache skalare Variablen repräsentieren. Ihrer Anordnung entsprechend werden sie fortlaufend numeriert und bilden so den bereits aus dem OSI-Management bekannten *Registrierungsbaum*, der die eindeutige Namensgebung von Managementobjekten und deren Attributen sicherstellt; skalare Variablen bilden dabei die Blätter des Registrierungsbaumes. Um mehrere Instanzen eines Typs darstellen zu können, verwendet das Internet-Informationsmodell Tabellen, in denen jeweils eine Tabellenzeile Managementinformation zu einer bestimmten Instanz enthält. Die Anzahl der Zeilen einer Tabelle kann sich dabei zur Laufzeit eines Agentensystems ändern. Zur besseren Strukturierung werden inhaltlich zusammengehörige skalare Variablen und Tabellen zu Gruppen zusammengefaßt [PeMc 97]. Im Gegensatz zu den beiden vorgenannten Architekturen verfügt das Internet-Informationsmodell über keinen Klassenbegriff und kennt somit auch keinen Vererbungsmechanismus (also auch keine Vererbungshierarchie) oder etwa ein Allomorphiekonzept, wie es etwa beim OSI-Management der Fall ist. Letzteres wird durch die Konvention nachgebildet, daß jede neue MIB-Version eine Obermenge der

[8] Nicht zuletzt besteht diese Arbeitsgruppe zu einem wesentlichen Teil aus Personen, die bereits bei der Standardisierung des OSI-Managements mitgewirkt haben.

ursprünglichen MIB sein muß. Ebenfalls nicht vorhanden ist ein Management Information Tree, der Enthaltenseinsbeziehungen zwischen Objektinstanzen bestimmt und damit Regeln hinsichtlich der Instantiierbarkeit von Objektklassen vorgibt.

Kommunikationsmodell

Die fehlende Objektorientierung wirkt sich ebenfalls auf das Kommunikationsmodell und das hierfür verwendete *Simple Network Management Protocol (SNMP)* [rfc1905] aus, das demzufolge keinerlei objektklassenbezogene Operationen kennt (wie das Erzeugen und Löschen von Managementobjekten), sondern ausschließlich attributbezogene Operationen wie Lese- und Schreiboperationen (Get, Set). Der ausgiebigen Verwendung von Tabellen in MIBs trägt der get-next-Operator Rechnung, mit dem diese auf einfache Art spaltenweise ausgelesen werden können. Hierbei wird jedoch jedes Element separat durch den Manager angefordert, was beispielsweise beim Auslesen ganzer Tabellen einen nicht unerheblichen Aufwand auf Managerseite darstellt. Der Forderung, die Übertragung größerer Datenmengen mittels eines Kommandos auslösen zu können, wurde in SNMPv2 durch die Einführung der Get-Bulk-Operation begegnet [Stal 98]. Asynchrone Ereignismeldungen eines Agenten werden einem Manager durch die unbestätigte Trap-PDU übermittelt, weshalb nicht sichergestellt werden kann, daß diese auch tatsächlich beim Manager ankommt. Während die Abwägung zwischen Protokolloverhead und Dienstgüte zu Lasten letzterer für einfache Agenten/Manager-Kommunikation noch hinnehmbar ist, so ist dies jedoch bei Managerhierarchien inakzeptabel, da das ereignisgesteuerte Zusammenwirken mehrerer Managementsysteme nur dann erfolgreich sein kann, wenn die Ereignismeldungen auch mit Gewißheit beim Empfänger ankommen. Während manche Hersteller in der Zwischenzeit hierfür eigene Lösungen (vgl. Abschnitt 3.3.1) entwickelt haben, um asynchrone Ereignismeldungen zwischen kooperierenden Managementsystemen zu bestätigen, so existiert für diesen Zweck seit 1993 die Inform-Operation[9]. Das relativ späte Eingehen auf die Notwendigkeit der Kooperation von Managementsystemen und damit die Ausrichtung der Internet-Managementarchitektur auf die Belange großer Netze erklärt sich durch das historische Wachstum des Internet-Managements, das in Abbildung 3.3 darstellt ist[10]. Die grau unterlegten Bestandteile kennzeichnen inzwischen ungültige Elemente des Internet-Managements; die angegebenen Zahlen beziehen sich hierbei auf die Nummern der entsprechenden *Requests for Comments (RFCs)*.

Funktionsmodell

Das Internet-Management kennt kein Funktionsmodell im Sinne generischer, vererbbarer und delegierbarer Managementfunktionalität; es setzt sich vielmehr aus einer Menge von

[9] Explizit untersagt [rfc1909] ist das Aussenden von Inform-PDUs durch Agentensysteme als Mittel zur gesicherten Übertragung von Traps.

[10] Die Darstellung beschränkt sich auf die unmittelbar mit der vorliegenden Arbeit zusammenhängenden Standards.

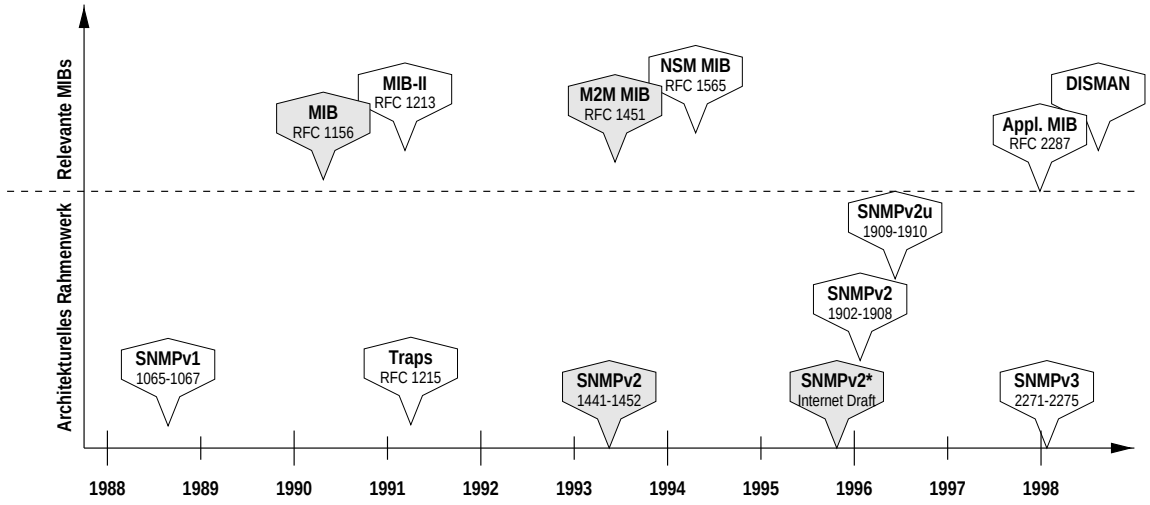


Abbildung 3.3: Die historische Entwicklung der Internet-Managementarchitektur

MIBs zusammen, die ihrerseits geeignet kombiniert werden können. Beispiele hierfür sind die MIB-II [rfc1907], die Host Resources MIB [rfc1514] oder die weiter unten angesprochenen MIBs für das Dienste- und Anwendungsmanagement.

Wir werden im folgenden einige Internet-MIBs vorstellen, die einen unmittelbaren Bezug zu der dieser Arbeit zugrundeliegenden Fragestellung haben.

Die historische Manager-to-Manager MIB

Hinsichtlich der Kooperation von Managementsystemen, wurde im Rahmen der SNMPv2-Vorschläge die Manager-to-Manager MIB (M2M-MIB) [rfc1451] entwickelt, deren Ziel darin besteht, einem Manager den Zugriff auf die Informationen anderer Manager zu ermöglichen. Obwohl sie seit 1996 den Status „historic“ hat und somit offiziell nicht mehr gültig ist, berührt sie das Kernthema der vorliegenden Arbeit und soll daher eingehender betrachtet werden. Ein Managementsystem, das diese MIB implementiert, agiert gegenüber anderen Managern in einer Agentenrolle und bietet diesen folgende Dienste an, die jeweils eine Tabelle der M2M-MIB bilden:

- Die Konfiguration von **Alarmen**: Es kann festgelegt werden, unter welchen Bedingungen der Manager einen Alarm an andere Manager schickt. Die hierzu notwendigen Informationen beinhalten u.a. neben der Object-ID der zu überwachenden Variable sowie deren aktuellen Wert auch die Art der Überwachung (absoluter Wert, Differenz zum vorherigen Meßwert) und die minimalen und maximalen Schwellwerte. Werden diese unter- bzw. überschritten, wird ein **Event** generiert. Die `snmpAlarmTable` der M2M-MIB stellt somit rudimentäre Filtermöglichkeiten zur Verfügung.
- Die Verwaltung dieser **Events** geschieht in der `snmpEventTable`, deren Aufgabe darin besteht, die Reaktionen auf einen Event den jeweiligen Eventtypen zuzuordnen.

- Diese Aktionen wiederum werden in einer weiteren Tabelle definiert, in der diejenigen Manager verzeichnet sind, die sich für ein bestimmtes Ereignis interessieren und die daher durch eine **Notification** benachrichtigt werden sollen. Dies geschieht durch das Aussenden einer Inform-PDU.

Obwohl die Mechanismen zur Weiterleitung von Ereignismeldungen nahezu identisch zu denen der *Remote Network Monitoring (RMON) MIB* sind, besteht die wesentliche Erweiterung darin, daß die überwachten Zielsysteme nicht mehr im selben Netzsegment liegen müssen, sondern frei gewählt werden können. Angesichts der Tatsache, daß es sich bei diesen Zielsystemen ausschließlich um Managementsysteme handelt, die ihrerseits bestimmte Netzbereiche (u.U. mit RMON) verwalten, wäre eine solche Beschränkung auch nicht sinnvoll. Der dreistufige Vorgang der Überwachung, Ereigniserzeugung und -weiterleitung bezieht sich jeweils auf einzelne MIB-Variablen[11], die für eine beliebige Zahl von Ressourcen gelten. Folglich ist es nicht möglich, Schwellwertüberschreitungen mehrerer MIB-Variablen miteinander zu verknüpfen, obwohl dies aufgrund impliziter Filterung zweifellos die Aussagekraft der letztendlich ausgesandten Ereignismeldung erhöhen würde. Eine Verdichtung der Rohdaten zu echter Managementinformation geschieht also nicht; vielmehr ist unter ungünstigen Umständen das Auslösen von „Ereignisstürmen“ möglich.

Der durch ihre Bezeichnung geweckten Erwartungshaltung, durch umfassende Konfigurationsmöglichkeiten die reibungslose Kooperation von Managementsystemen zu erlauben, wird die M2M-MIB nicht gerecht. Letztendlich bietet sie weniger Funktionen als der aus dem OSI-Management bekannte *Event Forwarding Discriminator*, die dort bereits auf der Ebene einfacher Agentensysteme agieren, um bereits möglichst an der Quelle Mechanismen zur Verdichtung von Ereignismeldungen zu realisieren. Dies geschieht mit der M2M-MIB jedoch erst auf der Ebene der (Mid-level-) Managementsysteme. Wie die Analyse der Anforderungen im vorigen Kapitel gezeigt hat, umfaßt der Themenkomplex der Kooperation von Managementsystemen weit mehr als die wechselseitige Konfiguration von Ereignisweiterleitungsmechanismen, der in diesem Falle noch dazu lediglich unvollkommen (nämlich bezogen auf einzelne MIB-Variablen) gelöst wurde. Es ist unmittelbar einsichtig, daß bei einer umfassenden Konfiguration der M2M-Ereignisfilter der Umfang von M2M-MIB-Instantiierungen sehr groß werden kann. Sicherheitsmechanismen wie beispielsweise die Verwaltung von Zugriffsrechten sind ebenfalls nicht Bestandteil der M2M-MIB, da sie sich gänzlich auf das Sicherheitskonzept des zugrundeliegenden Managementprotokolls abstützt. Die starke Abhängigkeit vom SNMPv2 Party-Konzept ist auch der Grund, weshalb die M2M-MIB mit der Aufgabe der SNMPv2-Version von 1993 ebenfalls nicht mehr gültig ist. Über eine neue, erweiterte Version dieser MIB ist derzeit nichts bekannt.

[11] [rfc1451] führt hierzu aus: „[...]An example of an alarm condition is when the monitored variable falls outside a configured range. Each alarm condition triggers an event, and each event can cause (one or more) notifications to be reported to other management stations [...]“

MIBs für das Dienste- und Anwendungsmanagement

Im folgenden besprechen wir MIBs, die als Vorstufe eines SNMP-basierten Dienste- bzw. Anwendungsmanagements betrachtet werden können und somit ebenfalls für unsere Zwecke relevant sind.

Die **Network Services Monitoring MIB** (NSM MIB) [rfc1565] definiert allgemeine Attribute, die ein effektives Monitoring von Netzdiensten, (also speziellen Anwendungen wie Datei-, Druck- oder Verzeichnisdienste) ermöglichen sollen. Wie es der Name bereits ausdrückt, steht hier das passive Monitoring im Vordergrund; Eingriffsmöglichkeiten wie das Setzen von Variablen oder das Auslösen von Aktionen sind nicht vorgesehen. Die MIB besitzt eine Tabelle für die in Frage kommenden Dienste (`applTable`) sowie eine Tabelle für die derzeit offenen Kommunikationsverbindungen dieser Dienste (`assocTable`). Werden die Verbindungen vom Manager hin zum Netzdienst aufgebaut, so werden sie als *inbound associations* bezeichnet; in letzterem Fall (d.h. der betroffene Netzdienst initiiert die Verbindung) als *outbound associations*. Einige Variablen in `applTable` sind nicht spezifisch für das Management von Netzdiensten, sondern gelten prinzipiell für alle Arten von Software. Beispiele für solche Attribute sind `applOperStatus`, das den operationellen Zustand einer Anwendung beschreibt (up, down) und `applUptime`, das die Zeit seit der letzten Instantiierung der Anwendung angibt.

Die **System Application MIB** [rfc2287] erweitert die *Host Resources MIB* [rfc1514] um zwei umfangreiche Gruppen, die Informationen in bezug auf die installierte Software sowie momentan ausgeführte Anwendungen bereitstellen. Die `sysApplInstalledGroup` besteht aus zwei Tabellen, in denen die installierten Softwarepakete sowie deren Bestandteile (d.h. Dateien) verzeichnet sind. Hierfür ist es erforderlich, daß die Softwarepakete mit Hilfe eines speziellen Werkzeuges installiert wurden, das bereits bei der Installation Einträge in der globalen Inventardatenbank des Betriebssystems vornimmt. Bei PC-basierten Betriebssystemen (Windows98, Windows NT) trägt die Setup-Routine diese Informationen in die sogenannte *Registry* ein; das UNIX-Derivat AIX von IBM besitzt hierfür das *System Management Information Tool (SMIT)*, das Einträge in der vom *Object Data Manager (ODM)* verwalteten Systemdatenbank anlegt. Einen Sonderfall stellt das Attribut `sysApplInstallElmtRole` dar, welches für ausführbare Dateien Abhängigkeiten und Einschränkungen über boolesche Werte (z.B. „required“ oder „dependent“) definiert und es so einem Agenten ermöglicht, zur Laufzeit von Anwendungen Aussagen über ihren Status zu machen.

Die zweite Gruppe (`sysApplRunGroup`) besteht aus einer Tabelle, die für jede momentan aktivierte Anwendung einen Eintrag enthält. Dieser Eintrag setzt sich (neben dem für Tabellen obligatorischen Index) aus der Startzeit und dem operationellen Status der Anwendung zusammen. In der MIB ist der Status der Anwendung identisch mit dem Status des zugehörigen Hauptprozesses und kann die Werte „running“, „runnable“, „waiting“, „exiting“ und „other“ annehmen. Eine zweite Tabelle enthält die zu den Anwendungen der ersten Tabelle gehörenden Prozesse sowie die hierfür typischen Managementinformationen (verbrauchte CPU-Zeit, belegter Arbeitsspeicher, Aufrufparameter, Eigentümer usw.).

Beiden Tabellen dieser Gruppe ist eine dritte Tabelle zugeordnet, die Einträge beendeter oder abgebrochener Anwendungen bzw. Prozesse enthält. Befinden sich keine Prozesse der Anwendung mehr in der Prozeßliste, gilt die Anwendung als korrekt (*complete*) beendet. Im anderen Fall muß von einem Fehler ausgegangen werden und der Rückgabewert lautet *failed*. Die Tabellen enthalten folglich ein Log für Anwendungen. Da Logging- bzw. History-Funktionen elementare Managementfunktionen darstellen, die für eine Vielzahl von Anwendungsbereichen gelten, wäre es natürlich sinnvoller, diese Funktionalität in der Form eines abgesetzt implementierten Logging-Dienstes (wie z.B. OSI mit der Log Control SMF) anzubieten. Das Fehlen eines Funktionsmodells und das Nichtvorhandensein von Vererbungsmechanismen im Internet-Management verhindern dies jedoch.

Managementinformation zu den laufenden Anwendungen kann im wesentlichen durch Polling der Prozeßliste gesammelt werden. Um eine hohe Aktualität der Information zu gewährleisten, müßte ein Agent dieses Polling in kurzen Abständen durchführen, was aber dazu führen kann, daß der Agent einen großen Teil der Rechenressourcen des Systems für sich beansprucht. Es ist generell schwierig, allein aufgrund der Außenbezüge einer Anwendung (hier: die laufenden Prozesse) auf ihren inneren Zustand zu schließen und anhand dessen die Ursache einer eventuellen Terminierung zu identifizieren. Noch schwieriger gestaltet sich das Management im verteilten Fall, bei dem sich eine Anwendung aus Komponenten zusammensetzt, die auf verschiedenen Systemen ablaufen. Hier kann effizientes Management nur durch Instrumentierung der Anwendungen selbst mit expliziten Managementschnittstellen erreicht werden. Ein Designziel der System Application MIB bestand jedoch darin, keine Anwendungensinstrumentierung zu fordern. Die unmittelbare Konsequenz daraus ist das Nichtvorhandensein von Eingriffsmöglichkeiten in den Betrieb einer Anwendung [StWe 95]. Ferner liegt der Fokus dieser MIB auf Anwendungen, die sowohl lokal installiert sind, als auch lokal ablaufen. Beides ist in unserem Fall nicht gegeben, weshalb wir bei der Bearbeitung unserer Fragestellung auf lediglich einen kleinen Teil dieser MIB zurückgreifen können.

Die **Application Management MIB**, die bisher noch nicht verabschiedet wurde[12], führt eine dienstorientierte Sicht (*service level view*) auf verteilte Anwendungen ein und setzt – im Gegensatz zu den vorher besprochenen MIBs – entsprechende Anwendungsinstrumentierung voraus. Sie gestattet beschränkte Eingriffsmöglichkeiten in den Betrieb einer Anwendung (Starten, Stoppen) und verfügt hierzu über mehrere Tabellen, die Anwendungen und Dienste sowie deren Ressourcen (geöffnete Dateien, Kommunikationsbeziehungen, Prozesse) verwalten. Eine solche Tabelle ordnet beispielsweise einer Anwendung einen oder mehrere benannte Dienste zu. Eine weitere Tabelle beinhaltet Informationen zu denjenigen Prozessen, welche die Dienste der ersten Tabelle realisieren. Für NFS-, FTP- oder WWW-Server beispielsweise ergeben sich wichtige Leistungsdaten aus der Statistik der Dateizugriffe. Relevante Fragestellungen sind hier „Auf welche Dateien wurde wie oft (lesend oder schreibend) zugegriffen?“ oder „Wie groß war die durchschnittlich übertragene Datenmenge (in Bytes)?“. Hierzu definiert die MIB eine weitere Tabelle

[12] Der entsprechende Internet Draft datiert vom 19. August 1998.

(`applOpenFileTable`) mit Einträgen zu den von einer Anwendung geöffneten Dateien. Die wichtigsten Attribute umfassen: Dateiname, Öffnungszeitpunkt der Datei, Anzahl Lese-/Schreibzugriffe, Anzahl Lese-/Schreibfehler, Anzahl gelesener/geschriebener Zeichen, Zeitpunkt des zuletzt erfolgten Lese-/Schreibzugriffs, Dateigröße. Hierdurch soll einerseits die Bereitstellung grundlegender Daten zur Durchsatz- und Antwortzeitermittlung als Grundlage für das Leistungsmanagement erreicht werden, andererseits sind ebenfalls Informationen für das Konfigurationsmanagement enthalten.

Bei der Application Management MIB wurde versucht, objektorientierte Techniken wie (Mehrfach-)Vererbung über gemeinsame Indizes in den Tabellen zu simulieren. Sie gerät dadurch relativ umfangreich und unübersichtlich, was in erster Linie an der Komplexität der Tabellenindizierung liegt. Es ist gegenwärtig noch nicht entschieden, inwieweit die Hersteller vollständige Implementierungen dieser MIB anbieten werden.

Neue Entwicklungen für modulare SNMP-Agenten

Die möglichen Ausprägungsformen heutigen SNMP-basierten Managements sind in Abbildung 3.4 dargestellt: Damit eine SNMP-Managementplattform spezifische Parameter einer Ressource verwalten kann, muß ihr vorher eine formale Beschreibung der Ressource in Form einer oder mehrerer MIBs bekanntgegeben werden (linker Teil der Grafik). Dies geschieht manuell durch den zuständigen Administrator. Die programmtechnische Instrumentierung der Ressourcen erfolgt gewöhnlich über eine C- bzw. C++ Programmierschnittstelle, die die gesamte Menge an Eingriffsmöglichkeiten darstellt. Um die Unabhängigkeit von einer konkreten Programmiersprache zu gewährleisten, wird die Programmierschnittstelle durch MIBs repräsentiert. Diese Kapselung der Instrumentierung kann dabei auf drei verschiedene Arten erfolgen:

Monolithischer Agent (siehe (1) in Abbildung 3.4): Die von SNMP-Agenten bereitgestellte Managementinformation ist **statisch**, d.h. sie bleibt während des gesamten Lebenszyklus des Agenten konstant. Dies ist charakteristisch für Netzkomponenten wie Router, Switches oder Hubs, die einerseits über die MIB-II sowie eine oder mehrere – zum Teil herstellerspezifische – MIBs verfügen.

Den Anforderungen des Systemmanagements sind monolithische Managementagenten nicht mehr gewachsen, da ein umfassendes Management eines Endsystems (Workstation, PC) weit über das bloße Bereitstellen von hardwarenahen Informationen hinausgeht. Vielmehr müssen Informationen bezüglich des Betriebssystems sowie der darauf ablaufenden Dienste (Namensdienste, Dateidienste) und Applikationen bereitgestellt werden, die naturgemäß der hohen Dynamik von Software unterliegen. Zwar wird der modularen Kombinierbarkeit dieser Dienste und Anwendungen, wie im ersten Fall, durch mehrere MIBs, die ihrerseits auf eine bestimmte Softwarekomponente zugeschnitten sind, begegnet. Der wesentliche Unterschied **erweiterbarer Agenten** ((2) in Abbildung 3.4) zu ersteren besteht jedoch darin, daß die Menge und Ausprägungsform der Dienste und Anwendungen eines Endsystems zur Laufzeit des Agenten wechselt: Applikationen werden installiert bzw. entfernt; eine Workstation übernimmt die Funktion eines Name-, File- oder Webservers bzw.

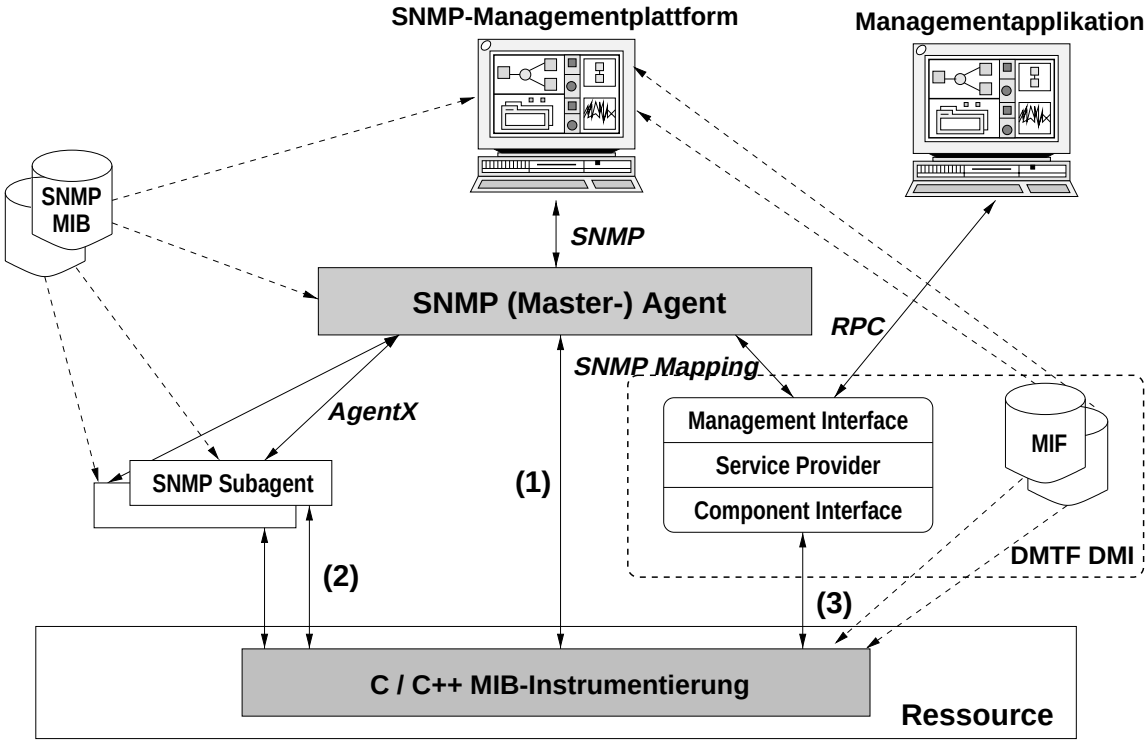


Abbildung 3.4: Möglichkeiten des Zugriffs auf SNMP-basierte MIB-Instrumentierung

gibt diese ab. Gefordert ist daher ein Mechanismus, der es gestattet, SNMP-Agenten zur Laufzeit um neue Managementfunktionalität zu erweitern bzw. diese wieder zu entziehen. Dies geschieht durch das Master- und Subagenten-Konzept, das in den Arbeiten der *Agent Extensibility Working Group (AgentX)* der IETF entwickelt wurde [WhGu 98], welche die Interaktionen der SNMP-Protokollmaschine (Master-Agent) mit gegebenenfalls mehreren MIB-Instrumentierungen (Subagenten) spezifizieren. Das in [rfc2257] standardisierte AgentX-Protokoll definiert neben der Zuordnung und Weiterleitung von SNMP-Anfragen vom Master-Agenten an einen oder mehrere Subagenten ebenfalls das dynamische Laden bzw. Entfernen von MIB-Modulen durch das An- bzw. Abmelden von Subagenten beim Master-Agenten zur Laufzeit. Kommunikation zwischen Subagenten ist im AgentX-Protokoll ausgeschlossen. Mit AgentX realisierte modulare Agenten sind für Managementapplikationen vollkommen transparent, da diese lediglich mit dem Master-Agenten kommunizieren und die Subagenten somit nicht in Erscheinung treten. Der Vollständigkeit halber sei erwähnt, daß sich Master- und Subagenten jeweils auf unterschiedlichen Systemen befinden können, was jedoch in der Praxis kaum angewandt wird.

Der Zugriff auf Ressourcen über das **Desktop Management Interface (DMI)** [DMTF 96], einem in der PC- und Server-Welt verbreiteten Industriestandard der Desktop Management Task Force (DMTF), ist ebenfalls über SNMP möglich (siehe (3) in Abbildung 3.4). Die – in der Regel proprietäre – Schnittstelle zur MIB-Instrumentierung (in der DMTF-Terminologie als **Component Interface (CI)** bezeichnet) wird durch den **Ser-**

vice Provider (SP) gekapselt; das CI beispielsweise spezifiziert Mechanismen, die das dynamische An- und Abmelden von Subagenten beim Master-Agenten gestatten. An einer offengelegten, standardisierten Schnittstelle, dem **Management Interface (MI)**, wird die Managementinformation des SP Managementapplikationen bereitgestellt, die entweder lokal oder von entfernten Systemen (per RPC) darauf zugreifen können. Diese Schnittstelle steht sowohl Master-Agenten als auch Managementapplikationen (unter Umgehung des Master-Agenten) offen, um auf die angebotene Managementinformation zuzugreifen. Die Beschreibung der Managementinformation des MI erfolgt dabei im sog. **Management Information Format (MIF)**, das große Ähnlichkeit zur Internet SMI aufweist[13] und daher anhand von der DMTF spezifizierter Abbildungsregeln problemlos in diese umgewandelt werden kann. Auf Protokollebene wurde ein SNMP-Mapping spezifiziert, so daß DMI-kompatible Managementagenten problemlos in SNMP-Umgebungen integriert werden können.

Die Ziele und Konzepte von DMI, das de facto eine eigenständige Managementarchitektur darstellt, sind sowohl konzeptionell als auch hinsichtlich ihrer technischen Ausführung mit denen des Internet-Managements identisch, weshalb an dieser Stelle auf eine ausführlichere Beschreibung von DMI verzichtet wird. Die Arbeiten an DMI verliefen zeitgleich zur Weiterentwicklung von SNMPv2 und AgentX, was die Ähnlichkeit der Konzepte erklärt. Der eigentliche Nutzwert von DMI besteht aus bereits heute verfügbaren Implementierungen der DMTF-Spezifikationen für PC- und Serverkomponenten sowie der darauf laufenden Software. Demgegenüber steht die Erweiterung des Internet-Managements um MIBs für solche Ressourcenklassen (wie beispielsweise die *Host Resources* und die *Application MIBs*) erst am Anfang. Erste Ansätze werden im folgenden Abschnitt vorgestellt.

Grundsätzlich muß ein Managementsystem wissen, welche Arten von MIBs eine gegebene Ressource unterstützt. Da das Internet-Management keinerlei zu OSI oder CORBA vergleichbare Discovery-Mechanismen kennt, um dies zur Laufzeit von den Ressourcen zu erfragen, müssen die in Frage kommenden MIBs zuvor explizit dem Managementsystem bekanntgegeben werden. In der Praxis wird diesem Problem dadurch begegnet, daß eine Vielzahl standardisierter MIBs im Lieferumfang einer Managementplattform enthalten ist; bei herstellerspezifischen MIBs ist dies natürlich nicht gegeben. Diese müssen durch den Administrator explizit in die Plattformdatenbank geladen werden. Hiermit ist zwar eine Möglichkeit geschaffen, die Anzahl unterstützter MIBs eines Agentensystems dynamisch den Gegebenheiten anzupassen; eine Erweiterung der Funktionalität einzelner MIBs ist jedoch nicht vorgesehen und würde im Übrigen auch nur mit hohem Aufwand auf Seiten des Managementsystems durchführbar sein.

[13] MIF definiert ebenfalls Managementinformation in Form skalarer Attribute, die zu Gruppen zusammengefaßt werden können. Mehrfach instantiierbare Managementobjekte werden analog zur Internet SMI als Tabellen definiert.

Verteiltes SNMP-basiertes Management: DISMAN

Wie bereits oben beschrieben wurde, bestand der erste Ansatz im Internet-Management zur Bildung von Managementhierarchien in der RMON-MIB; mit der Einführung der M2M-MIB wurde dieses Konzept explizit auf Managementsysteme ausgedehnt. Ein denkbarer Ansatz zur Erweiterung SNMP-basierten Managements, um zur Laufzeit von einem Managementsystem Aufgaben an Agentensysteme zu delegieren, wurde in [MEBS 95] vorgestellt. Dies impliziert eine signifikante Erweiterung des Aufgabenumfangs von SNMP-Agenten, da diese sich bisher auf das Bereitstellen und Modifizieren von Managementinformation beschränkten; eine Analyse und Vorverarbeitung der Managementinformation durch Agenten war bisher nicht vorgesehen (Ausnahme: RMON) und ausschließlich den SNMP-Managementsystemen vorbehalten.

Naturgemäß sind die Arbeiten der *IETF Distributed Management Working Group (DISMAN)* mit denselben Problemen konfrontiert wie jede Architektur, die in heterogener Umgebung verteiltes Management realisieren soll. Dementsprechend umfaßt DISMAN folgende Festlegungen:

- Ein Rahmenwerk zur Beschreibung der zu verteilenden Dienste, das sich naheliegenderweise auf das Internet-Informationsmodell abstützt, d.h. die Dienstschnittstellen liegen in Form von MIBs vor. Hier sind zur Zeit ein knappes Dutzend Basisdienste angedacht, die zum einen grundsätzlich in verteilten heterogenen Umgebungen erforderlich und daher in den vorgenannten OSI/TMN- und CORBA-Architekturen ebenfalls präsent sind (z.B. Domänenbildung, Festlegung der Empfänger von Ereignismeldungen). Andererseits gibt es Dienste, deren Notwendigkeit sich aus konzeptionellen Gegebenheiten des Internet-Managements ableitet (z.B. Ermitteln von Zielsystemen, zuverlässige Übermittlung von Ereignismeldungen, Auslösen von ICMP-Nachrichten (*ping* und *traceroute* zur Bestimmung von IP-Konnektivität).
- Ein Protokoll, mit dem die Dienste übermittelt werden. Hierfür ist SNMPv3 vorgesehen, was jedoch aufgrund der beschränkten maximalen SNMP-Paketlänge dazu führt, daß auszuführende Skripten lediglich zeilenweise übertragen werden können und somit in der Regel mehrere SNMP-PDUs notwendig sind, um ein Skript vollständig zu transferieren.
- Eine (Skript-)Sprache, in der diese Dienste implementiert sind. Hier sind gegenwärtig die Arbeiten am weitesten fortgeschritten: Mit der Definition der *SNMP Scripting Language* und der dazugehörigen *Script MIB*, die auf den früheren Arbeiten zur *Mid-Level-Manager MIB* aufbaut, steht neben einer Skriptsprache zur Ausführung von SNMP-Operationen auf Managementobjekten auch ein Instrumentarium zum Laden, Ausführen, Unterbrechen, Wiederaufsetzen, Stoppen und Entfernen von Managementskripten zur Verfügung. Zur Steuerung dieser Vorgänge ist es erforderlich, daß die Zielsysteme einerseits die Script-MIB implementieren und andererseits eine geeignete Laufzeitumgebung in Form virtueller Maschinen bzw. Interpreter für die konkrete Implementierungssprache (z.B. Java, Tcl) bereitstellen.

Die Öffnung des SNMP-basierten Managements in Richtung eines verteilten Managements von Diensten und Anwendungen ist prinzipiell begrüßenswert, da diese Architektur dann über Mechanismen verfügt, die über das reine Komponentenmanagement hinausgehen. Beim Betrachten der oben angesprochenen Ähnlichkeiten von DISMAN mit den vorher angesprochenen OSI/TMN- und CORBA-Architekturen wird jedoch deutlich, daß der DISMAN-Funktionsumfang eine echte Teilmenge der in den anderen Architekturen bereits vorliegenden Dienste darstellt. Konzeptionelle Neuerungen bestehen zum überwiegenden Teil aus der Eingrenzung der Unzulänglichkeiten des zugrundeliegenden Internet-Managements; diese können jedoch (wie z.B. im Falle des stückweisen Übertragens von Skripten) nicht vollständig neutralisiert werden.

Demgegenüber ist die Problematik der Migration von Programmcode mit bestehenden, konkurrierenden Architekturen (CORBA, Java) bereits heute überwiegend gelöst. In der Tat erinnert der DISMAN-Anforderungskatalog ein wenig an das Pflichtenheft zur Spezifikation eines leichtgewichtigen, SNMP-basierten Object Request Brokers. Inwieweit die Erweiterung des Internet-Managements um Mechanismen für das verteilte Management sinnvoll und notwendig ist, muß aufgrund des frühen Entwicklungsstadiums von DISMAN offenbleiben. Bisher liegen sämtliche DISMAN-Dokumente als Internet Drafts[14] vor; die Überführung der Drafts in entsprechende RFCs ist gegen Ende 1998 geplant. Wir stellen im folgenden Kapitel Verfahren vor, die es gestatten, die Dienste einer konzeptionellen starken Managementarchitektur auch auf Architekturen auszudehnen, deren Funktionsumfang geringer ist.

Bewertung

Trotz der kontinuierlichen Weiterentwicklung der Internet-Managementarchitektur bleibt das Problem nicht vorhandener Konsistenzsicherung hinsichtlich verfügbarer Managementinformation zwischen Agenten und Managern auch weiterhin ungelöst: Es ist für ein SNMP-basiertes Managementsystem nicht automatisch ermittelbar, welche über den Umfang der MIB-II hinausgehende Menge an Managementinformation von Agenten angeboten wird. Diese in der Praxis kritische Discovery-Problematik ist sowohl im OSI-Management (mit der Management Knowledge Management Function) als auch in CORBA (DII, Interface Repository, Query Service) gelöst. Es gibt darüber hinaus Anzeichen, daß die Ausdehnung des ursprünglich auf das Komponentenmanagement fokussierenden Internet-Managements auf Bereiche wie das System- und Anwendungsmanagement Schwierigkeiten grundlegender Art hervorruft: Die oben angesprochene Application Management MIB liefert ein gutes Beispiel für die Anwendung des SNMP-Managements auf einen Bereich, für den diese Architektur ursprünglich nicht vorgesehen war. Schließlich sind Netzkomponenten wie Sternkoppler und Router bereits von ihrem Aufbau „tabellenartig“ strukturiert: In die 19-Zoll-Chassis dieser Geräte wird *eine nach oben beschränkte* Zahl von Modulen eingesetzt, die ihrerseits Eigenschaften besitzen, welche leicht durch

[14] Internet Drafts sind jeweils 6 Monate gültige Vorschläge für potentielle Standards.

skalare Variablen instrumentiert werden können. Eine Repräsentation dieser Systeme (deren Informationsmenge überdies verhältnismäßig statisch ist) durch skalare Variablen und Tabellen bietet sich daher an.

Bei abstrakten Softwarekomponenten wie Diensten und Anwendungen ist diese einfache Struktur nun nicht mehr gegeben, da einerseits die Menge ihrer Instanzen *unbeschränkt* und darüberhinaus sehr dynamisch ist. Aufgrund der Beibehaltung der Tabellenstruktur ist man jedoch gezwungen, diese Softwaremodule ebenfalls auf Tabellen, diesmal mit *a priori unbekannter Länge* abzubilden. Hierdurch ergibt sich einerseits die Notwendigkeit, zur Laufzeit Tabellenzeilen hinzuzufügen bzw. zu entfernen, andererseits müssen (Vererbungs-) Beziehungen zwischen diesen Tabellen durch weitere Tabellen dargestellt werden. Diese Tabellen müssen nun konsistent gehalten werden, was in Ermangelung von Mechanismen wie Fremdschlüsseln (wie man sie von relationalen Datenbanksystemen kennt) in der Regel sehr komplex gerät. „Simple“ ist heutzutage de facto nur noch das Protokoll, mit dem auf die Managementagenten zugegriffen wird. Die MIBs selbst (und damit die diese MIBs implementierenden Agenten) erreichen durch die Beibehaltung der Tabellen und der damit einhergehenden komplizierten Indexverwaltung eine beachtliche Größe: Zur Administration der Beziehungen müssen spezielle Tabellen definiert werden, die mit den eigentlichen Ressourcen nichts mehr unmittelbar zu tun haben, sondern lediglich der besseren Indizierung derjenigen Tabellen dienen, die die eigentliche Managementinformation beinhalten.

Ferner bedingt die wechselhafte Geschichte der Internet-Managementarchitektur daß Entwickler, die sich heutzutage mit der Implementierung einer SNMP-basierten Lösung für verteiltes Management befassen, in einem Dilemma stecken: Sie haben die Wahl zwischen einem 6 bis 7 Jahre alten Minimalumfang, dessen Mängel offenkundig sind (vgl. hierzu [Yemi 94] und [Zele 95]) und einer nicht unbeträchtlichen Zahl relativ junger RFCs bzw. Internet Drafts, die durchaus noch Modifikationen unterliegen. Es ist somit nachvollziehbar, daß Hersteller von Ressourcen zumeist die erste Alternative wählen und neue Geräte mit MIBs ausstatten, die lediglich der ersten Version des SNMP-Managements entsprechen.

Zusammenfassend sei an dieser Stelle gesagt, daß gegenwärtig ein breiter Konsens besteht, daß das ursprüngliche SNMP Management Framework für die heutigen Bedürfnisse nicht mehr geeignet ist und grundlegend erneuert werden muß, um die weite Verbreitung dieser Architektur auch in Zukunft zu gewährleisten. Dies betrifft wesentliche Aspekte (wie die vorher erwähnten modularen Managementagenten sowie DISMAN) und führt damit zu einer umfassenden Restrukturierung und Neuausrichtung dieser Managementarchitektur. Jedoch hat wohl kaum eine Aktivität in der Internet Engineering Task Force zu solch großer Uneinigkeit geführt, wie die Frage, *wie* ein neues SNMP-basiertes Management aussehen soll. Beispielhaft sei an die zweite Version des SNMP-Managements (SNMPv2) erinnert, das zuerst in den RFCs 1440 bis 1450 im Jahre 1993 standardisiert, 1997 durch eine neue Version (RFCs 1901-1910) überflüssig wurde und durch das gegenwärtig in der Abschlußphase befindliche SNMPv3 ersetzt werden soll, in das die Konzepte der oben

angesprochenen Arbeitsgruppen nahtlos integrierbar sein werden. In [Harr 97] findet man einen guten Überblick über die historische Entwicklung von SNMPv1 bis SNMPv3.

Herstellerseitig ist jedoch nach den negativen Erfahrungen mit SNMPv2 die Bereitschaft, ein weiteres Mal Investitionen für Implementierungen einer neuen SNMP-Version bereitzustellen, erheblich eingeschränkt. Es ist daher gegenwärtig noch nicht entschieden, inwieweit das Internet-Management seine bisherige Bedeutung beibehalten kann; die in den vorigen bzw. folgenden Abschnitten skizzierte Vielzahl alternativer, konkurrierender Managementarchitekturen erschweren den Übergang zu SNMPv3 zusätzlich. Das Internet-Management scheidet somit sowohl aufgrund konzeptioneller Schwächen als auch aufgrund der Unsicherheit bezüglich seiner weiteren Entwicklung gegenwärtig als Kandidat für eine Enterprise Management Architektur aus.

3.1.4 Open Distributed Processing

Während bisher technisch orientierte Implementierungsarchitekturen für das Management im Mittelpunkt unserer Betrachtung standen, fehlte bisher jedoch ein verbindliches, standardisiertes Rahmenwerk, das die Entwicklung allgemeiner Anwendungskomponenten (in unserem Fall: Managementsysteme, -applikationen und Agenten) erlaubt, die in offener, heterogener und verteilter Systemumgebung interagieren können. Die ISO begann 1987 mit der Arbeit an einem entsprechenden Modell und standardisierte 1991 zusammen mit der ITU-T eine Architektur für den Entwurf von objektorientierten verteilten Systemen in heterogener Umgebung. Ergebnis dieser Überlegungen ist das **Reference Model of Open Distributed Processing** (RM-ODP). Diese ISO-Norm, deren jüngste Fassung aus dem Jahr 1995 datiert, besteht aus insgesamt vier Teilen [ISO 10746-1, ISO 10746-2, ISO 10746-3, ISO 10746-4].

Die Offenheit von ODP impliziert, daß Programmierschnittstellen und Kommunikationsprotokolle offengelegt und standardisiert sein müssen. Die Unterstützung von heterogenen, verteilten Systemen durch das Modell bezieht sich auf Rechner und Netzkomponenten verschiedener Hersteller, heterogene Betriebssysteme und Kommunikationsprotokolle sowie den Einsatz unterschiedlicher Programmier- und Datenbanksprachen für die verteilten Anwendungen. RM-ODP adressiert die Probleme, die sich bei der organisatorischen Struktur von Anwendungskomponenten ergeben. Fragestellungen sind hierbei z.B. „*Welche* Objekte erbringen einen benötigten Dienst und *wo* befinden sie sich?“, „*Wie* kann der Dienst genutzt werden?“, „*Wie* können Dienste organisiert werden, damit der Benutzer den Eindruck gewinnt, es handelt sich um eine einzige integrierte verteilte Anwendung?“ und „*Welche* Unterstützung benötigen Anwendungskomponenten von der darunterliegenden verteilten Systemplattform?“.

RM-ODP ist ein Meta-Modell, welches durch Konzepte, Regeln und die Defintion von Infrastrukturobjekten eine Architektur für verteilte Anwendungen in heterogener Systemumgebung beschreibt. Ziel ist es, beim Entwurf eines verteilten Systems die *Komponenten* (Objekte) identifizieren zu können, die an spezifizierten und typisierten *Schnittstel-*

len Dienste anbieten. Diese können durch Interaktionen an den Schnittstellen, d.h. durch Kommunikation auf Anwendungsebene genutzt werden. Zwei weitere wichtige Aspekte sind *Portabilität* und *Transparenz*. Damit Anwendungskomponenten portabel sind, bedarf es standardisierter verteilter Plattformen. Weiterhin sollen Details der Verteilung vor den Komponenten verborgen bleiben. Hierzu sind Mechanismen erforderlich, die verschiedene Transparenzen (*access, concurrency, location, migration, replication, failure, etc.*) bereitstellen (vgl. hierzu die Ausführungen in Abschnitt 2.3.1).

Aufgrund der Komplexität ist der Entwurf eines verteilten Systems mittels einer einzelnen Spezifikation schwierig, wenn nicht sogar unmöglich. Deshalb führt das RM-ODP unterschiedliche Sichten (*Viewpoints*) auf das System ein, um die Komplexität nach dem „divide et impera"-Prinzip zu reduzieren. Sie spiegeln diejenigen Sichten wider, die verschiedene Personen auf ein verteiltes System haben: Auftraggeber definieren anhand ihrer Unternehmensziele die Anforderungen an das System, Software-Architekten entwerfen darauf aufbauend das Design sowie die Spezifikation des Systems, das wiederum von Programmierern implementiert wird. Schließlich erfolgt die Installation des fertigen Systems durch Techniker. ODP **Viewpoints** sind orthogonale Sichtweisen auf die unterschiedlichen Aspekte eines verteilten Systems. Um Redundanz und daraus entstehende Inkonsistenzen zu vermeiden, soll die Zerlegung durch Viewpoints möglichst disjunkt sein. **Viewpoint Languages** legen das Vokabular und die Grammatik fest, die zur Beschreibung der Information eines Viewpoints eingesetzt werden. Jeder Viewpoint enthält Konzepte, die das Vokabular der zugehörigen Sprache darstellen, und Regeln (*Structuring Rules*), die die Grammatik der Sprache bilden. Welche konkrete (Spezifikations- bzw. Programmier-) Sprache schließlich als Viewpoint Language eingesetzt wird, läßt das Modell offen [BlSt 97]. Voraussetzung ist lediglich, daß die Syntax und Semantik der gewählten Sprache die Konzepte und Regeln der Viewpoint Language ausdrücken können. RM-ODP definiert folgende fünf Viewpoints:

- **Enterprise Viewpoint:** Hier wird die Gesamtumgebung für das System sowie sein Zweck beschrieben. Außerdem werden die Anforderungen (*Requirements*) an das System, zu erfüllende Bedingungen (*Constraints*), ausführbare Aktionen (*Actions*) und DV-Zielvorgaben (*Policies*) aus Unternehmenssicht definiert.

- **Information Viewpoint:** Dieser Viewpoint legt die Struktur und Semantik der Informationen des Systems fest. Weitere Punkte sind die Definiton von Informationsquellen und -senken von sowie die Verarbeitung und Transformation von Information durch das System. Hierzu gibt es Integritätsregeln und Invarianten. Beispiele für *Information Languages* sind die *Object Modeling Technique (OMT)* [RBPE 91] (siehe Kapitel 5) und OSI GDMO (siehe Abschnitt 3.1.1).

- **Computational Viewpoint:** Hier wird ein System in logische, funktionale Komponenten zerlegt, die für die Verteilung geeignet sind. Das Ergebnis sind Objekte, die Schnittstellen besitzen, an denen sie Dienste anbieten bzw. nutzen. Die meisten ob-

jektorientierten Sprachen, wie z.B. C++, Java, Smalltalk und auch IDL eignen sich als *Computational Language*.

- **Engineering Viewpoint:** Dieser Viewpoint beschreibt die erforderliche Systemunterstützung, um eine Verteilung der Objekte aus dem Computational Viewpoint zu erlauben. Hierzu werden generische Infrastrukturobjekte eingeführt, die die oben genannten Verteilungstransparenzen realisieren, um eine Kommunikation der verteilten Objekte zu ermöglichen. So entsteht das Modell einer generischen, objektorientierten, verteilten Plattform.
- **Technology Viewpoint:** Dieser Punkt beschreibt die Wahl konkreter Technologien zur Implementierung und Realisierung des Systems. Hierin enthalten ist sowohl die Spezifikation der Hardware (Hersteller und Modell der Rechner, Netzkomponenten, etc.) als auch der Software (Betriebssystem, Kommunikationsprotokolle und Programmiersprachen). Da die Implementierung eines ODP-Systems auf unterschiedlichen technischen Plattformen möglich sein soll, sollten zwischen diesem und den anderen Viewpoints keine Abhängigkeiten bestehen.

Die vorliegende Arbeit beschäftigt sich insbesondere mit der Untersuchung des Computational sowie des Engineering Viewpoints, da deren Konzepte diejenigen Objekte beschreiben, die für das System- und Anwendungsmanagement (und damit für die in dieser Arbeit behandelte Fragestellung) relevant sind. Wir machen uns dabei insbesondere die Tatsache zunutze, daß im Rahmen dieser beiden Viewpoint-Sprachen Begriffe definiert bzw. Anforderungen an deren Verhalten formuliert werden, die für verteiltes Management eine große Bedeutung haben. Beispiele für solche Begriffe sind: Systeme, Prozesse, Kommunikationsbeziehungen, Schnittstellen usw. Wir werden auf die von uns verwendeten ODP-Konzepte in Kapitel 5 noch genauer eingehen.

Bewertung

Wie bereits oben angesprochen, handelt es sich bei RM-ODP um einen *Meta-Standard*, der ein Rahmenwerk für allgemeine verteilte Verarbeitung darstellt, um den Einsatz *spezifischer* OO-Technologien zu ermöglichen. Beispiele solcher Technologien sind die vorher besprochenen Managementarchitekturen bzw. Rahmenwerke für verteilte objektorientierte Programmierung. Somit liegt ODP auf einem höheren Abstraktionsniveau als die bisher besprochenen Architekturen.

Die Begründung für die Verwendung von RM-ODP im Rahmen dieser Arbeit lautet wie folgt: Wir hatten bei der Besprechung der OSI-Managementarchitektur festgestellt, daß diese zwar im Rahmen der *Generic Management Object Classes* generische MOCs definiert, jedoch beziehen sich diese ausschließlich auf Elemente der sieben Schichten des OSI-Referenzmodells wie z.B. Protokollinstanzen oder Dienstzugangspunkte. GMOCs, die für unsere Belange relevant wären, d.h. Managementsysteme, deren Bestandteile und Dienste, existieren bisher in keiner Managementarchitektur. Da wir jedoch solche generischen MOCs als Basisklassen für die MOCs verteilter kooperativer Managementsysteme

(und deren Bestandteile) benötigen, sind wir auf standardisierte Vorgaben angewiesen, um die universelle Anwendbarkeit unseres Lösungsansatzes sicherzustellen. RM-ODP leistet genau dies, indem durch die Definition von Anforderungen an verteilte Anwendungen exakt solche allgemein verwendbaren Basis-MOCs implizit festgelegt werden. Wir werden diesen Ansatz in Kapitel 5 genauer vorstellen. Er ist ebenfalls in [KeNe 97c] dokumentiert.

3.1.5 Telecommunication Information Networking Architecture

Das **Telecommunication Information Networking Architecture Consortium (TINA-C)** wurde 1993 durch zahlreiche Telekommunikationsunternehmen mit dem Ziel ins Leben gerufen, eine Software-Architektur zu entwickeln, die die schnelle und flexible Einführung von neuen Telekommunikationsdiensten erlaubt. Ein weiterer, wesentlicher Aspekt von TINA umfaßt das integrierte Management dieser Dienste sowie der Kommunikationsnetze, innerhalb derer diese Dienste vorhanden sind. Hierbei soll von der Transportinfrastruktur abstrahiert werden, um unabhängig von den Übertragungstechnologien und -komponenten zu sein. Diese Unabhängigkeit wird durch die Einführung einer Middleware, dem sog. **Distributed Processing Environment (DPE)**, realisiert, die die transparente Interaktion von Softwarekomponenten über Domänengrenzen erlaubt. Ein natürlicher, aber nicht ausschließlicher DPE-Kandidat ist CORBA, das aufgrund seines Potentials und seiner Verbreitung vom TINA-Konsortium favorisiert wird [PTBH 98]. TINA kann als eine Spezialisierung des RM-ODP für die Telekommunikation aufgefaßt werden, die beispielsweise im Rahmen der *Binding Architecture* Mechanismen für den Aufbau und die Administration von Kommunikationsverbindungen bereitstellt.

TINA unterscheidet zwei Arten von Diensten, einerseits *Nutzdienste bzw. Telekommunikationsdienste*, die dem Kunden oder Endanwender zur Verfügung stehen (z.B. flexible Gebührenaufteilung zwischen Anrufer und Gegenstelle, gebührenfreie Anrufe usw.) und andererseits Dienste, welche Betrieb, Administration und Wartung der Nutzdienste gestatten, also *Managementdienste*. Analog zu CORBA sind auch hier die Mechanismen identisch, mit denen Telekommunikations- und Managementdienste entworfen, spezifiziert, implementiert und angeboten werden. Ein wesentlicher Aspekt besteht darin, daß TINA für die Bereitstellung von Telekommunikationsdiensten eine ganzheitliche Betrachtungsweise verfolgt: Ausgehend von einem *Enterprise Model* eines Dienstes, das die Aktoren sowie deren Beziehungen zueinander definiert, werden die Sichtweisen auf einen solchen Dienst anhand der weiteren RM-ODP Viewpoints beschrieben, um so zu einer den Anforderungen entsprechenden Implementierung zu gelangen.

Aufgrund ihres geringen Alters konnte bei der Festlegung von TINA auf Konzepte zurückgegriffen werden, die bereits in TMN (siehe Abschnitt 3.1.1), ODP (siehe Abschnitt 3.1.4) sowie den Intelligenten Netzen [MaPo 96] enthalten sind (vgl. auch [Hama 97]). Die TINA-Gesamtarchitektur setzt sich aus den folgenden vier Teilarchitekturen zusam-

men und ist ausführlich in der Dokumentation beschrieben, welche zum Abschluß des Forschungsvorhabens[15] im Dezember 1997 publiziert wurde:

- Die **Computing Architecture** legt diejenigen Methoden fest, mit denen TINA-konforme Applikationen unabhängig von einer konkreten Vermittlungstechnologie modelliert, spezifiziert und implementiert werden. Dies geschieht durch Rückgriff auf die *Information*, *Computational* und *Engineering* Viewpoints des RM-ODP, mit denen die Eigenschaften des DPE festgelegt werden.
- Innerhalb der **Service Architecture** werden Konzepte und Prinzipien definiert, die für Spezifikation, Analyse, Wiederverwendbarkeit, Design und Betrieb von Telekommunikations-Softwarekomponenten relevant sind.
- Ziel der **Network Resource Architecture** ist die generische, technologieunabhängige Beschreibung von Kommunikationsnetzen und deren Elementen. Diese generischen Objektklassen stammen zum überwiegenden Teil aus dem Objektkatalog der OSI/TMN-Architektur [ITU M.3100], der als Basis der Vererbungshierarchie für elementspezifische Klassen dient. Das *Network Resource Information Model (NRIM)* [Nata 98] beinhaltet somit neben MOCs für das Verbindungsmanagement ebenfalls Managementinformation zur Konfiguration von Ressourcen sowie zur Identifikation und Beseitigung von Fehlern [FKWW 95].
- Die **Management Architecture** spielt bei TINA eine Querschnittsrolle, d.h. jede der vorgenannten Teilarchitekturen ist selbst für die Überwachung und Steuerung derjenigen Ressourcen verantwortlich, die in ihrem Umfeld liegen. Somit fällt der TINA-Managementarchitektur die Aufgabe zu, Prinzipien, Modelle und Mechanismen sowohl auf der Grundlage der TINA-Modelle zu den Information und Computational Viewpoints zu definieren, als auch bereits existierende Management-Informationsmodelle (z.B. das OSI/TMN-Management) zu nutzen. Die TINA-Managementaktivitäten werden ferner unterteilt in:
  - *Computing Management*, welches einerseits auf die Aspekte des klassischen Systemmanagements abstellt und die Überwachung und Steuerung derjenigen Systeme, Kommunikationsplattformen und Transportmechanismen zum Ziel hat, auf denen die TINA-Applikationen aufsetzen. Dieser Teilaspekt wird im Kontext von TINA auch als *Infrastruktur-Management* bezeichnet.
    Andererseits befaßt sich TINA unter dem Begriff *Software-Management* ebenfalls mit Anwendungsmanagement. Hierunter fallen Tätigkeiten wie die Installation, Konfiguration, Aktivierung und Deinstallation von Softwarekomponenten, aus denen verteilte Anwendungen bestehen.
  - *Telecommunications Management* umfaßt die Überwachung und Steuerung von Telekommunikationsdiensten sowie derjenigen Dienste, die für das Management

[15] Während das TINA „core team“ Ende 1997 aufgelöst wurde, werden die Arbeiten an TINA im Rahmen von informellen Arbeitsgruppen fortgesetzt.

dieser Dienste zum Einsatz kommen. Unter diese Managementkategorie fällt ebenfalls das Management der Telekommunikationsnetze.

Das Ziel von TINA besteht besteht langfristig in der Ablösung von TMN [Proz 97]. Ein Forschungsvorhaben, das sich mit der Schaffung eines Migrationspfades von TMN zu TINA befaßt, wird in Abschnitt 3.2.4 vorgestellt.

Bewertung

Seine Bedeutung erhält TINA insbesondere durch die ganzheitliche Betrachtung zur Definition, Modellierung und Bereitstellung von Telekommunikationsdiensten. Hierbei stützt sich TINA zur Modellierung von Diensten sowohl auf das ODP-Referenzmodell sowie das OSI-Management ab (hierbei insbesondere die Informations- und Funktionsmodelle), als auch auf CORBA als Implementierungsarchitektur zur Erreichung der Verteilung. Aufgrund des geringen Alters von TINA liegen derzeit erste Implementierungserfahrungen ausschließlich in Form von Forschungsprototypen vor. Ein Beispiel hierfür findet man in [GrPa 97]. Aussagen über die Eignung dieser Architektur im Betrieb lassen sich daher gegenwärtig nicht treffen.

Eine grundsätzliche Problematik besteht darin, daß TINA lediglich sehr allgemein gehaltene Festlegungen macht, so daß die Wahlfreiheit und der Spielraum für die Anwender dieser Architektur ausgesprochen hoch ist. Obwohl TINA explizit das Management von Endsystemen und Anwendungen zu seinem Aufgabenumfang zählt, so sind doch die für diese Zwecke notwendigen Möglichkeiten bisher nicht vorhanden. Diese Architektur bietet daher keine unmittelbare Unterstützung für die in dieser Arbeit behandelte Problematik; ihre Eignung als Architektur für das Enterprise Management kann aufgrund des Fehlens entsprechender Tragfähigkeitsnachweise nicht abschließend beurteilt werden.

3.1.6 Open Distributed Management Architecture

Für den Bereich System- und Anwendungsmanagement entwickelt die ISO auf dem RM-ODP aufbauend die **Open Distributed Management Architecture** (ODMA) [ITU X.703]. Dieser Standard beschreibt eine Architektur zur Spezifikation und Entwurf von verteilten Managementanwendungen wie auch von Applikationen für das Management von verteilten Anwendungen. ODMA erweitert die Konzepte und Managementfunktionen der OSI *Systems Management Architecture* [ISO 10040] auf offene, verteilte Systeme und stützt sich dabei in weiten Teilen auf RM-ODP ab [Sido 98]. Begleitdokumente zu ODMA beinhalten Übersichten zur Angleichung der ODP-Terminologie an die Begriffswelt des OSI-Systemmanagements. Ein ODMA-System wird mit Hilfe der fünf RM-ODP Viewpoints beschrieben wobei ODMA als Notation zur Beschreibung von Objekten des Information Viewpoints sowohl OMT als auch GDMO sowie das OSI *General Relationship Model (GRM)* als Notation für Computational Objects verwendet. Schnittstellendefinitionen konkreter Implementierungen können entweder in OMG IDL oder in GDMO

deklariert werden, da sowohl OSI als auch CORBA als Kandidaten zur Implementierung von ODMA-Systemen vorgesehen sind. Interaktionen zwischen Objekten des Engineering Viewpoints erfolgen protokollorientiert, wobei diese sich an den CMIS-Dienstelementen anlehnen.

Die Abstützung auf OSI/TMN und CORBA ist der Ursprung für die Fähigkeiten von ODMA, sowohl potentiell verteilte Ressourcen wie Netzkomponenten, Systeme und Anwendungen zu überwachen und zu steuern sowie die hierzu notwendigen Managementaktivitäten ebenfalls verteilt zu erbringen. Desweiteren beinhaltet ODMA Mittel zur dynamischen Delegierung von Managementaktivitäten durch Manager an Agenten sowie Funktionalität zur Koordination verteilter Managementaktivitäten. Die Verwendung von RM-ODP bringt die Erreichung von Transparenz mit sich; diese Transparenz bezieht sich dabei sowohl auf die verwendeten Kommunikationsprotokolle, als auch auf die in Abschnitt 2.3.1 besprochenen allgemeinen Aspekte der Verteilung, wie z.B. die Portabilität von Managementapplikationen. Im Computational Viewpoint der ODMA sind Manager und Agent Computational Objects mit Managementschnittstellen. Es handelt sich hierbei stets um sog. *operational Interfaces*. Ein Manager besitzt eine Client-Schnittstelle, über die er Operationen wie `get(attr)` oder `set(attr)` auf einer Server-Schnittstelle eines Agenten ausführen kann. Umgekehrt kann der Agent asynchrone Ereignisse über seine Client-Schnittstelle an den Manager senden. Dieser empfängt die Meldungen an seiner Server-Schnittstelle. Ereignisse werden also wie Managementoperationen modelliert, die von Agenten ausgehen. Dieses Vorgehen entspricht exakt der Zustellung von Ereignismeldungen in CORBA. Bevor diese Interaktionen stattfinden können, müssen Manager und Agent entsprechende Bindungen eingehen.

Die Erarbeitung von Migrationskonzepten von gegenwärtigen proprietären oder TMN-basierten Managementlösungen hin zu ODMA zählt ebenfalls zum Aufgabenumfang der für ODMA zuständigen Gruppe, jedoch liegen diese zum gegenwärtigen Zeitpunkt noch nicht vor.

Bewertung

Die Vorversion zum ODMA-Standard beinhaltet im wesentlichen eine Mischform aller gegenwärtig aktuellen Technologien für verteilte Anwendungen sowie deren Management (OSI/TMN, RM-ODP, CORBA). Zudem erscheint diese Architektur erst zu einem relativ späten Zeitpunkt und bietet auch in konzeptioneller Hinsicht wenig Neues: Während diese Basisarchitekturen entweder bereits seit längerer Zeit im Einsatz sind bzw. den letzten Schliff für den Einsatz in realen Umgebungen erhalten, ist ODMA zwar standardisiert, jedoch bisher weitestgehend unbeachtet geblieben: Gegenwärtig sind keinerlei Forschungsprojekte bekannt, die sich mit ODMA-Implementierungen befassen. Konkrete Produkte sind somit zumindest mittelfristig nicht zu erwarten. Ferner finden sich in ODMA keinerlei Festlegungen, wie die beträchtlichen Unterschiede zwischen OSI/TMN und CORBA überbrückt werden können, um den geforderten ganzheitlichen Managementansatz realisieren zu können. Das Fehlen eines effektiven Mehrwerts von ODMA für die dieser Arbeit

zugrundeliegende Fragestellung sowie der unausgereifte Entwicklungsstand bedingt, daß wir von einer Verwendung dieser Architektur absehen müssen.

3.1.7 Web-based Enterprise Management

Die bisher neueste Managementarchitektur, die aufgrund ihrer expliziten Ausrichtung auf das Enterprise Management für unsere Arbeit relevant ist, ist die **Web-based Enterprise Management (WBEM)** Initiative, die im Jahre 1996 von den Firmen BMC Software, Compaq, Cisco Systems, Intel und Microsoft ins Leben gerufen wurde und heute insbesondere von Seiten der Hersteller favorisiert wird. Ziel von WBEM ist die Definition eines einheitlichen Rahmenwerkes zur Administration von Netzkomponenten, Endsystemen und Anwendungen. Diese sollen von einer integrierten Oberfläche aus unter Verwendung identischer Mittel (bezogen auf das Datenmodell sowie die Zugriffsmechanismen und Managementdienste) auf einheitliche Art administriert werden können. Ein weiteres Designziel besteht in der Integration „fremder“ Managementarchitekturen, wie dem in Abschnitt 3.1.3 besprochenen Desktop Management Interface (DMI), dem Internet-Management und dem OSI-Management. Hierbei ist insbesondere angedacht, daß WBEM als Architektur für das Managementsystem zum Einsatz kommt, welches über geeignete Proxies mit Agenten kommuniziert, die aus anderen Architekturwelten stammen. Die Kooperation zwischen einem WBEM-basierten Managementsystemen und anderen Managern ist bisher nicht explizit vorgesehen. Insgesamt besteht WBEM aus folgenden Komponenten [Thom 98]:

Das Herzstück von WBEM ist der sogenannte **CIM Object Manager (CIMOM)**, der im wesentlichen unserem Begriff eines verteilten kooperativen Managementsystems entspricht. Die Speicherung der Daten von CIMOM (wie z.B. die MIBs der verwalteten Ressourcen, Instanzen-Informationen, Zugriffsrechte) geschieht im **CIM Repository**, welches bisher in Form von Dateien realisiert ist. Als Benutzerschnittstelle zum CIMOM fungieren WWW-Browser, in denen die WBEM-Managementapplikationen ablaufen. Der WBEM-Begriff **Provider** umfaßt sowohl unseren (vgl. Abschnitt 2.1) Agentenbegriff als auch Proxies, über die Agenten aus anderen Managementarchitekturen angesprochen werden: WBEM sieht sowohl Provider für XML (s.u.) und die Windows Registry vor, als auch für die Fremdprotokolle SNMP und DMI. Das in Abschnitt 3.1.3 angesprochene und erst zwei Jahre alte DMI ist insofern interessant, als WBEM die Rolle des DMI-Nachfolgers zugedacht ist.

Als Kommunikationsmechanismus zwischen CIMOM und Providern einerseits und zwischen CIMOM und den Managementapplikationen andererseits war ursprünglich das **Hypermedia Management Protocol (HMMP)** gedacht; mittlerweile ist man jedoch davon abgekommen und favorisiert stattdessen das im WWW eingesetzte **Hypertext Transfer Protocol (HTTP)**, mit dem in der **Extensible Markup Language (XML)** beschriebene Daten transportiert werden sollen. Alternativ zu HTTP kann jedoch auch die Microsoft-Objektechnologie ActiveX eingesetzt werden können, die jedoch gegenwärtig nur auf Windows-Betriebssystemen ablauffähig ist.

Auch im Bereich des WBEM-Informationsmodells haben sich einige Veränderungen ergeben: Ursprünglich sollte das *Hypermedia Management Scheme (HMMS)* als Informationsmodell genutzt werden. Nach der Übernahme der WBEM-Aktivitäten durch die *Desktop Management Task Force* wurde HMMS durch das *Common Information Model (CIM)* [DMTF 98] ersetzt, das ursprünglich aus dem Bereich des Software-Engineering stammt. Als Notation zur Beschreibung von Managementinformation wurde das *Managed Object Format* eingeführt, das gewisse Ähnlichkeiten mit der Syntax des Internet-Informationsmodells hat.

Bewertung

Der Kurzüberblick über WBEM hat aufgezeigt, daß seit Gründung der WBEM-Initiative vor knapp zwei Jahren nicht nur das Informationsmodell ausgetauscht wurde, sondern auch das Kommunikationsmodell grundlegende Modifikationen erfahren hat. Angesichts der Tatsache, daß WBEM bisher keine allgemeinen Managementdienste spezifiziert hat (d.h. bisher kein Funktionsmodell besitzt) und das Organisationsmodell dem des Internet-Managements sehr ähnlich ist, kann man davon sprechen, daß quasi die gesamte Architektur innerhalb kurzer Zeit nahezu vollständig ausgetauscht worden ist, was deren Beurteilung aufgrund des offensichtlichen Mangels an Referenzimplementierungen naturgemäß erschwert.

Der derzeit mangelnde Reifegrad von WBEM und die starke Abhängigkeit von der Vermarktungsstrategie des gegenwärtigen Marktführers im Bereich der PC-Software spiegelt sich auch an folgendem Beispiel wider: Während die Vorversion des Microsoft WBEM Software Development Kits, der bisher einzigen WBEM-Entwicklungsumgebung, noch die Möglichkeit vorsah, in Java programmierte Software zu unterstützen, wurde dies in der Endfassung der ersten Version wieder zurückgezogen. Dies mag zwar aus vermarktungspolitischen Gründen zur Abschottung bestehender Marktsegmente einsichtig erscheinen; der Offenheit von WBEM ist dies jedoch abträglich. Ferner zeigen solche Schritte die Abhängigkeit der WBEM-Strategie von den individuellen Zielen der Mitglieder des WBEM-Konsortiums auf. Angesichts der gravierenden Richtungsänderungen, die diese Architektur innerhalb sehr kurzer Zeit erfahren hat, gestalten sich Prognosen über die zukünftige Entwicklung von WBEM als ausgesprochen schwierig.

Angesichts dieser Randbedingungen erscheint die Positionierung von WBEM als Architektur für das Enterprise Management nicht gerechtfertigt, da der geringe Reifezustand sowie das Fehlen einer nachvollziehbaren technischen Vision gegen einen unternehmensweiten Einsatz von WBEM sprechen.

3.1.8 Zusammenfassung: Managementarchitekturen

Während das am längsten bestehende OSI-Management auch heute noch die ausgefeiltesten Mechanismen für effizientes Management großer Kommunikationsnetze bietet (Scoping und Filtering, dezentrale Ereignisverarbeitung) und mit Abstand über den größten

Umfang an generischer Managementfunktionalität verfügt, so steht der umfassenden Verbreitung dieser Architektur vor allem die aus der Verwendung des Managementprotokolls CMIP resultierende hohe Komplexität entgegen. Abhilfe gegen die Verwendung des umständlichen XOM/XMP ist erst seit kurzer Zeit in Gestalt der TMN/C++ Programmierschnittstelle erfolgt. Außerdem bedingt der Einsatz von CMIP das Vorhandensein eines verhältnismäßig umfangreichen Protokollstacks (ACSE, ROSE). Des weiteren ist in kommerziellen Managementprodukten lediglich ein geringer Anteil der spezifizierten SMFs implementiert. Die enge Verzahnung von TMN mit OSI zeigt einerseits die Fokussierung auf das Telekommunikationsmanagement auf und signalisiert andererseits das Eingeständnis des Scheiterns dieser Architektur im Bereich der Datenkommunikation. Ein weiterer Kritikpunkt, denen sich das OSI-Management ausgesetzt sieht, ist der Mangel an Verteilungstransparenz: Zur Adressierung von MOs muß grundsätzlich deren Ort bekannt sein. Ferner gestattet OSI nur eine grobgranulare Verteilung, da lediglich vollständige Agenten anstelle einzelner Managementobjekte verteilt werden können.

Einige der in OSI explizit definierten Managementdienste (Object Management Function, MKMF) sind in CORBA bereits implizit vorhanden. Als eine der Stärken von CORBA existieren vielfältige Abbildungsmechanismen der Notation zur Beschreibung von Managementobjekten auf sämtliche relevanten Programmiersprachen: Den mehr als einem halben Dutzend standardisierten *Language Mappings* hat beispielsweise das OSI-Management lediglich die Unterstützung von zwei Programmiersprachen (C, C++) entgegenzusetzen. In jüngster Zeit ist außerdem eine zunehmend engere Verflechtung von ODP und CORBA festzustellen, was die enge Kopplung zwischen einem abstrakten Rahmenwerk für verteilte Anwendungen und einer konkreten Implementierungsarchitektur fördert. Zwei Beispiele illustrieren dies: Zur Spezifikation der *ODP Trading Function* [ISO 13235] existiert in CORBA der mittlerweile standardisierte Trading Service. Die OMG Interface Definition Language ist seit 1997 im Rahmen der ODP-Normierung zu einer ITU-Richtlinie sowie zu einem ISO-Standard avanciert [ITU X.920]. Der vermutlich wichtigste Vorteil von CORBA beruht jedoch auf der Tatsache, daß sowohl Nutz- als auch Managementdaten auf einheitliche Art modelliert und implementiert werden und somit das gesamte Instrumentarium von Software-Engineering-Methoden und -Werkzeugen auch für das Management genutzt werden kann. Wir werden davon im weiteren Verlauf dieser Arbeit regen Gebrauch machen.

Während TINA im Bereich der Telekommunikation positioniert wird, ist der konzeptionelle Mehrwert dieser Architektur im Vergleich zu ODP/CORBA zur Zeit umstritten, da ein wesentlicher Anteil von TINA von eben diesen beiden Architekturen stammt. Hinsichtlich der Marktakzeptanz von TINA ist festzustellen, daß diese zumindest für einen kurz- und mittelfristigen Zeitraum äußerst fraglich ist: Die zentrale Ursache hierfür ist in der Tatsache zu sehen, daß der Mehrwert selbst einer ausgereiften Architektur wie TMN erst seit relativ kurzer Zeit von den Telekommunikationsgesellschaften anerkannt ist und gegenwärtig hauptsächlich eine Migration von proprietären hin zu TMN-basierten Managementsystemen erfolgt. Ferner sind erste TMN-Managementsysteme erst seit kurzer Zeit

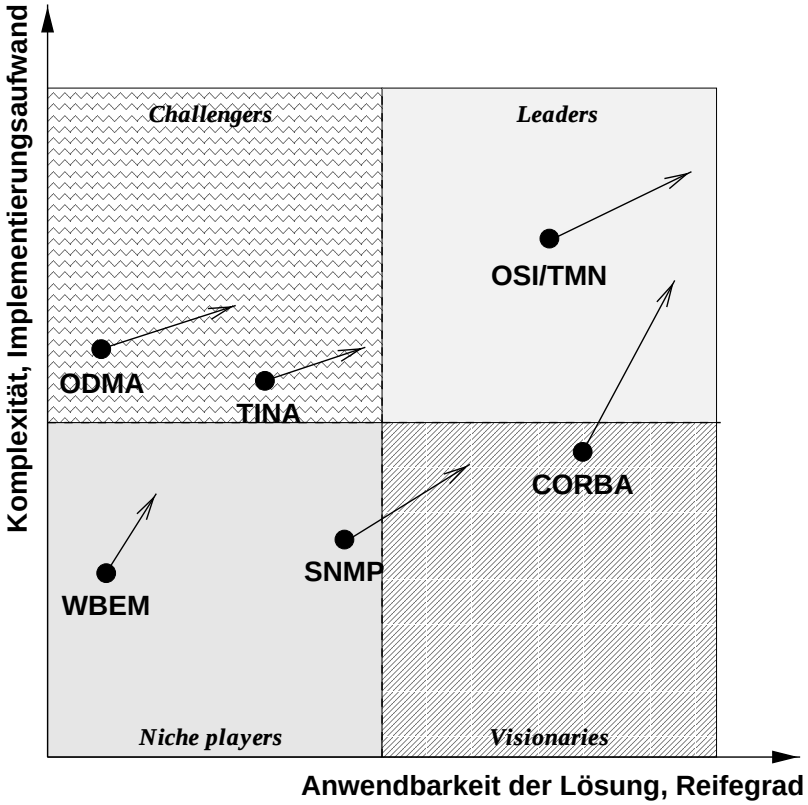


Abbildung 3.5: Bewertungsmatrix für Managementarchitekturen

auf dem Markt. Es bleibt daher offen, inwiefern die Betreiber von Telekommunikationsnetzen die Bereitschaft aufbringen werden, innerhalb eines kurzen Zeitraums eine weitere kostspielige Migration ihrer Managementsysteme durchzuführen.

Die zu TINA gemachten Anmerkungen treffen in verstärkter Form auch auf ODMA zu, da diese beiden Architekturen von ihrer Ausrichtung nahezu identisch sind: Nicht zuletzt beruht auch ODMA zu einem nicht unbeträchtlichen Anteil auf OSI/TMN, RM-ODP und CORBA.

WBEM bleibt gegenwärtig seinem Anspruch schuldig, für unternehmensweites Management eingesetzt zu werden, da diese Architektur einerseits weitestgehend auf PC-Betriebssysteme fokussiert ist und andererseits (mit Ausnahme des Internet-Managements) keine tragfähigen Pläne vorliegen, fremde Managementarchitekturen zu integrieren. Dem Argument einer breiten Unterstützung von WBEM durch zahlreiche Hersteller ist ebenfalls mit Vorsicht zu begegnen, da insbesondere große Hersteller oft in vielen (oft konkurrierenden) Konsortien mitarbeiten, um mit Gewißheit Produkte für diejenige Architektur anbieten zu können, die sich letztendlich durchsetzen wird. Dies gibt andererseits auch Aufschluß darüber, daß auch etablierte Hersteller zum gegenwärtigen Zeitpunkt nicht mit Bestimmtheit sagen können, welche Architekturen für das integrierte Management unternehmensweiter Netze am aussichtsreichsten sind. Problematisch ist

ferner der sprunghafte Austausch maßgeblicher Komponenten innerhalb ausgesprochen kurzer Zeiträume.

Um den momentan sehr unübersichtlichen Bereich der Rahmenwerke für das Management etwas anschaulicher darzustellen, haben wir die in den vorigen Abschnitten besprochenen Managementarchitekturen in Anlehnung an die Vier-Felder-Matrix der Gartner Group in Abbildung 3.5 hinsichtlich ihres gegenwärtigen Standes sowie möglicher Entwicklungstendenzen klassifiziert, um ihre jeweilige Relevanz zumindest mittelfristig zu prognostizieren. Der Nutzen eines solchen Vergleichs wird ebenfalls durch folgendes Argument unterstrichen: Die bisherigen Ausführungen implizieren eine Aufteilung des Marktes für Managementarchitekturen auf zwei bisher nahezu disjunkte Bereiche, nämlich die der Daten- sowie der Telekommunikation. Mit dem Zusammenwachsen der Daten- und Telekommunikation fallen die beiden Anwendungsbereiche zukünftig zusammen. Somit sind sämtliche bisher besprochenen Architekturen potentielle Konkurrenten auf demselben Marktsegment.

3.2 Forschungsansätze mit Bezug zur Aufgabenstellung

Die Problematik zentralisierten Managements ist in zahlreichen Publikationen in bezug auf die gegenwärtigen Managementarchitekturen eingehend behandelt worden. Es besteht in der Fachwelt ein breiter Konsens darüber, daß der Schlüssel zur Bewältigung der zunehmenden Anforderungen in der Verteilung der Managementaufgaben auf mehrere Systeme liegt. Abschnitt 3.1 hat allerdings auch aufgezeigt, daß die Ansichten, *wie* diese Verteilung zu erfolgen hat und welche (Management-)Architektur dafür prädestiniert ist, divergieren. Der vorige Abschnitt hat ebenso einige Begründungen für die Verwendung des OMG-Frameworks geliefert.

Dieser Abschnitt untersucht nun, welche Vorarbeiten zur Verwendung von CORBA für Managementbelange bereits geleistet wurden und inwiefern diese zur Lösung der in Kapitel 1 beschriebenen Fragestellung dieser Arbeit herangezogen werden können.

3.2.1 Management by Delegation

Ein inzwischen sehr bekannter Ansatz zur Verteilung von Managementfunktionalität beruht auf dem **Management by Delegation (MbD)**-Paradigma, das erstmals im Jahre 1991 in [YeGY 91] vorgestellt wurde und seitdem kontinuierliche Verfeinerungen erfahren hat (siehe [GoYe 95] und [Moun 97]) bzw. auf unterschiedliche Problemstellungen angewandt worden ist (vgl. hierzu beispielsweise [MEBS 95] und [GoYe 98]).

Das Prinzip von MbD beruht auf der Überlegung, daß die Skalierbarkeit des Managements signifikant erhöht werden kann, wenn anstelle des Transfers von Rohdaten zu einem

zentralen Managementsystem Teile der Funktionalität dieses Managementsystems an die Ressourcen delegiert werden. Neben dem Nachweis der Gültigkeit dieser Annahme anhand diverser Managementszenarien definiert Goldszmidt in [Gold 96] ein Rahmenwerk zur Delegierung und wendet dieses auf das SNMP-basierte Management an. MbD gestattet die dynamische Verteilung von Managementfunktionalität sowie deren Ausführung zur Laufzeit auf Agentensystemen. Um dies zu ermöglichen, müssen die Agentensysteme über eine Laufzeitumgebung verfügen, die in der Lage ist, die delegierten Managementskripten (unter der Kontrolle durch das Managementsystem) auszuführen. Solche Agentensysteme, deren Funktionsumfang dynamisch erweitert bzw. verringert werden kann, werden als **Elastic Servers** bezeichnet.

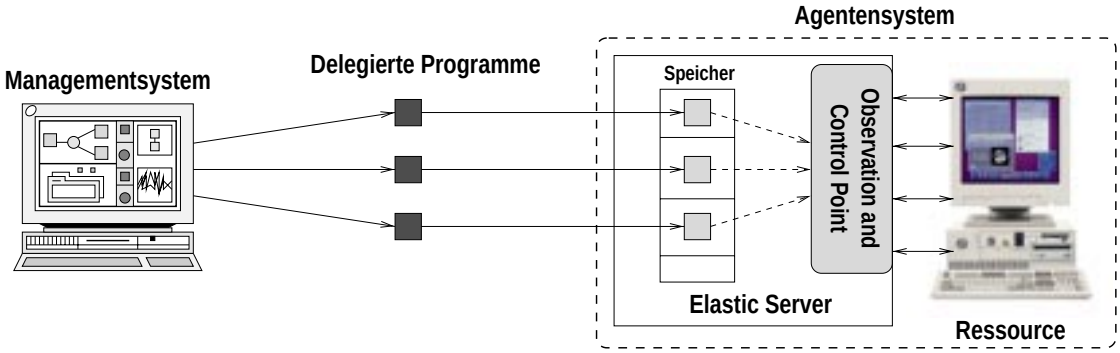


Abbildung 3.6: Management by Delegation

Der *Management by Delegation*-Ansatz geht über die eher softwaretechnische Problematik der dynamischen Erweiterung von Managementagenten hinaus: Es umfaßt vielmehr die Definition eines vollständigen Rahmenwerkes zur Delegierung von Managementfunktionalität, das in folgende Bestandteile gegliedert werden kann (siehe Abbildung 3.6):

- Hier sind an erster Stelle die **delegierten Programme** zu nennen, also die eigentliche Managementfunktionalität, die in einer vom Laufzeitsystem der Agenten ausführbaren Sprache beschrieben sein muß. Der Sprachumfang des ersten MbD-Prototypen bestand aus einer Teilmenge von ANSI C, die die Ausführung externer, d.h. von der Laufzeitumgebung bereitgestellter Prozeduren erlaubt. Diese Prozeduren, deren Umfang zur Laufzeit verändert werden kann, müssen jeweils in einer Konfigurationsdatei aufgeführt sein. Denkbar sind natürlich auch andere Programmier- bzw. Skriptsprachen, die vom Laufzeitsystem ausgewertet werden können. Ein Designziel von MbD bestand darin, die Wahl der Programmiersprache bewußt offenzulassen, um keinen Einschränkungen ausgesetzt zu sein.
- Weitere wesentliche Komponenten sind das **Delegierungsprotokoll** sowie die dazugehörigen **Delegierungsdienste** zur Steuerung der delegierten Programme. Dienstelemente des Delegierungsprotokolls bestehen aus Diensten zur Verteilung delegierter Programme (*Delegate*), deren Aktivierung bzw. Löschung (*Instantiate, De-*

lete) und zur Kontrolle ihres Ablaufs (*Suspend, Resume, Terminate*). Ebenso zählen Dienstelemente zum Austausch von Nachrichten (*SendMsg, ReceiveMsg*) zum Umfang des Delegierungsprotokolls.

- **Observation and Control Points** stellen die Managementinstrumentierung für eine Ressource durch Abstraktion von ressourcenspezifischen Funktionen zur Verfügung. Sie bilden folglich die Schnittstelle, mit der delegierte Programme auf die von einer Ressource bereitgestellten Informationen zugreifen können.

Um auf Seiten des Managementsystems die zu delegierenden Dienste identifizieren zu können, müssen diese über einen eindeutigen Namen verfügen; eine architektur- bzw. implementierungsunabhängige Beschreibung des Dienstumfangs ist ebenfalls erforderlich. Die zu delegierende Funktionalität muß in einem offengelegten Format beschrieben sein, das von den jeweiligen Zielsystemen interpretiert bzw. ausgeführt werden kann.

Bewertung

MbD ist hinsichtlich konkreter Programmiersprachen und Protokolle bewußt neutral gehalten; die prototypische Implementierung von Goldszmidt basiert auf der Internet-Managementarchitektur, kann jedoch problemlos auch auf andere Architekturen übertragen werden. Nichtsdestoweniger erbt der MbD-Prototyp durch seine Wahl von SNMP als Delegierungsprotokoll zwangsläufig die in Abschnitt 3.1.3 diskutierten Nachteile SNMP-basierten Managements. Während einige der in Abschnitt 3.1 beschriebenen Managementarchitekturen deutlich bessere Voraussetzungen für MbD als das Internet-Management mitbringen, ließen die zum damaligen Zeitpunkt verfügbaren Alternativen jedoch keine andere Wahl zu.

MbD beruht auf dem Prinzip, daß jeweils das Managementsystem sowohl entscheidet, welche Funktionalität an Agentensysteme delegiert werden soll, als auch die Kontrolle über die Ausführung delegierter Programme hat und die Ergebnisse von Agentensystemen in Empfang nimmt. Das bedeutet, daß Agentensysteme über keinerlei Autonomie bei der Auswahl erforderlicher Funktionalität verfügen und ein Managementsystem jederzeit über Informationen hinsichtlich des Delegierungskontextes verfügen muß, was einen hohen Aufwand auf Seiten des Managementsystems impliziert.

Insgesamt ist MbD ein praktikabler und vielversprechender Ansatz, der insbesondere mit dem Aufkommen von Programmiersprachen wie Java und dem Vorhandensein virtueller Maschinen für eine Vielzahl von Plattformen eine wichtige Rolle zur Erreichung skalierbaren Managements spielt. Nicht zuletzt deshalb eignet sich MbD sehr gut für die dieser Arbeit zugrundeliegende Fragestellung. Auf die Anwendung des MbD-Prinzips auf unsere Aufgabenstellung werden wir in Kapitel 6 eingehen.

3.2.2 Generic Management Architecture

Unter dem Titel **Generic Management Architecture (GEMA)** wurde in [Schr 96] eine Architektur veröffentlicht, die darauf abzielt, eine „Verteilungsschicht“ oberhalb der existierenden OSI- und Internet-Managementprotokolle zu definieren, die von den Spezifika konkreter Managementarchitekturen abstrahiert, um den auf dieser Schicht aufsetzenden Managementapplikationen eine einheitliche Ablaufumgebung zu bieten. Hiermit sollen Aspekte wie Offenheit, Verteilung und Integrationsfähigkeit erreicht werden, die wir in der Anforderungsanalyse 2.3 identifiziert haben.

Hierzu definiert GEMA ein auf OMG IDL aufbauendes Informationsmodell, das zur Integration des OSI- und Internet-Managements die Transformation dieser Informationsmodelle in das GEMA-Modell erfordert. Dazu werden formale Abbildungsregeln definiert. Als wesentliche Komponente des GEMA-Kommmunikationsmodells wird der **Party Interaction Dispatcher** eingeführt, mit dessen Hilfe ortransparente Kommunikation zwischen Managementsystemen und Agenten realisiert wird. Die hierbei eingesetzten Mechanismen haben große Ähnlichkeiten mit gewöhnlichen Object Request Brokern.

Bewertung

Die Vielfalt der in Abschnitt 3.1 vorgestellten Architekturen läßt Zweifel aufkommen, inwieweit das Erfinden einer neuartigen Managementarchitektur im Zeitalter halbjährlich erscheinender neuer und durch große Herstellerkonsortien forcierte Management-Rahmenwerke noch gerechtfertigt ist. Nicht zuletzt wird hierdurch zwangsläufig die Interoperabilitätsproblematik mit existierenden Managementarchitekturen ein weiteres Mal aufgeworfen.

Die Ausgestaltung der zentralen Elemente von GEMA weist große Ähnlichkeiten mit CORBA auf. Die vom Autor vorgebrachten Vorzüge von GEMA gegenüber CORBA in Bezug auf Sicherheit, Konformität zu Managementprotokollen und Zuverlässigkeit mögen zum Zeitpunkt der Untersuchung ihre Berechtigung gehabt haben; mit dem heute vorliegenden CORBA-Funktionsumfang sind diese Aussagen jedoch nicht mehr aufrechtzuerhalten. Nicht zuletzt aus diesen Gründen müssen wir von einer Verwendung von GEMA im Rahmen dieser Arbeit absehen.

3.2.3 ESPRIT-Projekt MAScOTTE

Im Zuge des vierten europäischen Rahmenprogramms zur Förderung der Forschung in informationsverarbeitenden Disziplinen (ESPRIT) begann im November 1995 ein Projekt mit zweijähriger Laufzeit, das den Titel **MA**nagement **S**ervi**c**es for **O**bject-oriented dis**T**ributed sys**TE**ms (MAScOTTE) trägt und folgende Teilnehmer umfaßt: Bull (Frankreich, Hersteller), CSELT (Italien, Anwender), ESA/ESRIN (Italien, Anwender), Fraunhofer IITB (Deutschland, Forschung – Systemintegrator), Iona (Irland, Hersteller) und MARI (Großbritannien, Systemintegrator).

Das Ziel von MAScOTTE [Masc 97] liegt in der Definition, Spezifikation und Implementierung von Managementdiensten für verteilte objektorientierte Systeme. Als Architektur zur Anwendung dieser Managementdienste ist CORBA vorgesehen. Das dem Projekt zugrundegelegte Szenario besteht aus den Anforderungen, die Anwender CORBA-basierter Systeme an das Management haben sowie der Definition hierfür geeigneter Dienste. Folglich konzentrieren sich die Untersuchungen weniger auf die Administration von Netzen und Systemen, sondern auf das Management der Middleware (also der CORBA-Infrastruktur selbst) und die darauf aufbauenden (CORBA-konformen) Anwendungen. Hierzu wurde in MAScOTTE eine von den Objektklassen der X/Open *Managed Set Facility* (vgl. die Ausführungen zum CORBA-Funktionsmodell in Abschnitt 3.1.2) abgeleitete sogenannte CORBA-MIB definiert, die u.a. in [UsBr 97] genauer erläutert wird. Folgende Komponenten eines CORBA-konformen Systems sind potentielle Managementobjekte: Object Request Broker, Client-Objekte, Application Objects, CORBAservices, CORBAfacilities. Beispiele für relevante Managementattribute sind: Interface-Name, Anzahl und Referenzen von Instanzen pro Objektklasse, Aktivierungsmodus und Typinformationen zu Instanzen, Rechenzeitverbrauch und aktive Verbindungen pro Instanz. Die Intention des Projekts besteht darin, eine für alle ORBs einheitliche Menge an Instrumentierung bereitzustellen, die durch Vererbung produktspezifisch verfeinert werden kann.

Das MAScOTTE-Konsortium favorisiert CORBA-basiertes Management (CORBA-konforme Agenten und Manager), schließt jedoch ausdrücklich Managementsysteme, die auf anderen Managementarchitekturen basieren (hier: das OSI-Management), mit ein. Diese Form von Outband Management ist notwendig, da einige der oben angeführten MO-Kandidaten weitreichende Eingriffsmöglichkeiten in die Funktionsweise eines Object Request Brokers gestatten. So soll es beispielsweise möglich sein, die Ursachen für den Zusammenbruch eines ORBs aufzuspüren und zu beseitigen. Dies ist mit ausschließlich CORBA-basiertem (d.h. Inband-) Management nicht möglich.

Um Managementoperationen durchführen zu können, setzt MAScOTTE voraus, daß Entwickler sowohl eines ORBs, als auch einer verteilten Anwendung entsprechende Managementinformation in OMG IDL bereitstellen. Die Summe dieser Managementinformation bezeichnet MAScOTTE als CORBA-MIB; ihre Struktur erhält sie durch die Verwendung von Vererbungs- und Enthaltenseinsbeziehungen. Solcherart definierte Managementinformation wird von Diensten ausgewertet, die ihrerseits von Managementapplikationen aus unterschiedlichen Architekturwelten genutzt werden können. Solche universell nutzbaren Managementdienste werden im Kontext von MAScOTTE als **Management Facilities** bezeichnet und werden in zwei Klassen eingeteilt:

1. **CORBA Framework Management Facilities** beinhalten Dienste für das Management des ORB-Kernsystems.
2. **Facilities for managing object aggregates** können wiederum in zwei Kategorien eingeteilt werden:

   2.1 **Generic Object Management Facilities** sind auf alle (Client- und Server-) Ob-

jekte anwendbar.

2.2 **Specific Service Management Facilities** beziehen sich auf das Management von CORBA-Diensten. Im Gegensatz zu den anderen Ausprägungsformen von Management Facilities sind diese nicht Gegenstand von MAScOTTE.

Beispiele für im Rahmen von MAScOTTE entstandene Management Facilities sind:

- CORBA Agent Facility: Generisches (d.h. nicht auf konkrete Produkte zugeschnittenes) Management der CORBA-Infrastruktur.
- Notification Facility: Monitoring von Kommunikationsbeziehungen mit Filter- und Loggingmöglichkeiten.
- Application Definition Facility: Bereitstellung einer Managementsicht auf vollständige Applikationen, d.h. Definition von Gruppen zusammengehöriger Objekte.
- Connectivity Test Facility: Prüfen von Kommunikationsverbindungen zwischen Objekten.
- CORBA Discovery Facility: Ermitteln der Enthaltenseinshierarchie eines gegebenen CORBA-Agenten.

Bei den beiden letzteren handelt es sich um Hilfsdienste („Experimental Facilities“) für die drei erstgenannten Dienste; ihre Spezifikation beschränkt sich daher auf den im Rahmen des Projekts notwendigen Umfang, d.h. es werden nur diejenigen Teilaspekte von Diensten spezifiziert und implementiert, die unmittelbar für die Funktionsweise des in diesem Projekt entwickelten Systems erforderlich sind.

Ein weiterer Teil von MAScOTTE befaßt sich mit der Nutzung der CORBA-basierten Managementdienste von einer kommerziellen – also nicht CORBA-konformen – Managementplattform aus (hier: ISM/OpenMaster von Bull). Hierzu ist es notwendig, die Dienstbeschreibungen in die Darstellungsweise der Plattform (hier: OSI SMI) zu überführen und die Dienstaufrufe der Plattform (hier: M-GET, M-SET, M-ACTION, M-EVENT-REPORT) auf die entsprechenden Objektdienste abzubilden. Auf der Plattform laufen Managementapplikationen, die zu einem Teil generisch (Monitor, Alarm, Event, Performance) und zum anderen Teil auf die CORBA-konforme Dienstumgebung zugeschnitten sind (Connectivity Analysis, Monitoring Policies, Reporting Policies, Routing Policies).

Aus den Ergebnissen von MAScOTTE ist neben einem CORBA MIB-Browser [IITB 96] ein Produkt zur Administration eines kommerziellen Object Request Brokers hervorgegangen. Es handelt sich dabei um den *OrbixManager* [IONA 97] der Firma Iona, der Dienste für das Management von *Orbix* bereitstellt, einem weit verbreiteten CORBA-2.0-konformen ORB derselben Firma. Das Ziel von OrbixManager besteht darin, die managementrelevanten Parameter von Orbix in Form von IDL-Schnittstellen zu beschreiben, um sie einem Managementsystem in Form einer offengelegten *Management Library* zur Verfügung zu stellen. Sie besteht einerseits aus passiven Informationen wie zum Beispiel

Monitoring-Parametern des ORB-Kerns zur Definition von Schwellwerten und zur Erstellung von Meßverfahren. Andererseits stehen sämtliche Konfigurationsparameter des ORBs, die ansonsten in zwei Konfigurationsdateien abgelegt sind, über IDL-Schnittstellen dem (aktiven) Management zur Verfügung. Im Idealfall ist jeder ORB in einem verteilten Rechenverbund mit einer solchen Managementschnittstelle ausgestattet. Auf diese CORBA-Objekte können nun Managementdienste, wie z.B. allgemeine, von der OMG standardisierte Dienste (vgl. hierzu Abschnitt 3.1.2) oder spezialisierte, auf dieses Produkt abgestimmte Dienste (z.B. Verteilung der Konfigurationsparameter auf alle ORBs eines Verbundes, Entdeckung eines Absturzes von ORB-Applikationen, Schwellwertüberwachungen) angewendet werden. [Cros 97] führt dies genauer aus.

Für die Administration des OrbixManager stehen eine graphische Benutzerschnittstelle sowie Kommandozeilenwerkzeuge zur Verfügung. Da die meisten gegenwärtigen Managementsysteme bislang keine CORBA-Schnittstellen besitzen, gehört ebenfalls ein SNMP Proxy zum Umfang von OrbixManager, der eine Konvertierung der CORBA-requests in SNMP-PDUs vornimmt. Diese Umsetzung beschränkt sich jedoch auf diejenigen MIB-Variablen, die unmittelbar Bestandteil von OrbixManager sind; es ist nicht möglich, diesen Proxy um andere MIBs zu erweitern.

Bewertung

Insgesamt stellt das MAScOTTE-Projekt einen der ersten in die Praxis umgesetzten Implementierungsansätze für CORBA-basiertes integriertes Management dar. Obwohl der Fokus dieses Projekts auf dem Management von Object Request Brokern liegt und damit auf den ersten Blick lediglich mittelbar mit der Fragestellung dieser Arbeit zusammenhängt, stellt die Instrumentierung von Object Request Brokern zur Sicherstellung des fehlerfreien Betriebs der Kommunikationsinfrastruktur eine wichtige Basis für die in der vorliegenden Arbeit entwickelten Lösung dar. Außerdem konnte die Relevanz der von uns entworfenen Konzepte und Dienste anhand der parallel in MAScOTTE erzielten Ergebnisse verifiziert werden:

- Die Nutzung von IDL als Notation zur Beschreibung von Managementinformation sowie die Abstützung auf standardisierte Managementdienste gestatten die Realisierung offener Managementlösungen.
- Der Notwendigkeit der Integration in bestehende Managementarchitekturen wurde durch die Entwicklung statischer Proxy-Agenten Rechnung getragen. Deren Funktionsumfang ist jedoch a priori festgelegt und kann, im Gegensatz zu den von uns entwickelten und in Kapitel 4 vorgestellten Lösungen, nicht dynamisch um neue Managementinformationen erweitert werden.
- Ein zur *CORBA Discovery Facility* vergleichbarer Dienst wurde im Zuge unserer prototypischen Implementierung eines *multiarchitekturellen Managers* entwickelt, da es notwendig ist, die innere Struktur von CORBA-konformen Managementagenten zu ermitteln. Abschnitt 4.2 führt dies genauer aus.

Durch die Fokussierung von MAScOTTE auf das Management einer noch in der Entwicklungsphase befindliche Technologie wurde zwar durch einen bottom-up Ansatz eine reichhaltige Menge an Managementinformation identifiziert. Jedoch bleibt offen, inwieweit die entwickelten Komponenten um neue Anforderungen von Seiten der Administratoren (Top-down-Sichtweise) erweitert werden können, und zwar insbesondere um Aspekte der Instrumentierung CORBA-konformer Applikationen. MAScOTTE beschränkt sich hier auf die Weise, wie ein ORB eine solche Anwendung sieht, nämlich als voneinander unabhängige Objekte. Es werden somit keine Vorgaben hinsichtlich der minimalen Menge an erforderlicher Managementinformationen gemacht; das Management von CORBA-Applikationen ist daher relativ rudimentär: So ist beispielsweise die Feststellung der Existenz bestimmter Objektinstanzen zwar eine notwendige, jedoch nicht hinreichende Begründung für die Funktionsfähigkeit einer verteilten Anwendung.

3.2.4 EURESCOM-Projekt P508

Das Projekt P508[16] des *European Institute for Research and Strategic Studies in Telecommunications (EURESCOM)* befaßt sich mit den Aspekten der Verwendung von CORBA als Middleware für Telekommunikationsnetze. Ziel des Projektes ist, eine Vorstellung zu entwickeln, ob, wann und wie TINA (siehe Abschnitt 3.1.5) in den europäischen öffentlichen Telekommunikationsnetzen eingesetzt werden kann und mögliche Migrationsszenarien, ausgehend von heutigen Intelligenten Netzen, zu entwerfen. Die Motivation hierzu liegt in der Ähnlichkeit der *funktionalen* Aspekte von Telekommunikations- und Datennetzen, also der Mechanismen für Betrieb, Administration, Management und Bereitstellung von Diensten; demgegenüber bestehen Unterschiede bei *nicht-funktionalen* Anforderungen wie zum Beispiel Verfügbarkeit (tolerierbare Ausfallzeiten von wenigen Minuten pro Jahr) und Skalierbarkeit (hinsichtlich einer zweistelligen Millionenanzahl von Benutzern) sowie das Vorhandensein von Echtzeitanforderungen.

Die zwei zentralen Fragestellungen des Projekts lauten daher (vgl. auch [P508 97]):

1. Welche Verfahren erlauben eine „sanfte“ Migration von IN zu TINA bzw. wie kann die Interoperabilität zwischen klassischen IN und TINA sichergestellt werden?

2. Um welche Belange müssen Object Request Broker erweitert werden, um für Telekommunikationszwecke eingesetzt werden zu können?

Die Migration von IN zu TINA impliziert den sukzessiven Austausch von IN-Komponenten, wobei zur Sicherung bereits getätigter Investitionen naheliegenderweise mit denjenigen Komponenten begonnen werden sollte, die in relativ kleiner Anzahl vorhanden sind. Kandidaten hierfür sind **Service Management Functions (SMF)**, **Service Control Functions (SCF)** oder **Service Data Functions (SDF)**. In großen Stückzahlen

[16] Die Laufzeit dieses inzwischen abgeschlossenen Projektes betrug 20 Monate (von Februar 1995 bis Ende 1996).

eingesetzte Komponenten wie **Connection Control Functions (CCF)** oder **Service Switching Functions (SSF)** sollten mit sogenannten **Adaptation Units** gekapselt werden, um den Umstellungsaufwand gering zu halten. Es ist daher davon auszugehen, daß die IN-Switching-Funktionalität mittelfristig beibehalten wird. Dieses Vorgehen ist in Analogie zu den in Abschnitt 4.3 behandelten *multiarchitekturellen Agenten* zu sehen.

Das Problem der Interoperabilität zwischen bestehenden Intelligenten Netzen und neuen, TINA-konformen Netzen wird durch die Einführung von sogenannten **Interworking Units** gelöst, die den in Abschnitt 4.4 vorgestellten *Gateway-basierten* Integrationsansatz realisieren. P508 beschränkt sich auf die Ebene der SCFs, da hier das Hauptaugenmerk auf der Interaktion von Diensten liegt. Wichtig ist hierbei unter anderem, daß die Dienste von ROSE[17] auf vergleichbare CORBA-Dienste abgebildet werden. In beiden Fällen ist es erforderlich, SS.7[18]-Protokolldateneinheiten in entsprechende CORBA-Objektaufrufe umzusetzen. Der Unterschied zwischen beiden Verfahren liegt darin, an welcher Stelle die Umsetzung erfolgt: Im ersten Fall ist es im IN-Vermittlungssystem selbst, im zweiten Fall wird ein Konverter zwischen die kommunizierenden IN- bzw. TINA-konformen Vermittlungssysteme geschaltet. Für eine detaillierte Diskussion dieser beiden Ansätze wird auf Kapitel 4 verwiesen.

Bezogen auf die zweite Fragestellung formuliert P508 Anforderungen an CORBA-Plattformen, die für die Telekommunikation geeignet sind. Hier ist an erster Stelle die **Echtzeitfähigkeit** zu nennen, die jedoch gegenwärtig von keinem kommerziellen ORB erreicht wird. Die OMG hat nicht zuletzt auf der Grundlage der in diesem Projekt identifizierten Anforderungen mit dem *Control and Management of Audio/Video Streams RFP* eine Architektur verabschiedet, die CORBA um Echtzeitfähigkeiten erweitert [OMGTC 98].

Eine weitere in der Telekommunikation besonders wichtige Anforderung ist die softwaretechnisch (durch Replikationsdienste) implementierte Hochverfügbarkeit sowie die hardwareseitig (durch redundante Komponenten) realisierte Fehlertoleranz.

Grundsätzlich ist dafür Sorge zu tragen, daß das CORBA-System überhaupt auf einem SS.7-Protokollstack ablauffähig ist, da das SS.7-basierte **Kernel Transport Network**, über das sämtliche Kommunikation erfolgt, (zumindest mittelfristig) nicht modifiziert werden soll. Während Vermittlungssysteme über einen SS.7-Protokollstack verfügen, ist für CORBA momentan lediglich die Unterstützung der TCP/IP-Protokollsuite vorgeschrieben. Ein Inter-ORB-Protokoll auf der Grundlage von SS.7 muß daher definiert werden. Für die detaillierte Diskussion, auf welcher Ebene des SS.7-Protokollstacks dies erfolgen sollte sowie für die Definition des SS.7-Inter-ORB-Protokolls (SIOP) sei auf die sehr umfangreiche P508-Projektdokumentation ([P508-D1], [P508-D2]) verwiesen.

Die Migration von IN zu TINA betrifft natürlich auch die angebotenen (Nutz-)

[17] **R**emote **O**perations **S**ervice **E**lement; ein Dienst der OSI-Anwendungsschicht zum Aufruf von Operationen auf nicht-lokalen Partnerinstanzen. Insbesondere CMIS macht von ROSE Gebrauch.

[18] Signalling System No. 7 ist ein standardisiertes [CCITT Q.700], paketvermittelndes, verbindungsloses outband-Signalisierungsprotokoll, das dem Austausch von Verbindungsauf- bzw. -abbauinformationen zwischen Service Control Functions (SCF) dient.

Dienste, deren Schnittstellen zum gegenwärtigen Zeitpunkt in ASN.1 vorliegen. Um die Dienste in einer CORBA-Umgebung nutzen zu können, müssen diese ASN.1-Schnittstellenbeschreibungen in IDL überführt werden. Die Entwicklung von Compilern zur Umwandlung von ASN.1-Konstrukten in OMG IDL ist daher der nächste Schritt auf dem Migrationspfad zu TINA[19].

Bewertung

Zusammenfassend ist anzumerken, daß das Eurescom-Projekt P508 ausschließlich die Verwendung von CORBA für die Bereitstellung der Nutzfunktionalität von Telekommunikationsnetzen behandelt, also die Definition einer CORBA-basierten Kommunikations- und Diensteinfrastruktur für die Erbringung von Telekommunikationsdienstleistungen zum Ziel hat. Die Fragen des Managements solcher Dienste und auf das TINA-Gesamtsystem abzielende Managementaspekte werden nicht betrachtet. Nichtsdestoweniger haben zahlreiche Aspekte der Migration von IN zu TINA hohe Relevanz für die vorliegende Arbeit:

- Die grundsätzliche Fragestellung, wie heutige Intelligente Netze TINA-Konformität erhalten, liefert neben den in Kapitel 2 aufgeführten Szenarien eine weitere praxisnahe Begründung für die Notwendigkeit einer intensiven Auseinandersetzung mit der Interoperabilitätsproblematik von Managementarchitekturen.
- Die Verfahren, die die Interoperabilität von IN und TINA gewährleisten, d.h. die Spezifikationscompiler und Protokoll-Gateways sind identisch zu denen in Kapitel 4.
- Die ebenfalls in Kapitel 4 ausführlich behandelten Integrationsansätze sind vollständig auf das hier vorgestellte Szenario anwendbar.
- Ferner sind die nicht-funktionalen Anforderungen der Telekommunikation an CORBA-Umgebungen (wie z.B. Effizienz, Robustheit, Skalierbarkeit) vollständig auf die in dieser Arbeit behandelten Einsatzszenarien übertragbar.

3.3 Beispiele kommerzieller Lösungen und Werkzeuge

Auf Seiten der Hersteller von Managementsoftware ist die Notwendigkeit des integrierten Enterprise Managements seit geraumer Zeit erkannt worden. Daher ist es nicht verwunderlich, daß sich nun in Produktankündigungen und Pressemitteilungen zahlreicher Hersteller Aussagen über die Verfügbarkeit integrierter Enterprise Management Systeme

[19] Vgl. hierzu die Ausführungen in Abschnitt 4.4.3 sowie in Anhang B.

finden. Der folgende Abschnitt geht daher auf einige kommerzielle Systeme genauer ein. Die Funktionalität dieser Werkzeuge wird kurz beschrieben, analysiert und hinsichtlich ihrer Verwendbarkeit für die in dieser Arbeit behandelte Fragestellung bewertet. Besonderes Augenmerk wird auf die Erfüllung der im vorigen Kapitel formulierten Anforderungen an Enterprise Management Systeme gelegt.

3.3.1 IBM Tivoli TME 10

Das **Tivoli Management Environment 10 (TME 10)** von IBM ist für die vorliegende Arbeit deswegen interessant, weil es herstellerseitig als Enterprise Management Framework positioniert ist, das unter anderem den geregelten Betrieb unterschiedlicher Plattformen sicherstellen soll. Während die Integration der bisher eigenständigen Softwarepakete für unterschiedlichste Managementaspekte (Softwareverteilung, Netzmanagementplattformen, Trouble-Ticket-Systeme, Ereigniskorrelation, Backup und Restore) zweifellos positiv ist, so stellt sich bei einer genauen Betrachtung heraus, daß sich die Integration dieser heterogenen Systeme häufig nur auf die Bedienoberflächen beschränkt. Die wünschenswerte Integration auf Ebene der Daten ist bisher noch nicht geschehen. Wir werden uns daher im folgenden auf zwei Tivoli-Produkte beschränken, die in der Tivoli-Strategie eine zentrale Rolle für das Enterprise Management spielen.

IBM NetView for AIX

Ein in die Tivoli-Produktlinie integriertes Werkzeug, das in seiner gegenwärtigen Funktionalität erste Ansätze verteilten Managements bietet, ist *IBM NetView for AIX* in der uns vorliegenden Version 4.1. Ursprünglich aus der Lizenzierung von Hewlett-Packard's Netzmanagementsoftware *OpenView* hervorgegangen, ist dieses SNMP-basierte Produkt seitdem von IBM in Eigenregie weiterentwickelt worden. Es verfügt nun hinsichtlich der dieser Arbeit zugrundeliegenden Fragestellung über interessante Eigenschaften, die über den Funktionsumfang bisheriger isolierter SNMP-Managementplattformen hinausgehen und nachfolgend kurz skizziert sind. Eine ausführliche Gegenüberstellung von IBM NetView und HP OpenView anhand eines am Lehrstuhl neu entwickelten und in der Praxis eingesetzten Kriterienkataloges (siehe Abschnitt 2.4) findet der an Details interessierte Leser in [Domb 97].

Ein Anzeichen für die Positionierung von NetView als kooperatives Managementsystem ist das Vorhandensein einer sogenannten *Manager Overtake* Funktion, die es einem Managementsystem ermöglicht, die Aufgaben eines ausgefallenen Partnersystems automatisch zu übernehmen. Damit ein Managementsystem vom Ausfall eines Partnersystems Kenntnis erhält, ist es notwendig, den Status derjenigen Prozesse zu überwachen, aus denen das Partnersystem besteht, bzw. den für das Partnersystem verfügbaren Speicherplatz im Dateisystem zu ermitteln. Die Entscheidung, ursprünglich vom Partnersystem überwachte Ressourcen nunmehr selbst zu steuern, wird anhand einer kritischen Menge ausgefallener Prozesse bzw. der Unterschreitung einer Mindestmenge an Speicherplatz getroffen.

Die hierzu nötigen Informationen liefert die sogenannte *NetView6000 MIB*, welche die oben angesprochenen Prozeßstatus- bzw. Speicherplatz-Informationen in insgesamt zwei Tabellen enthält.

Es ist unmittelbar einsichtig, daß die in dieser MIB enthaltenen zwei Tabellen nur einen sehr rudimentären Schritt in die Richtung eines Managements verteilter kooperativer Managementsysteme darstellen. Außerdem sind sämtliche MIB-Variablen als „read-only“ klassifiziert; Eingriffsmöglichkeiten im Sinne (pro-)aktiven Managements sind bisher nicht vorhanden. Auch hier zeigt sich, daß die Definition geeigneter MIBs für Managementsysteme bislang noch nicht erfolgt ist. Kapitel 5 zeigt auf, welche Management**information** erforderlich ist, um das Management solcher Systeme zu gewährleisten.

Die Notwendigkeit eines weiteren Problembereichs, nämlich die Bereitstellung geeigneter Management**funktionalität** wird deutlich bei der Betrachtung des NetView Polling-Algorithmus, der in Abbildung 3.7 schematisch dargestellt ist.

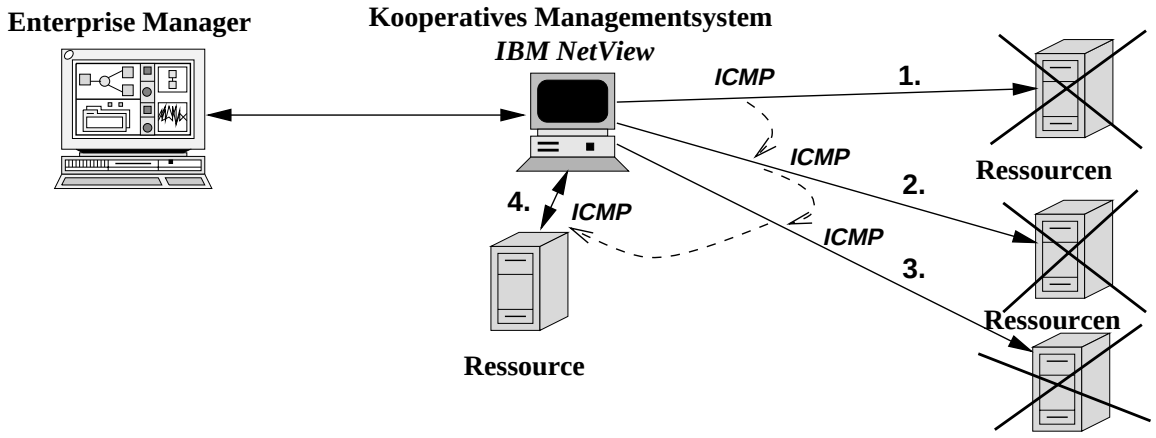


Abbildung 3.7: Verhalten des IBM NetView Polling-Algorithmus

Ziel eines Polling-Algorithmus ist, festzustellen, welche überwachten Ressourcen momentan verfügbar sind, um diejenigen Icons, die diese Ressourcen repräsentieren, in der Topologie-Darstellung der Managementplattform geeignet einzufärben. Hierdurch wird ein Netzadministrator in die Lage versetzt, auf einen Blick zu erkennen, welche Ressourcen momentan in Betrieb sind, bzw. für welche Ressourcen Maßnahmen zur Fehlerdiagnose eingeleitet werden müssen. Die Ermittlung der Verfügbarkeit von Ressourcen[20] geschieht durch das Versenden von ICMP echo-Paketen, die in geeigneten Zeitabständen (i.d.R. alle 5 Minuten) zu den Ressourcen gesandt werden.

[20] Die Gleichsetzung der Verfügbarkeit von Ressourcen mit dem Bestehen von IP-Konnektivität (ICMP ist ein Protokoll der Schicht 3) ist charakteristisch für das *Netz*management, das sich mit Komponenten der unteren 3 Schichten eines Kommunikationssystems befaßt. Diese Annahme ist unzulässig für das System- und Anwendungsmanagement, da sich beispielsweise das Nichtvorhandensein eines Dienstes erst auf Schicht 7 bemerkbar macht. NetView und IBM Systems Monitor (s.u.) nutzen daher andere Mechanismen zur Feststellung von Verfügbarkeit.

Der bei NetView verwendete Polling-Algorithmus für eine Ressource lautet in Pseudocode wie folgt:

```
pollingtimeout := 10 Sekunden;
if (pollingtimeout < 150 Sekunden) then
        poll( Ressource )
        if (antwort_erhalten) then system_inoperativ := false
                else pollingtimeout := 2 * pollingtimeout
        endif;
else system_inoperativ := true
endif;
```

Der Algorithmus basiert auf der Prämisse, daß der Zustand einer Ressource nur dann auf *inoperative* (ohne Funktion, fehlerhaft) geändert werden sollte, wenn nachweislich mehrere Versuche gescheitert sind, zur Ressource Kontakt aufzunehmen. Damit soll verhindert werden, daß eine Ressource, die sich zum Polling-Zeitpunkt gerade in der Initialisierungsphase befindet, als fehlerhaft markiert wird. Ferner wird nach jedem erfolglosen Versuch der Timeout-Wert (beginnend mit 10 Sekunden) für das Polling verdoppelt, um sicherzugehen, daß auch im Falle extrem hoher Round-Trip-Delays aktive Ressourcen auch als solche erkannt werden. Für die maximale Anzahl von Versuchen innerhalb eines Pollingzyklus hat sich in der Praxis ein Wert von „4“ bewährt; dies impliziert, daß eine nicht erreichbare Ressource nach frühestens 150 Sekunden den Status „inoperativ“ zugewiesen bekommt. Um zu verhindern, daß während dieser Zeit keine weiteren Systeme befragt werden können, ist die maximale Anzahl der simultan ausstehenden ICMP-Antworten auf den Wert 3 festgelegt.

Während ein solcher Algorithmus beim Ausfall einzelner Komponenten einen tolerierbaren Kompromiß zwischen Netzbelastung und Plattformaktivität darstellt, ist es unmittelbar einleuchtend, daß ein solcher Algorithmus sehr schlecht skaliert, wenn ein Teil eines großen Rechnernetzes, beispielsweise bereits durch den Ausfall einer zentralen Komponente (z.B. Router), vom restlichen Kommunikationssystem getrennt wird. Bereits beim Ausfall von 3 Ressourcen sind sämtliche Pollingaktivitäten des Managementsystems für die Dauer von 2,5 Minuten (150 Sekunden) blockiert. Wünschenswert ist es daher, aus mehreren Polling-Algorithmen denjenigen auswählen zu können, der am geeignetsten für die spezifische Topologie eines Rechnernetzes erscheint. Obwohl in jüngster Zeit mehrere intelligente Polling-Algorithmen entwickelt wurden, können diese in herkömmlichen Managementplattformen nicht eingesetzt werden, da hier keinerlei Möglichkeiten bestehen, solche Algorithmen zur Laufzeit einzubinden. Kapitel 6 stellt daher einen Ansatz vor, der unter anderem die dynamische Auswahl solcher Algorithmen gestattet.

IBM Systems Monitor / Mid-Level Manager

Unter dem Namen **Systems Monitor** [IBMSysMon94] brachte IBM in der ersten Hälfte der neunziger Jahre ein Produkt auf den Markt, das einen ersten Schritt auf dem Wege zu

verteiltem Management großer Rechnernetze darstellt. Dieses SNMP-basierte Werkzeug besteht aus insgesamt drei Teilkomponenten:

Ein sogenannter **Mid-level Manager (MLM)** ist jeweils für eine ihm zugeordnete Managementdomäne zuständig. Ziel ist hierbei, das zentrale Managementsystem (Top-level Manager, hier: *NetView for AIX)* zu entlasten, da die Rohdaten von Ressourcen bereits an relativ niedriger Stelle der so realisierten Managementhierarchie vorverarbeitet und verdichtet werden. Dies bezieht sich sowohl auf Statusabfragen als auch auf die Sammlung von Daten und die Ereignisfilterung. Zwischen Top-level Manager und den jeweiligen Mid-Level Managern geschieht die Kommunikation größtenteils[21] ereignisgesteuert[22], um den Kommunikationsaufwand zwischen diesen maßgeblichen Systemen auch im Fehlerfalle in Grenzen zu halten. Eine wesentliche Beschränkung des SNMP-Managements in bezug auf die Skalierbarkeit in großen Netzen wird auf diese Art aufgehoben. Mid-level Manager fragen ihrerseits die bei den Ressourcen anfallenden Daten in konfigurierbaren Zeitzyklen per ICMP ab (Polling). Dies ist insofern vertretbar, als ein Mid-level Manager in der Regel nur jeweils für eine eingeschränkte Zahl an Ressourcen (die IBM-Empfehlung liegt bei 50–100 Endsystemen pro MLM [IBMSVSz95]) im LAN-Bereich eingesetzt wird.

Ein Spezifikum der MLM-Architektur liegt in der Fähigkeit, die Konfiguration des MLM von einem Top-level Manager aus vorzunehmen. Der MLM verfügt zu diesem Zweck über einen SNMP-Agenten und eine sogenannte **Mid-level Manager MIB**, die insgesamt sieben Tabellen umfaßt: Die *Alias*-Tabelle gestattet, mehreren Systemen einen Aliasnamen zuzuweisen, um diese zu gruppieren. Aufgabe der *Analysis-Table* ist es, arithmetische Operationen auf mehreren MIB-Variablen auszuführen und das Ergebnis in Form einer neuen MIB-Variable bereitzustellen. Die *Threshold and Collection Table* ermöglicht die Definition von Schwellwerten für bestimmte MIB-Variablen und das Versenden asynchroner Ereignismeldungen an in der *Trap Destination Table* definierte Top-level Manager, sobald die Schwellwerte erreicht werden. Für die Weiterleitung von Traps ist die *Filter Table* verantwortlich: Sie definiert, welche von den Ressourcen empfangenen Traps zu welchem Zeitpunkt an einen oder mehrere Top-level Manager weitergeleitet werden. Eng damit verbunden ist die *Trap Reception Table*, die festlegt, was mit Traps geschehen soll, für die kein Eintrag in der Filter-Tabelle vorhanden ist. Die *Data Collection Table* schließlich gibt Aufschluß darüber, in welche Datei Log-Informationen geschrieben werden und wie groß diese maximal sein darf. Es ist offensichtlich, daß die drei Tabellen zur Schwellwertdefinition, Ereignisfilterung und -weiterleitung nahezu identisch zu der in Abschnitt 3.1.3 beschriebenen *Manager-to-Manager MIB* sind.

Bei der zweiten Komponente des IBM Systems Monitor handelt es sich um den sogenannten **System Information Agent (SIA)**, dessen Aufgabe darin besteht, für das

[21] Um einen Ausfall des MLM-Dämons *midmand* zu registrieren, verwendet der Top-level-Manager Polling mit SNMP GET-Anfragen auf MLM-Variablen. Das üblicherweise für Status-Polling verwendete ICMP würde lediglich die Nichtverfügbarkeit des Gesamtsystems erfassen, jedoch nicht den Ausfall des MLM-Prozesses.

[22] SNMP-Traps werden hier aufgrund der Bedeutung der Daten über TCP, also verbindungsorientiert, an das Managementsystem übermittelt.

Systemmanagement relevante Informationen über Endbenutzersysteme (primär UNIX-Workstations) bereitzustellen. Die über 600 Variablen umfassende MIB des SIA baut auf der von der IETF standardisierten Host Resources MIB [rfc1514] auf und ergänzt diese um zahlreiche weitere Parameter. Ein weiteres Merkmal besteht in der (statischen) Erweiterung der SIA-Managementfunktionalität durch Hinzufügen benutzerdefinierter Anweisungen in die sogenannte *Command Table*, deren Zeilen sich jeweils aus einem oder mehreren miteinander verknüpften UNIX-Kommandos und zahlreichen weiteren Parametern zusammensetzen. Der Aufgabenumfang des SIA kann somit individuell um Überwachungsfunktionalität für weitere Ressourcen vergrößert werden. Es sei an dieser Stelle erwähnt, daß es sich bei SIA um ein reines Monitoring-Werkzeug handelt; aktive Eingriffe in den Betrieb eines überwachten Systems sind lediglich auf Umwegen durch den Administrator möglich.

Die dritte Teilkomponente ist das sogenannte **End User Interface (EUI)**, eine graphische Benutzeroberfläche, die der Konfiguration der beiden vorgenannten Komponenten dient. Hierunter fallen elementare Aufgaben wie das Laden neuer MIBs und Trap-Definitionen sowie das Festlegen von Parametern (wie z.B. Pollingintervalle oder Timeout- und Retry-Werte). Die oben angesprochenen Erweiterungen des SIA werden ebenfalls über diese Schnittstelle eingegeben.

Bewertung

Zusammenfassend handelt es sich beim *IBM Systems Monitor* um eines der ersten Produkte für verteiltes hierarchisches Management, dem ein großer kommerzieller Erfolg beschieden war und in großen Netzen mit einem hohen Anteil an SNMP-überwachten Systemen häufig eingesetzt wird. Ein Anzeichen dafür ist auch, daß sich der ursprüngliche Produktname „Mid-level Manager“ in der Fachwelt mittlerweile als Bezeichnung für ein System zur lokalen Vorverarbeitung von Managementdaten in hierarchischen Managementsystemen nachhaltig etabliert hat (siehe auch die Abschnitte 2.1.2 und 3.1.3) und daher auch in dieser Arbeit verwendet wird. Ein weiterer wichtiger Pluspunkt besteht darin, einige Konfigurationsattribute des Mid-level-Managers per SNMP abzufragen bzw. setzen zu können. Die Notwendigkeit, das Managementsystem selbst überwachen und steuern zu können, wurde hier erkannt und hat zu einem erfolgreichen Produkt geführt, auch wenn die Anzahl der verfügbaren MIB-Variablen eine umfassende Konfiguration nach den Gesichtspunkten des Enterprise Managements noch nicht zuläßt. Insbesondere ist ihnen die Verwandtschaft zu den Konfigurationsoptionen von IBM NetView deutlich anzusehen; Terminologie und Dienstanzahl tragen die Züge des NetView-Managementmodells. Auch ist der volle Funktionsumfang (z.B. Manager Overtake) des Systems Monitor wohl nur zu nutzen, wenn NetView als Top-level Manager vorhanden ist. Die enge Verzahnung des Systems Monitor mit einem Managementsystem des gleichen Herstellers ist zwar aus Vermarktungssicht unmittelbar einsichtig; für ein integriertes und damit herstellerübergreifendes Enterprise Management ist jedoch der Funktionsumfang der MLM-MIB nicht ausreichend.

Der Einsatz der MLM-Komponente mindert einige Nachteile des SNMP-Managements in bezug auf die Skalierbarkeit für große Umgebungen; konzeptionelle Schwächen des SNMP-Managements wie die komplizierte Verwaltung dynamischer Tabellen (hier: die *Analysis*-, *Command*-, und *Alias*-Tabellen) lassen die Konfiguration des Gesamtsystems jedoch zu einer anspruchsvollen Aufgabe werden, die dem Administrator tiefgreifende SNMP-Kenntnisse abverlangt. Desweiteren ist beiden MIBs des Systems Monitor (SIA-MIB und MLM-MIB) klar anzusehen, daß sie ausschließlich auf das Monitoring von UNIX-Systemen ausgerichtet sind. Aufgrund der in Abschnitt 3.1.3 vorgenommenen Bewertung von SNMP erscheint die Eignung des IBM Systems Monitor für die vorliegende Arbeit fraglich, da zahlreiche Merkmale dieses Produkts in unserem Fall bereits durch die Wahl einer entsprechend leistungsfähigen Managementarchitektur erreicht werden können.

3.3.2 Cabletron SPECTRUM Enterprise Management

Seit 1990 ist *Cabletron Systems, Inc.* mit seiner Netzmanagementsoftware SPECTRUM auf dem Markt präsent, die in jüngster Zeit unter der Bezeichnung „SPECTRUM Enterprise Management“ angeboten wird.

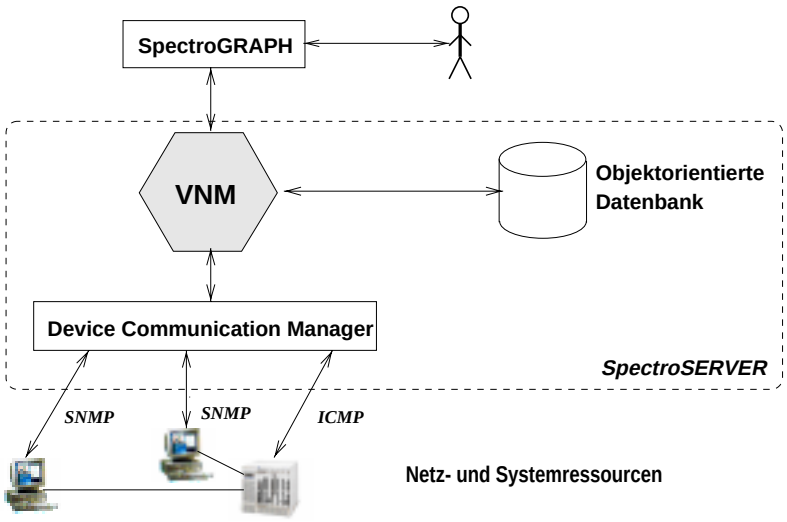


Abbildung 3.8: Aufbau der Managementplattform SPECTRUM

SPECTRUM setzt sich aus folgenden Bestandteilen zusammen (siehe Abbildung 3.8):

- **Virtual Network Machine (VNM)**: Sie ist der zentrale Teil von SPECTRUM, der neben dem Netzmodell auch das gesamte Managementwissen in Form sog. *Inference Handler* enthält. Innerhalb der Virtual Network Machine laufen zahlreiche Threads ab, die zum überwiegenden Teil Kontrollfunktionen (Polling von Attributen, Entgegennehmen von Ereignismeldungen) wahrnehmen und Aktionen durch das Triggern

von Inference Handlern anstoßen können. Die Ausführung eines Inference Handlers erfolgt dann jeweils innerhalb eines Threads.

- **Objektorientierte Datenbank**: In der Datenbank werden alle für das Managementsystem relevanten Informationen abgelegt: Diese umfassen Objektklassen, Instanzen, Attribute, Relationen, Regeln und Assoziationen. Der Zugriff auf die Datenbank erfolgt für den Benutzer transparent ausschließlich durch die VNM, die bei ihrer Initialisierung den gesamten Datenbankinhalt von der Platte in den Hauptspeicher lädt.

- **Device Communication Manager (DCM)**: Da SPECTRUM verschiedene Managementprotokolle unterstützt, entspricht der DCM der VNM-Schnittstelle zu den verwendeten Managementprotokollen. Aufgabe des DCM ist es beispielsweise, von den Agenten kommende SNMP-Traps in Meldungen an die VNM umzusetzen.

- **SpectroGRAPH**: Diese Anwendung bildet die graphische Benutzerschnittstelle zu SpectroSERVER. Sie ermöglicht verschiedene Sichtweisen (geographische, topologische, organisatorische) auf ein Rechnernetz und dessen Elemente. Informationen werden durch graphische Symbole visualisiert; der aktuelle Status einer Netzkomponente wird durch die Farbe des zugehörigen Icons angezeigt.

Die Gesamtheit von VNM, DCM und Datenbank wird als **SpectroSERVER** bezeichnet. SpectroGRAPH steht mit SpectroSERVER in einem *Client/Server*-Verhältnis; es ist daher möglich, mehrere SpectroGRAPHen an einem SpectroSERVER zu betreiben. Ein Benutzer kann mit SpectroGRAPH unter anderem

- neue Modelle erzeugen,

- Assoziationen zwischen Modellen herstellen (Das Verbinden von zwei Komponentenicons generiert in der VNM eine CONNECTS_TO Beziehung zwischen den Modellen) sowie

- die Benutzer von SPECTRUM verwalten.

also Modifikationen an MO-*Instanzen* vornehmen.

Um neues deklaratives Wissen (Objektklassen, Attribute, Relationen oder Regeln) in SPECTRUM einzubringen, sind Eingriffe in den SpectroSERVER nötig, die mit dem **Model Type Editor** (MTE), einem graphischen Werkzeug zum Erzeugen und Löschen von Modelltypen, Relationen und Regeln, vorgenommen werden. Das Hinzufügen zusätzlichen prozeduralen Wissens durch Inference Handler hingegen bedeutet eine Erweiterung der VNM und kann weder mit SpectroGRAPH noch mit dem Model Type Editor, sondern nur durch Neucompilierung der VNM mit den Erweiterungen in Form von C++ Quelltext erfolgen. Eine detaillierte Beschreibung des Datenmodells von SPECTRUM sowie ein darauf aufbauender Prototyp zum Management eines Filetransfer-Dienstes findet der interessierte Leser in [Kell 93].

Der zentrale Gedanke des SPECTRUM Enterprise Managements auf Basis des „Distributed SpectroSERVER“ besteht nun darin, jeder Domäne eines Rechnernetzes einen SpectroSERVER zuzuweisen, der für das Management sämtlicher Ressourcen seiner Domäne verantwortlich ist. Domänen (in der SPECTRUM-Terminologie als „Landscapes“ bezeichnet) werden an der Benutzeroberfläche SpectroGRAPH als Icons dargestellt, die bei Auswahl durch den Benutzer eine Verbindung über ein proprietäres Kommunikationsprotokoll zu dem für diese Domäne zuständigen SpectroSERVER aufbauen. Dieser liefert dann die Topologie-Darstellungen der Ressourcen in Form hierarchisch aufgebauter Maps sowie die für das Management relevanten Ressourcen-Informationen. Parallel zueinander können mehrere Domänen bearbeitet werden, d.h. es bestehen dann Verbindungen zu zahlreichen SpectroSERVERn. Das Gesamtsystem besteht also aus mehreren eigenständigen Managementplattformen, die unter einer einheitlichen graphischen Benutzerschnittstelle zusammengefaßt sind. Eine Hierarchisierung verteilter SpectroSERVER kann mit den gegenwärtig verfügbaren Mitteln nicht erreicht werden; dies ist problematisch, da beispielsweise die Filterung von Ereignissen für jeden SpectroSERVER einzeln definiert werden muß. Bei einer großen Anzahl an Domänen impliziert dies einen sehr hohen Konfigurationsaufwand.

Hinsichtlich der Ausfallsicherheit der SpectroSERVER ist anzumerken, daß es möglich ist, die Domänen-Datenbanken auf mehrere SpectroSERVER zu replizieren und Parameter zur Synchronisation der Datenbankinhalte sowie zu den Zeitbedingungen eines Ausfalls anzugeben. Dies alles geschieht jedoch von Hand durch den Netzadministrator, da es bisher keine Instrumentierung gibt, die eine diesbezügliche Konfiguration der SpectroSERVER gestattet. Hier zeigen sich die Nachteile der Nutzung eines proprietären Kommunikationsprotokolls zur Kommunikation mit den SpectroSERVERn: Es ist bislang nicht machbar, beispielsweise SNMP-Agenten zu entwickeln, die diese Funktionalität bieten. Gegenwärtig ebenfalls nicht möglich ist die Kooperation mehrerer verteilter SpectroSERVER: Dies äußert sich unter anderem darin, daß Ressourcen, die einzelne Domänen miteinander verbinden, durch den Netzadministrator jeweils in die entsprechenden Domänen eingefügt werden müssen.

Bewertung

Auch SPECTRUM erbt mit seiner Fixierung auf das Internet-Management die konzeptionellen Nachteile dieser Architektur. Die Kooperation unterschiedlicher SpectroSERVER erfolgt jedoch auch weiterhin mittels eines proprietären Protokolls und macht von der hierfür in SNMPv2 definierten INFORM-PDU keinen Gebrauch. Auch lassen sich die übermittelten Arten von Information nicht durch den Benutzer definieren, so daß eine entsprechende Konfiguration der verteilten SpectroSERVER ebenfalls nicht möglich ist. Somit kommt es auch hier zu einem „Nebeneinander“ der einzelnen *Distributed SpectroSERVER*, die keine gegenseitige Administration zulassen.

3.4 Zusammenfassung: Möglichkeiten und Defizite existierender Ansätze

Wir haben in diesem Kapitel die große Vielfalt an bereits vorhandenen Managementarchitekturen, Forschungsansätzen und Produkten hinsichtlich der im vorigen Kapitel formulierten Anforderungen untersucht und in bezug auf ihre Eignung für die vorliegende Arbeit bewertet. Die in Abbildung 3.9 enthaltene Tabelle faßt die Ergebnisse unserer Untersuchungen überblicksartig zusammen.

Ansatz bzw. Architektur / Anforderung / Kriterium	Basiseigenschaften: Offenheit / Herstellerunabhängigkeit	Reifegrad	Einfachheit für Anwender	Verbreitung	Flexibilität bzgl. Erweiterungen	Modularität	Objektorientiertheit	Transparenz der Verteilung	Zuverlässigkeit	Verteilung / Delegierung von Managementaufgaben	Skalierbarkeit	Betriebs- / Entwicklungsaspekte: Effizienter Datentransfer	Vorhandene MIBs / Objektkataloge	Abbildung auf Programmiersprachen	Entwicklungswerkzeuge	Erlernbarkeit für Entwickler	Generische Dienste: Logging-Mechanismen	Sicherheitsdienste	Dienste für abgeleitete MI	QoS-Überwachungsdienste	Vermittlungsdienste (Trading)	Suchdienste
OSI/TMN	++	++	−−	○	++	++	++	−−	++	+	++	++	++	○	−	−	++	++	++	++	○	++
Internet-Mgmt.	++	++	○	++	+	+	−−	−−	−	−	−−	−−	++	○	−	+	−	−	−	−	−−	−−
OMG CORBA	++	+	+	+	++	++	++	++	+	++	○	○	−−	++	++	+	++	++	−	○	++	++
ODP/TINA	++	+	−	−	++	++	++	++	++	++	○	○	+	++	+	−	++	++	○	○	++	++
ODMA	++	−	−	−−	++	++	++	++	++	++	+	+		++	+	−−	++	++	++	++	++	++
WBEM	−	−−	○	−	+	++	++	−	−	+	−	○	−	−	++	○	−	−−	−−	−−	−−	−−
MbD	+	+	○	−	++	++	−−	+	○	++	++	−−										
GEMA	−	−	○	−−	+	++	++	+	+	++	○	○	−−	−								
MAScOTTE	++	○	○	−	++	++	++	++	+	++	○	○	−−	++								
EURESCOM P508	++	○	○	−	++	++	++	++	+	++	+	○	−−	++								
Tivoli TME10	○	○	−	○	○	+	++	−	○	+	+	−	++	−	○[1]	−	+	+	−	○	−	○
C'tron Spectrum	−	−	+	−	○	+	++	−	○	−	○	−	++	−	+[1]	+	+	−	−	○	−	○

[1] Plattformspezifisch

Legende:

++	Erfüllt die Anforderung umfassend
+	weitgehend
○	teilweise
−	eingeschränkt
−−	unzureichend

Abbildung 3.9: Vergleich der behandelten Ansätze

Die Analyse bestehender Forschungsansätze in Abschnitt 3.2 führt zu dem Ergebnis, daß zwar in verwandten Problembereichen dieser Arbeit bereits Resultate vorliegen, die uns Hilfestellungen bei unserem Lösungskonzept geben können; die eigentliche Fragestellung der vorliegenden Arbeit ist bislang jedoch weder angegangen noch gelöst worden.

Es konnte ebenfalls demonstriert werden, daß die in Kapitel 2 formulierten Anforderungen von gegenwärtig erhältlichen Produkten nur in unzureichender Weise erfüllt werden. Produkte, die einen inhaltlich mit dieser Arbeit verwandten Themenkomplex behandeln, sind von ihrer technischen Ausgestaltung entweder auf einen verhältnismäßig kleinen Bereich ausgelegt oder stützen sich auf Basisarchitekturen ab, deren konzeptionelle Unzulänglichkeiten die Effektivität der Lösungen beeinträchtigen.

Bei der Analyse des Status Quo im Bereich der Architekturen für das Management sowie für verteilte Verarbeitung hat sich gezeigt, daß mit dem OSI/TMN-Management, CORBA und dem RM-ODP leistungsfähige Rahmenwerke für das Enterprise Management vorliegen, auf denen wir unseren Lösungsansatz aufbauen können. Andere Architekturen scheiden aufgrund der in Abschnitt 3.1.8 beschriebenen Mängel aus.

Das ODP-Referenzmodell bildet mit seinen Viewpoint Languages ein solides und umfangreiches Beschreibungsmodell für generische Klassen verteilter Anwendungen, die wir als Basis für unsere Objektmodelle verteilter kooperativer Managementsysteme einsetzen werden. Die Vorteile des Einsatzes von CORBA für das Management liegen insbesondere darin, daß *eine einzige* Architektur sowohl für die Entwicklung als auch für das Management eines verteilten Systems verwendet werden kann. Dies bedeutet, daß nicht nur die Beschreibung sowohl der Nutz- als auch der Managementobjekte des Systems in einer identischen Notation erfolgt, sondern ebenfalls Nutz- und Managementdaten einer Applikation mit demselben Kommunikationsmittel (nämlich dem Object Request Broker) übertragen werden. Management wird so zu einem wichtigen Teilaspekt verteilter Anwendungen. Da zusätzlich Nutz- und Managementdaten identisch modelliert und implementiert werden, kann das gesamte Spektrum verfügbarer Werkzeuge zur Softwareentwicklung genutzt werden.

Ebenfalls wurde deutlich, daß mit dem Vorhandensein einer Vielzahl heterogener Managementarchitekturen Enterprise Management nur dann machbar und aussichtsreich ist, wenn andere Managementarchitekturen in die Enterprise Management Architektur möglichst nahtlos integriert werden können. Gefordert ist also das in Abschnitt 2.1.4 vorgestellte Umbrella Management, welches Übergänge zwischen der ausgewählten Enterprise Management Architektur und anderen Architekturen schafft und somit eine *einheitliche* Sichtweise auf alle Netzkomponenten, Systeme und Anwendungen eines Kommunikationssystems *unabhängig* von ihrer ursprünglichen Architektur gewährleistet. Wir werden uns daher im folgenden Kapitel eingehend mit den Möglichkeiten zur Realisierung eines Umbrella Managements beschäftigen und anhand von uns entwickelter Prototypen Aussagen hinsichtlich der Effizienz und Effektivität der hierbei bestehenden Wahlmöglichkeiten treffen.

KAPITEL 4

Umbrella Management als Basis integrierten Enterprise Managements

Bei der Wahl einer Managementarchitektur sind die Freiheitsgrade eines IV-Betreibers aus verschiedensten Gründen sehr gering: Netz-, System- und Softwarekomponenten werden aufgrund ihrer Nutzfunktionalität beschafft, der Managementaspekt ist hierbei häufig von nachrangiger Bedeutung. Mit der Festlegung auf ein bestimmtes Einsatzumfeld (LAN/MAN-Bereich, Telekommunikation) ist zumeist auch die Managementarchitektur determiniert: Wie bereits in Abschnitt 3.1 ausgeführt wurde, wird im LAN/MAN-Bereich nahezu ausschließlich SNMP als Managementprotokoll eingesetzt, während im Telekommunikationsumfeld OSI/TMN-basiertes Management überwiegt. Mit dem Zusammenwachsen der Daten- und Telekommunikation befinden sich, wie wir u.a. in dem in Abschnitt 2.2.4 beschriebenen Managementszenario festgestellt haben, heutzutage bei Betreibern großer Corporate Networks mehrere Managementsysteme im Einsatz, die jeweils auf unterschiedlichen Managementarchitekturen beruhen.

Die in den Abschnitten 3.1.2, 3.1.4 und 3.4 motivierte Nutzung von CORBA und RM-ODP für das Management verteilter Systeme und Anwendungen bedingt zwangsläufig die Auseinandersetzung mit der Problematik der Koexistenz und Kooperation heterogener Managementarchitekturen: Angesichts der hohen Zahl an bereits existierenden Internet- und OSI-konformen MIBs und Objektkatalogen ist es bei der Einführung eines auf einer neuen Architektur basierenden Managementkonzepts eine *conditio sine qua non*, dafür zu sorgen, daß die bisher installierten Systeme möglichst *ohne Modifikation* innerhalb der neuen Architektur weiterbetrieben werden können. Dies ist, wie in Abschnitt 2.1.4 ausgeführt wurde, der Grundgedanke des Umbrella Managements. Letzteres ist umso wichtiger, als es häufig für bestimmte Klassen von Ressourcen (z.B. einfache Netzkomponenten wie Hubs oder Modems) nicht möglich ist, deren Managementagenten nachträglich zu modifizieren.

Die in diesem Kapitel vorgestellten Konzepte zur Erreichung der Interoperabilität von Managegementarchitekturen sind für das Umbrella Management unverzichtbar und speziell für Managementsysteme als zentrale Bestandteile integrierten Managements von außerordentlich hoher Wichtigkeit. Abschnitt 4.1 definiert das dieser Arbeit zugrundelie-

gende Interoperabilitätsszenario auf der Basis von CORBA als Enterprise Management Architektur und skizziert kurz die prinzipiellen Alternativen zur Erreichung der Interoperabilität.

Die Abschnitte 4.2, 4.3 und 4.4 schildern anhand mehrerer von uns entworfener und implementierter Prototypen die Spezifika der unterschiedlichen Realisierungsalternativen für das Umbrella Management sowie diejenigen Arbeiten, auf denen unser jeweiliges Lösungskonzept aufbaut. Das Kapitel schließt mit einer Bestandsaufnahme in Abschnitt 4.5 und illustriert die jeweils sinnvollen Anwendungsbereiche für die vorgestellten Lösungen.

4.1 Die Notwendigkeit eines Umbrella Managements

Wir haben aufgrund der von uns in Kapitel 3 durchgeführten Analyse existierender Managementarchitekturen CORBA als Enterprise Management Architektur ausgewählt. Diese Architektur bildet somit nicht nur das Rahmenwerk für die in Kapitel 5 vorgestellten Objektmodelle und Managementdienste, sondern agiert insbesondere in der Rolle als „Management-Backbone“, in den andere Managementarchitekturen integriert werden müssen. CORBA agiert somit in der Rolle einer Architektur für das Enterprise Management; die Integration anderer Architekturen erfolgt anhand definierter Übergänge. Abbildung 4.1 stellt dies graphisch dar.

Der naheliegende Weg zur Implementierung einer vollständig und ausschließlich auf CORBA basierenden Managementlösung besteht darin, sowohl das Managementsystem als auch die Agenten vollständig in Form verteilter CORBA-Objekte zu implementieren, die über einen ORB interagieren. Um dies sicherzustellen, müssen mindestens folgende drei Voraussetzungen erfüllt sein:

1. Grundsätzlich muß die Interoperabilität zwischen ORBs gewährleistet sein, d.h. Objekte auf unterschiedlichen Systemen innerhalb des Rechnernetzes müssen in der Lage sein, Daten über die ORBs auszutauschen. Obwohl die bereits im Dezember 1995 verabschiedete CORBA 2.0 Spezifikation dies explizit vorschreibt, ist es bis zum heutigen Tage keinesfalls selbstverständlich, daß zwischen ORB-Produkten unterschiedlicher Hersteller Daten ausgetauscht werden können. Dies bedeutet, daß die Kommunikation zwischen verteilten Komponenten eines Managementsystems derzeit noch nicht in jedem Fall möglich ist.

2. Eine weitere Voraussetzung ist das Vorhandensein von Managementplattformen, die über CORBA-konforme Schnittstellen sowohl für Managementapplikationen als auch zu den Agentensystemen verfügen. Da das Management hohe Anforderungen

an die Sicherheit stellt, bedeutet dies, daß weitreichende Mechanismen für die Autorisierung und Authentifizierung vorhanden sein müssen. Die relativ späte (im zweiten Quartal 1996 erfolgte) Standardisierung des *CORBA Security Service* hat bisher noch zu keinen tragfähigen Ergebnissen in Form kommerziell erhältlicher Produkte geführt. Folglich ist es zwar möglich, daß einzelne Plattformbestandteile wie Kernsystem, Zustandsmonitor oder Topologiemanager über einen (CORBA 1.2 konformen, d.h. lokalen) ORB kommunizieren; der Austausch von Daten mit Agentensystemen verbietet sich jedoch. Ein Beispiel für eine solche Implementierung ist die Produktlinie *IBM Tivoli TME 10*, die in Abschnitt 3.3.1 besprochen wurde.

3. Ferner ist schließlich die Verfügbarkeit einer gewissen Mindestanzahl von Management*diensten* notwendig, die beispielsweise das Erzeugen, Gruppieren und Löschen von Managementobjekten sowie die Überwachung von Schwellwerten gestatten oder die Definition von Managementdomänen sowie die Durchsetzung von Zielvorgaben ermöglichen. Hier liegen weitere Defizite: Obwohl gegenwärtig mehr als 18 verschiedene CORBAservices (siehe dazu auch Anhang A) spezifiziert sind, die allgemein verwendbare Grundfunktionalität bereitstellen, verfügen gegenwärtige CORBA-Entwicklungswerkzeuge lediglich über einen kleinen Teil dieser Dienste: Eine CORBA-Implementierung, die über mehr als ein Viertel dieser Dienste verfügt, gilt gegenwärtig als ausgesprochen fortgeschritten. In der Tat ist es schwierig, ORBs zu finden, deren Dienstumfang über essentiell notwendige Basisdienste hinausgeht. Die in Abschnitt 3.1.2 angesprochenen *Management*dienste von CORBA sind bisher nicht in Form konkreter Implementierungen am Markt erhältlich; im Rahmen der in Kapitel 5 behandelten Dienste für das Management verteilter kooperativer Managementsysteme stellen wir einige Dienste vor, die sich auf das Management verteilter kooperativer Managementsysteme beziehen.

Die Analyse von CORBA in Abschnitt 3.1.2 hat jedoch auch verdeutlicht, daß deren gegenwärtiger Entwicklungsstand aufgrund der Insuffizienzen heutiger Werkzeuge sowie des Mangels an geeigneten Implementierungen standardisierter Managementdienste die Entwicklung ausschließlich auf CORBA basierender Managementlösungen noch nicht gestattet. Mechanismen zur Sicherstellung der Interoperabilität von Managementarchitekturen sind daher auch im Hinblick auf die Wiederverwendung anderweitig standardisierter Dienste für integriertes CORBA-basiertes Enterprise Management erforderlich. Die Nutzung von Diensten einer Managementarchitektur durch Systeme, denen ein anderes Rahmenwerk zugrundeliegt, bietet neben der Sicherung bereits getätigter Investitionen insbesondere die Perspektive, Dienste architekturübergreifend zu vererben. So könnten beispielsweise CORBA-basierte Systeme, die derzeit noch nicht über geeignete Managementdienste verfügen, mit Hilfe von OSI-Managementdiensten administriert werden. Abschnitt 4.4.5 geht auf diesen Punkt detailliert ein.

Mit dem zukünftigen Aufkommen weiterer neuer Architekturen, die für Managementzwecke verwendet werden können (siehe die Analyse in Abschnitt 3.1) wird offensichtlich die Problematik der Koexistenz und Kooperation von Managementsystemen über die

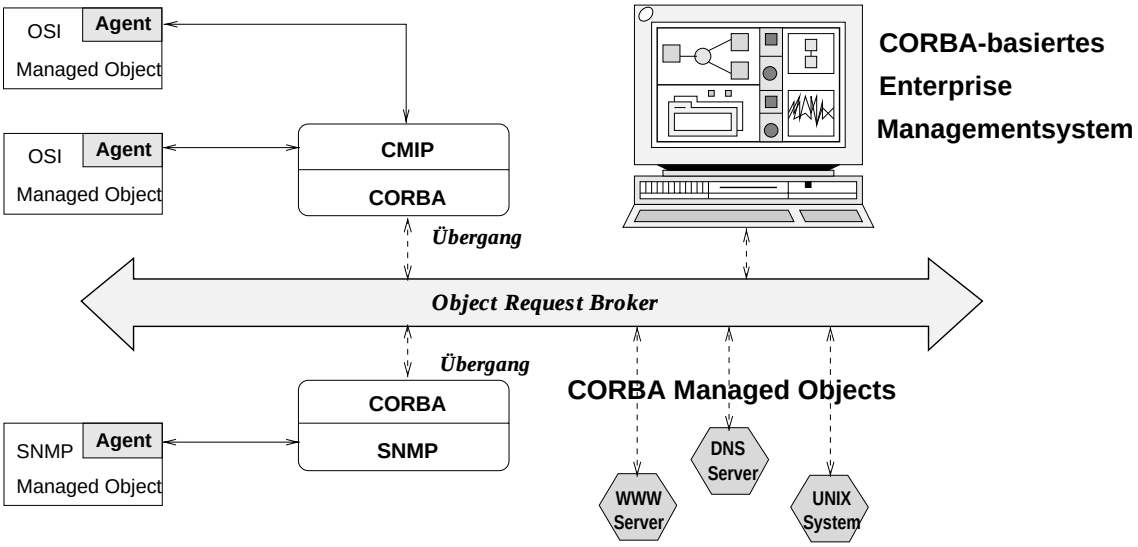


Abbildung 4.1: CORBA als Management-Backbone

Grenzen heterogener Managementarchitekturen hinweg immer weiter zunehmen. Wirklich integriertes Management impliziert also, daß Übergänge geschaffen werden, die eine nahtlose Kombination der Architekturen erlauben. Prinzipiell gibt es hier drei Alternativen, die sich dahingehend unterscheiden, welches System die Abbildung letztendlich vornimmt. Diese sind in Abbildung 4.2 dargestellt (siehe auch [KeNe 97a], [KaSe 94]):

- **Multiarchitektureller Manager**: Die Umsetzung zwischen unterschiedlichen Architekturen erfolgt innerhalb des managenden Systems, das damit mehrere oder alle „Management-Sprachen spricht“. Damit ist auch keine direkte Umsetzung der Protokolle notwendig, sondern jeweils nur eine Abbildung auf das entsprechende Kommunikations-API des Managers. Häufig findet die Übersetzung der Managementinformation allerdings nicht in der Infrastruktur des Managers statt, die gegenwärtig durch sogenannte **Managementplattformen** bereitgestellt wird (vgl. hierzu [HeAN 99]), sondern muß in den darauf aufbauenden Managementanwendungen durchgeführt werden. Hinsichtlich der Unterscheidung zwischen den Begriffen „Manager“ und „Managementplattform“ sei bereits an dieser Stelle angemerkt, daß in Zusammenhang mit Interoperabilitätsuntersuchungen die Rollenbezeichnung „Manager“ häufig als Synonym für ihre Implementierungsausprägung „Managementplattform“ verwendet wird. Wenn in den folgenden Ausführungen die Rolle eines überwachenden und steuernden Systems gemeint ist, werden wir die Begriffswelt des Organisationsmodells verwenden und von einem „multiarchitekturellen Manager“ sprechen; ist hingegen seine technische Realisierung gemeint, verwenden wir den Begriff „multiarchitekturelle (Management-) Plattform“. Abschnitt 4.2 stellt anhand eines von uns entwickelten und implementierten Prototypen die Verfahren zur Entwicklung multiarchitektureller Manager detailliert vor und bewertet diese.

- **Multiarchitektureller Agent**: Die notwendigen Abbildungen erfolgen bereits im Agenten, also im zu administrierenden System oder werden durch entsprechende Vorgaben an dessen Realisierung überflüssig gemacht. Abschnitt 4.3 führt dies genauer aus.
- **Management-Gateway**: Der Übergang wird durch ein Zwischensystem realisiert, das sowohl eine Übersetzung der Managementinformation als auch die Umsetzung der Managementprotokolle vornimmt. Dies geschieht in zwei Schritten: Zuerst müssen die verschiedenen Spezifikationssprachen, in denen die Managementobjekte beschrieben sind, algorithmisch ineinander überführt werden. Im zweiten Schritt müssen zur Laufzeit die Protokolle bzw. deren Elemente ineinander überführt werden. Abschnitt 4.4 beschreibt die hierzu notwendigen Verfahren und unsere prototypischen Implementierungen im Detail.

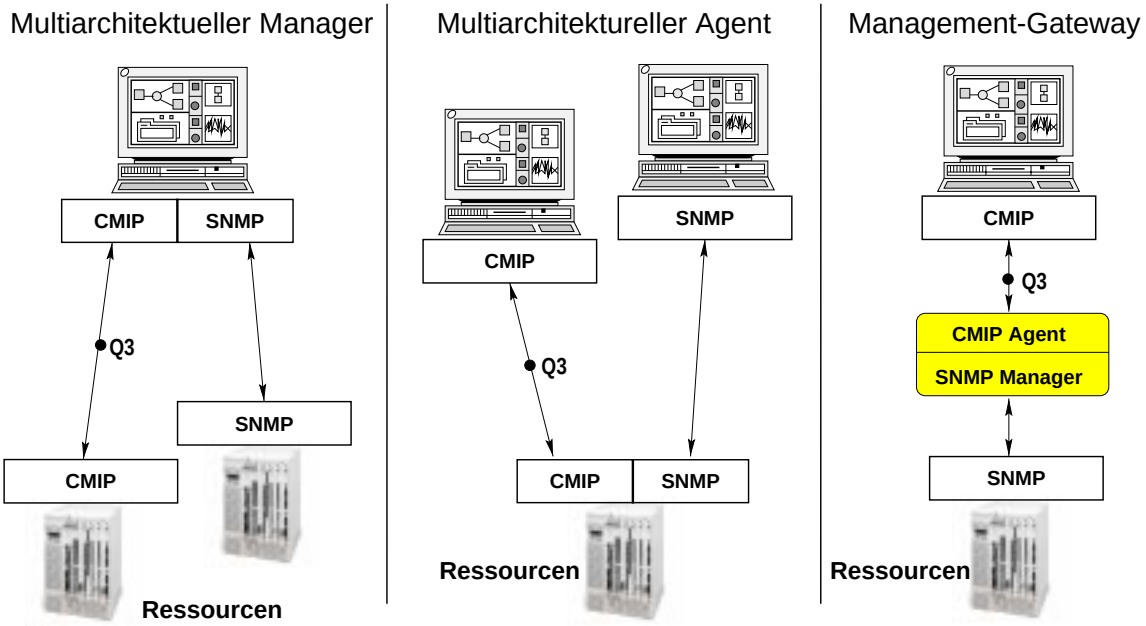


Abbildung 4.2: Übergänge zwischen Managementarchitekturen

Die genannten alternativen Ansätze werden in den folgenden Abschnitten 4.2 bis 4.4 anhand der von uns entworfenen und implementierten Prototypen detailliert vorgestellt. Unsere Implementierungserfahrungen bilden das Fundament, um in Abschnitt 4.5 die drei Realisierungsalternativen für das Umbrella Management gegenüberzustellen und hinsichtlich ihrer Komplexität sowie ihrer praktischen Anwendbarkeit zu bewerten.

4.2 Multiarchitektureller Manager

Wir werden in diesem Abschnitt die erste der drei vorgestellten Integrationsalternativen einer eingehenden Betrachtung unterziehen, um deren grundsätzliche Eignung sowie die sich aus ihrer Verwendung ergebenden Randbedingungen zu identifizieren.

Zunächst betrachten wir die grundlegenden Anforderungen, die sich aus der Erweiterung einer am Markt erhältlichen SNMP-Managementplattform um eine neue Managementarchitektur ergeben, in unserem Fall um CORBA. Dies spiegelt ein praxisnahes Szenario wider, das häufig in großen Corporate Networks gegeben ist: Bisher wird zur Administration des Kommunikationsnetzes das Internet-Management eingesetzt; im Zuge der Migration hin zu CORBA-basiertem Management sollen jedoch die bestehenden SNMP-Managementplattformen vorerst beibehalten werden, um die große Anzahl von SNMP-Agenten auch weiterhin nutzen zu können. Die sukzessive Behandlung der sich aus diesem Sachverhalt ergebenden Anforderungen an einen solchen multiarchitekturellen Manager (siehe dazu Abschnitt 4.2.1) legt ein systematisches Vorgehen fest, das die Aufteilung des Problembereichs in einzelne Arbeitspakete vornimmt.

Die Wahl der kommerziellen Managementplattform IBM NetView als Basis unserer Integration, deren Architektur wir in Abschnitt 4.2.2 kurz vorstellen werden, ist durch folgende Argumente gerechtfertigt: Hinsichtlich seiner Entstehungsgeschichte ist anzumerken, daß NetView ursprünglich von HP OpenView, dem Marktführer für Managementplattformen, abstammt und seitdem von IBM kontinuierlich verbessert wurde. Nichtsdestoweniger weist NetView sowohl hinsichtlich seiner Architektur als auch seiner Programmierschnittstellen auch heute noch eine hohe Ähnlichkeit mit OpenView auf. Somit ist unser Integrationsansatz ohne großen Aufwand auch auf das HP-Produkt anwendbar. Der ausgespochen hohe Marktanteil dieser beiden Systeme unterstreicht ferner die Praxisrelevanz unseres Ansatzes und verdeutlicht, daß es sich bei NetView um einen charakteristischen Vertreter heutiger Managementplattformen handelt. Die von uns entwickelte Lösung stützt sich somit auf eine breite Basis installierter Systeme ab.

Die Behandlung der sich aus der Anforderungsanalyse ergebenden Arbeitspakete wird in den Abschnitten 4.2.3, 4.2.4 und 4.2.5 vorgenommen. Hierbei werden die sich aus den technischen Gegebenheiten ergebenden unterschiedlichen Realisierungsalternativen identifiziert und evaluiert. Abschließend erfolgt eine Bewertung des plattfombasierten Integrationsansatzes in bezug auf seine Eignung für das Enterprise Management.

4.2.1 Anforderungen an multiarchitekturelle Manager

Moderne Managementplattformen verfügen über eine Vielzahl von Komponenten, die spezifische Managementdienste realisieren. Ein wesentliches Designkriterium unseres Ansatzes besteht darin, daß keine Funktionalität neu in CORBA implementiert werden muß, die nicht bereits Bestandteil des Managementsystems ist. Unser Ziel ist somit die Schaffung einer integrierten, plattformgestützten Informationsbasis, in der sämtliche Manage-

mentinformationen auf einheitliche Art gespeichert und verarbeitet werden – und zwar unabhängig davon, aus welcher Managementarchitektur diese Information stammt. Ferner gelten die folgenden Randbedingungen:

- Ressourcen eines verteilten Kommunikationssystems (Anwendungskomponenten, Drucker, Endsysteme etc.) besitzen CORBA-konforme Agenten; als Kommunikationsmittel zwischen der Plattform und den Agenten wird daher ausschließlich CORBA eingesetzt.

- Die Speicherung der Managementinformation über MO-Instanzen sowie deren weitere Verarbeitung geschieht in den Datenbanken des Managementsystems. Zur Interaktion eines Benutzers soll die graphische Benutzerschnittstelle des Managementsystems verwendet werden.

- Alternativ zur Plattform-GUI sollte es möglich sein, CORBA-basierte Managementapplikationen mit Informationen aus den Plattform-Datenbanken zu versorgen, damit diese den Zustand der Managementobjekte und somit auch der Ressourcen überwachen und steuern können. Bei diesen Managementapplikationen kann es sich einerseits um eigenständige (potentiell verteilte) CORBA-Applikationen handeln oder auch um CORBA/Java-fähige WWW-Browser, in die ressourcenspezifische Applets geladen werden.

Abbildung 4.3 stellt die Architektur eines solchen multiarchitekturellen CORBA/SNMP-Managementsystems dar.

Um die oben genannten Zielsetzungen zu erreichen, müssen folgende Voraussetzungen erfüllt sein:

1. Es muß eine Kommunikationsmöglichkeit geschaffen werden, über die das SNMP-basierte Managementsystem möglichst nahtlos auf CORBA-Objekte zugreifen kann.

2. Es muß eine Möglichkeit gefunden werden, wie die grundlegenden, bereits vorhandenen Basisdienste der Plattform für das Management von CORBA-Objekten eingesetzt werden können. Dies wird in Abschnitt 4.2.4 exemplarisch anhand von zwei grundlegenden Plattformdiensten demonstriert.

3. In der Plattform muß eine Informationsbasis vorhanden sein, in der ein Modell des zu überwachenden Kommunikationssystems und der darin enthaltenen Ressourcen gespeichert ist.

Diese Anforderungen legen die Aufteilung der Gesamtproblematik in einzelne Arbeitsschritte fest, die wir nach der Vorstellung der Architektur von IBM Netview in den folgenden Abschnitten im Detail besprechen werden.

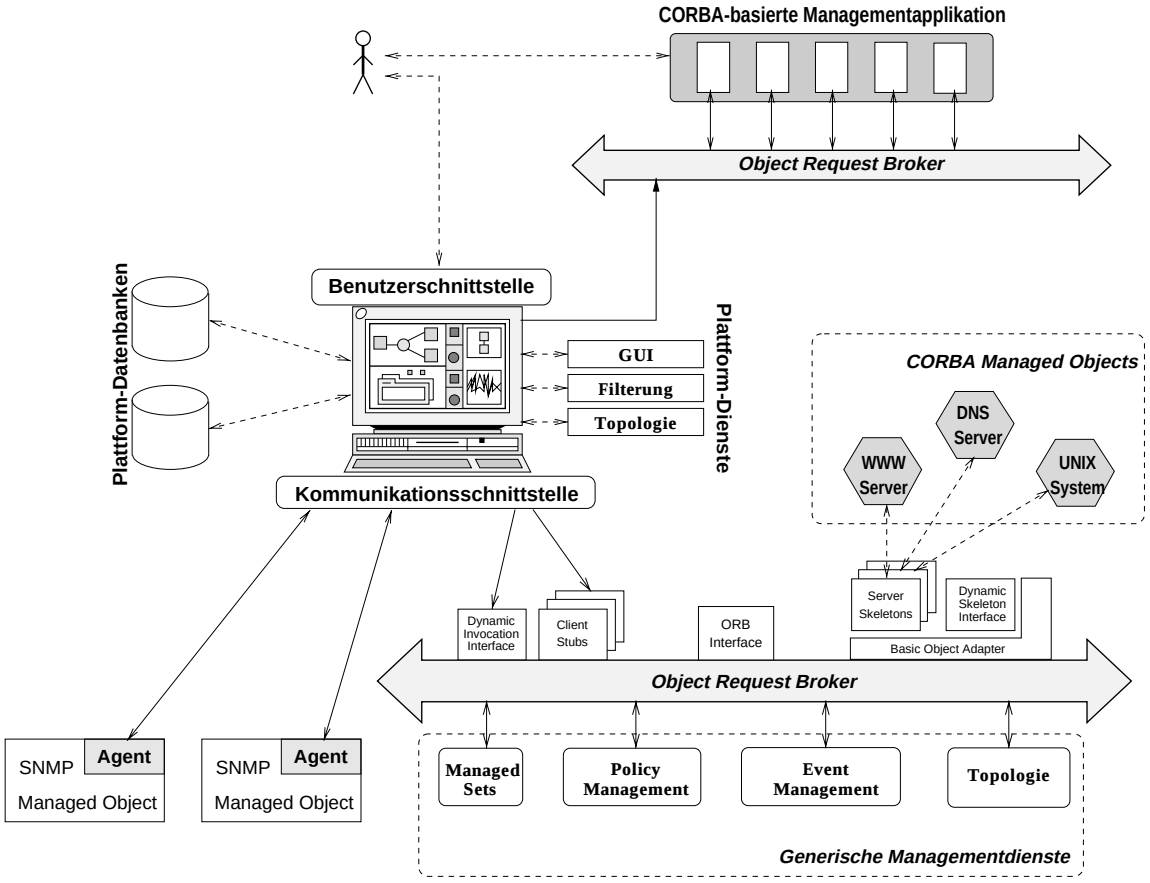


Abbildung 4.3: Architektur eines multiarchitekturellen CORBA/SNMP-Managers

4.2.2 Implementierungsbeispiel: IBM NetView for AIX als multiarchitektureller Manager

Wir haben in Abschnitt 3.3.1 im Rahmen der Vorstellung von IBM Tivoli TME 10 einige Aspekte von IBM NetView vorgestellt, die insbesondere auf seine Verwendung als kooperatives Managementsystem abgestellt haben. Um die technische Grundlage für unsere in den folgenden Abschnitten beschriebene Integration zu legen, werden wir kurz die Architektur dieses Managementsystems unter funktionalen Gesichtspunkten skizzieren. Auch hierfür gilt, daß die von NetView angebotenen Dienste charakteristisch für alle am Markt verfügbaren Managementplattformen sind. Diese Dienste umfassen folgende Bereiche:

- **Discovery und Polling:** Ressourcen eines Kommunikationsnetzes werden zur Laufzeit identifiziert[1] und ihr Zustand in regelmäßigen Abständen abgefragt. Die Ergebnisse werden in der Plattformdatenbank gespeichert, um von anderen Diensten weiterverarbeitet zu werden.

[1] Die Menge der von den Ressourcen unterstützten MIBs ist jedoch, wie bereits vorher ausgeführt wurde, nicht automatisch ermittelbar.

- Einer dieser Dienste, der auf Zustandsinformationen angewiesen ist, ist die **Topologieverwaltung**. Ihre Aufgabe besteht darin, die in der Datenbank gespeicherten Ressourcen auf eine Enthaltenseinshierarchie (sog. **Maps**) abzubilden, welche die hierarchische Netzstruktur geeignet repräsentiert (z.B. Endsysteme sind in Subnetzen enthalten; mehrere Subnetze bilden ein Netz; in einer Domäne sind mehrere Netze enthalten). Diese Enthaltenseinshierarchie wird an der graphischen Benutzerschnittstelle dem Administrator angeboten. Ferner wird der Zustand einer Ressource durch entsprechende farbliche Kennzeichnung seines Symbols an der graphischen Benutzerschnittstelle angezeigt. Für die Topologie von nicht auf dem *Internet Protocol (IP)* basierenden Ressourcen[2] ist der sog. *Generalized Topology Manager (GTM)* zuständig.

- Die Aufgabe der **Ereignisverarbeitung** besteht darin, Ereignismeldungen zu empfangen und nach benutzerdefinierten Kriterien zu filtern, um diese anschließend ihrer Bedeutung entsprechend an der graphischen Benutzeroberfläche darzustellen bzw. an spezifische Managementapplikationen weiterzuleiten. NetView kann sowohl SNMP-Traps als auch CMOT-Notifications verarbeiten. Die *Event Management Services (EMS)* der Plattform gestatten ferner die persistente Speicherung bestimmter Ereignismeldungen. Für SNMP-Traps können weitere Aktionen wie das Ausführen von Skripten zur Automatisierung von Managementaufgaben sowie eine Kategorisierung der Ereignismeldungen erfolgen. Wir werden im weiteren Verlauf ebenfalls von der Möglichkeit Gebrauch machen, neue Arten von Traps definieren zu können.

- Die **Schwellwertüberwachung** erlaubt die Definition von Maximal- bzw. Minimalwerten einzelner MIB-Variablen durch den Benutzer und sendet bei Über- bzw. Unterschreitung dieser Werte eine Ereignismeldung aus.

- Der **MIB-Browser** erlaubt den Zugriff auf den vollen Umfang an Detailinformation zu den Ressourcen, die in den baumartig stukturierten MIBs vorgehalten werden. Hierbei ist jedoch zu beachten, daß MIB-Browser auf das Informationsmodell jeweils einer Managementarchitektur (hier: das Internet-Informationsmodell) zugeschnitten sind; die Nutzung des NetView MIB-Browsers auch für CORBA-Objekte scheidet daher aus.

- Die **Kommunikationskomponente** eines Managementsystems schließlich gestattet die Anbindung unterschiedlicher Managementprotokolle, um die Administration von Ressourcen zu gewährleisten, denen verschiedene Managementarchitekturen zugrundeliegen. Im Fall von IBM NetView besteht diese aus folgenden Komponenten:

  – Der *Postmaster*-Dämon enthält sowohl einen SNMP- als auch einen CMOT-Stack und versetzt das Managementsystem somit in die Lage, Managementinformation sowohl aus OSI- als auch aus SNMP-Agenten zu verwalten.

[2] Dies ist für die uns interessierenden verteilten CORBA-Objekte der Fall, selbst wenn der ORB auf einem IP-Protokollstack aufsetzt und es sich demnach beim Endsystem um eine IP-basierte Ressource handelt.

– Die ORS-Datenbank (*Object Registration Services*) ist ein Verzeichnis von bekannten Agenten und den von ihnen verwalteten Managementobjekten. Hierbei spielt es keine Rolle, aus welcher Managementarchitektur diese stammen.

Wichtig für unseren Integrationsansatz ist die Tatsache, daß zu jedem dieser generischen Plattformdienste eine offengelegte Programmierschnittstelle existiert. Dies wird uns gestatten, einzelne Dienste unmittelbar für unsere Belange zu nutzen.

4.2.3 Anbindung des Managementsystems an einen ORB

Der Object Request Broker ermöglicht den Zugriff auf alle CORBA-Objekte, die bei ihm registriert sind (die Registrierung erfolgt durch Erzeugen einer Objektreferenz). Client-Anwendungen können über die Aufrufschnittstellen des ORBs (statisch oder dynamisch, vgl. hierzu Anhang A) die Methoden der Objekte aufrufen. Ein Managementsystem (das eventuell selbst aus verteilten Objekten besteht) kann also auf jedes Managementobjekt zugreifen, wenn es eine Referenz für dieses Objekt besitzt. Die Informationsquellen, um vorhandene Objektreferenzen zu ermitteln, sind in Anhang A beschrieben. Eine Anwendung, die auf CORBA-Objekte zugreifen soll, kann in jeder Programmiersprache implementiert werden, für die der verwendete ORB eine IDL-Sprachabbildung anbietet. Dabei spielt es, wie bereits oben erwähnt wurde, ebenfalls keine Rolle, in welcher Sprache die Objekte implementiert sind. Somit ist es für den Plattform-basierten Integrationsansatz lediglich notwendig, das Managementsystem als ein oder mehrere CORBA-Objekte erscheinen zu lassen, die über den ORB mit den Agentensystemen kommunizieren.

Sollen Managementsysteme als Basis für die Integration unterschiedlicher Managementarchitekturen verwendet werden, ist es naheliegend, unmittelbar an der Kommunikationsschnittstelle der Plattform anzusetzen, um so die Heterogenität verschiedener Managementprotokolle so weit „unten“ wie möglich abzuhandeln. Der ORB ist so direkt in die Kommunikationsinfrastruktur eines Managementsystems integriert und deren API als „protokollunabhängige“ Programmierschnittstelle für CMIP, SNMP und CORBA verwendet. Das Ziel besteht in der Schaffung einer umfassenden Kommunikationsmöglichkeit zwischen der Plattform und dem ORB. Dies bezieht sich sowohl auf die Anwendung der von der Plattform ausgehenden Operationen auf den Agenten als auch auf die Übermittlung asynchroner Ereignismeldungen von den Agenten an die Plattform. Wir werden im folgenden ausführen, welche Schritte dazu nötig sind.

Alternative 1: Kopplung des ORBs an das XMP-API

Eine Forderung, die an die Kommunikationskomponente von Managementsystemen gestellt wird, besteht darin, eine protokollunabhängige Programmierschnittstelle für den Zugriff auf Managementobjekte zur Verfügung zu stellen. Ein typischer Vertreter dieses Gedankens ist XMP/XOM (X/Open Management Protocol [XMP 92], X/Open OSI-Abstract-Data Manipulation API [XOM 91]), dessen Zielsetzung darin besteht, eine einheitliche

Programmierschnittstelle für die Managementprotokolle CMIP und SNMP bzw. die Darstellung von ASN.1-Datentypen in C bereitzustellen, um einheitliches Management unabhängig von der verwendeten Managementarchitektur zu erlauben. Dieser Versuch der Vereinheitlichung muß jedoch aus heutiger Sicht als gescheitert betrachtet werden, da sich die Heterogenität der Managementprotokolle nicht verbergen ließ: In Abhängigkeit des verwendeten Protokolls müssen zur Generierung der PDUs unterschiedliche Parameter in die XMP-Funktionsaufrufe eingesetzt werden. Letztere werden mit XOM erzeugt, einem API für die Repräsentation von ASN.1-Datentypen in C (siehe auch [PoCh 96]).

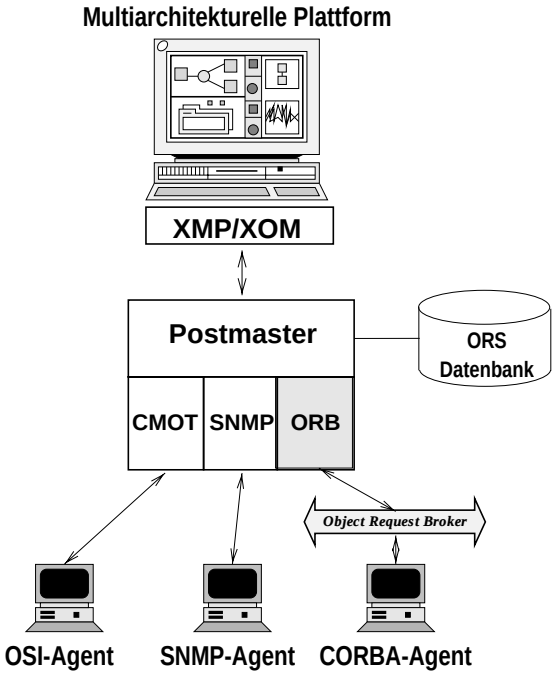


Abbildung 4.4: XOM/XMP-basierte Integration des ORBs in ein Managementsystem

Die Integration des ORBs mit dem XMP-API müßte folgende Schritte beinhalten:

- Das XMP/XOM-API wird verwendet, um Methoden auf CORBA-Objekten aufzurufen. Dazu sind ein *CORBA-Management-Service-Package* und diverse *Management-Content-Packages* nötig, die nach XOM konvertierte IDL-Datentypen beinhalten. Die *Management-Content-Packages* könnten durch einen Compiler generiert werden, der eine IDL-XOM-Sprachabbildung durchführt. Ein solcher Compiler existiert jedoch bisher nicht und müßte erst entwickelt werden.
- Der Postmaster-Dämon wandelt die für CORBA-Objekte bestimmten XMP-Funktionsaufrufe in CORBA-Methodenaufrufe um. Hierfür ist es sinnvoll, die dynamische Aufrufschnittstelle (DII) zu verwenden, weil sonst für jede Objektklasse ein eigener Client Stub benötigt wird, was bei der großen Menge von Managementobjekten zu Leistungseinbußen führen würde.

- Die ORS-Datenbank kann verwendet werden, um Adressierungsinformation für CORBA-Objekte zur Verfügung zu stellen. Da die „Adresse“ eines Objekts durch seine Referenz dargestellt wird, ist hier eine Abbildung zwischen Objektreferenzen und benutzerfreundlichen Namen denkbar. Eine solche Funktionalität wird durch den CORBA Naming Service geboten.

Folgende Argumente sprechen gegen diesen Lösungsansatz:

- Die Entwicklung von Managementanwendungen mit XMP/XOM ist kompliziert und umständlich, da zahlreiche Parameter der ausgesprochen umfangreichen Funktionssignaturen lediglich für CMIP relevant sind und somit für andere Architekturen mit leeren Wertebelegungen aufgefüllt werden müssen, was jedoch recht unübersichtlich ist. Aus diesem Grund wird das API auch für die Kommunikation mit SNMP-Agenten in der Praxis nicht verwendet; vielmehr wird hierfür ein eigenständiges, einfaches SNMP-API genutzt. Der Vorteil, den XMP/XOM bietet – ein gewisses Maß an Transparenz – kann den Nachteil der komplizierten Anwendungsentwicklung nicht ausgleichen. XMP hatte seine Existenzberechtigung bis vor kurzem dadurch, daß es das einzige standardisierte API für CMIP war. Mit der Standardisierung des TMN/C++ APIs ([NMF 98], [CCSH 97]) entfällt nunmehr diese Notwendigkeit. In neuesten Plattformimplementierungen (z.B. IBM NetView TMN Support Facility Version 3 vom Herbst 1998) ist XMP/XOM nicht mehr enthalten. Dies hat auch nachträglich unsere Entscheidung bestätigt, XMP/XOM nicht zur Integration des ORB in die Kommunikationsinfrastruktur der Plattform zu verwenden.
- Der Empfang und die Weiterleitung von CORBA-Ereignismeldungen durch den *Postmaster*-Dämon (pmd) ist, wenn überhaupt, nur schwer realisierbar. Im Gegensatz zu SNMP-Traps und CMIP-Notifications ist eine CORBA-Ereignismeldung kein Protokollelement mit definierter PDU-Struktur, sondern ein beliebiger Methodenaufruf auf einem Consumer-Objekt. Hieraus folgt, daß der Code des Postmasters jedesmal erweitert werden müßte, wenn neue Ereignismeldungen definiert werden. Da sich die Arten von Ereignismeldungen zwischen Ressourcenklassen stark unterscheiden, kommt dies in der Regel häufig vor.
- Die gesamte Integration kann nur durch Erweiterung des Quellcodes von NetView vorgenommen werden, was bei einem kommerziellen Produkt natürlich illusorisch ist. Es müßten zumindest folgende Programmteile erweitert werden:
  - Der *Postmaster*, um XMP-Funktionsaufrufe in CORBA-Methodenaufrufe umzuwandeln und um CORBA-Ereignismeldungen zu empfangen und diese an den Filterprozeß weiterzuleiten.
  - Die Komponente zur Filterung von Ereignissen, um CORBA-Ereignismeldungen empfangen und verarbeiten zu können. Dies gestaltet sich insofern schwierig, als diese, wie bereits oben ausgeführt wurde, kein festes vordefiniertes Format haben.

- Der Log-Manager, um CORBA-Ereignismeldungen persistent speichern zu können.
- Die Zugriffsmechanismen des Postmasters auf die ORS-Datenbank, um sie für die Adressierung von CORBA-Objekten verwenden zu können.

Ferner impliziert die Belegung der XMP-Funktionsaufrufe mit entsprechenden Aufrufparametern, daß ein Managementsystem zu jedem Agenten wissen muß, mit welchem Managementprotokoll auf dessen Informationen zugegriffen werden kann. Die hierfür in der Praxis angewandte Lösung besteht darin, bereits an oberster Stelle der Topologiehierarchie zwischen den Protokollwelten zu unterscheiden.

Diese gravierenden Einschränkungen disqualifizieren den XOM/XMP-basierten Integrationsansatz für eine universell und flexibel anwendbare Lösung.

Alternative 2: Kapselung der Plattform-APIs

Die Alternative zu XMP/XOM besteht im unmittelbaren Zugriff auf die plattformspezifischen Infrastrukturdienste, wie z.B. Topologiedatenbank, Ereignisfilter oder Ressourcenverwaltung. Wie in Abschnitt 4.2.2 ausgeführt wurde, verfügt jede dieser Komponenten über eine eigene Programmierschnittstelle.

Die Idee dieses Integrationsansatzes besteht nun darin, die Programmierschnittstellen der vom Managementsystem angebotenen Dienste mit Hilfe sog. **IDL-Wrapper** zu kapseln, damit diese als CORBA-Objekte erscheinen und über den ORB von anderen Objekten angesprochen werden können. Die Verwendung von Wrappern ist gängige Praxis zur Integration älterer („legacy“) Technologien in neue objektorientierte Umgebungen. Dies ist möglich, weil objektorientierte Systeme nach dem Prinzip „Perception is Reality“ arbeiten: Es ist nicht erforderlich, daß ein System objektorientiert implementiert worden ist; maßgeblich ist lediglich, daß seine Bestandteile wie Objekte aussehen.

Die Nutzung dieser produktspezifischen und nicht standardisierten Funktionen impliziert allerdings den Verlust der Unabhängigkeit von bestehenden Plattformimplementierungen: Die Lösung ist damit vermutlich nur unter hohem Aufwand auf andere Plattformen zu portieren. In der Praxis wird dieser offenkundige Nachteil durch die Tatsache abgeschwächt, daß *IBM NetView* dieselben Wurzeln wie *HP OpenView* hat. Beide Plattformen decken, wie oben ausgeführt wurde, hinsichtlich ihres Anteils den Großteil des Marktes für Managementsysteme ab. Ferner gilt diese Abhängigkeit ebenso für alle auf einer Plattform aufsetzenden Managementapplikationen, was jedoch deren Anwendung in der Praxis nicht beeinträchtigt. Obwohl diese Einschränkungen ein allgemein anwendbares Verfahren ausschließen, überwiegen jedoch die oben beschriebenen Nachteile der ersten Alternative (nämlich die XMP/XOM-basierte Integration). Wir haben uns daher für die zweite Alternative entschieden, die uns überdies die Vorteile bietet, bestehende Plattform-Infratrukturdienste zu nutzen und eine integrierte, plattformgestützte Informationsbasis zu realisieren.

4.2.4 Nutzung bestehender Plattform-Infrastrukturdienste

Wir erläutern nun, wie die Basisdienste der Plattform für das Management von CORBA-Objekten eingesetzt werden können. Diese Integration ist insbesondere für die folgenden beiden Dienste notwendig, da sämtliche weiteren Dienste auf ihnen aufbauen:

- Die **Verarbeitung von Ereignismeldungen**. Hierzu zählt das Filtern und Weiterleiten an bestimmte Anwendungen, das Speichern in Log-Dateien und das Anzeigen an der Benutzeroberfläche. Um diese Dienste für CORBA-Ereignismeldungen nutzen zu können, müssen geeignete Verfahren und Schnittstellen entwickelt werden.
- Die topologische Darstellung der vorhandenen Ressourcen und ihrer Beziehungen zueinander an der Oberfläche der Plattform. Dies wird als **Topologiemanagement** bezeichnet. Es ermöglicht die Lokalisierung von Systemen, auf die in einem weiteren Schritt die Dienste des Konfigurations- und Fehlermanagements angewendet werden. Hinzu kommt die Visualisierung von Zustandsänderungen, welche durch Farbänderung der Symbole, welche die Ressourcen darstellen, an der Operatorkonsole angezeigt werden. Voraussetzung dafür ist, daß in den Datenbanken der Plattform ein Abbild der zu verwaltenden Systeme gespeichert ist. Die Mechanismen, die entwickelt wurden, um die Visualisierungsdienste des Managementsystems für die Darstellung der Ressourcen einer CORBA-Umgebung nutzen zu können, werden nachfolgend beschrieben.

Aufgrund unserer Ausführungen in Abschnitt 3.1.2, die unter anderem die im Rahmen der Aktivitäten der OMG Telecom DTF behandelten *Notification* und *Topology* Managementdienste beschreiben, erscheint es auf den ersten Blick erstaunlich, daß wir bei unserem Integrationsansatz diese beiden Dienste in den Mittelpunkt unserer Integration stellen. Die Begründung hierfür lautet, daß der CORBA Topology Service gegenwärtig nicht weiterentwickelt wird, und der Notification Service noch nicht verabschiedet wurde. Somit sind in gegenwärtigen CORBA-Implementierungen diese Dienste zwangsläufig noch nicht vorhanden. Sie sind jedoch von grundlegender Natur und essentiell, da sämtliche weiteren Managementdienste darauf angewiesen sind. Die Tatsache, daß die OMG Telecom DTF zuerst eine Normierung dieser beiden Dienste angestrebt hat, bestätigt diese Sichtweise. Die Nutzung dieser beiden von der Plattform bereitgestellten Dienste reflektiert folglich die Notwendigkeit, eine Substitution noch nicht vorhandener CORBA-Dienste durch Plattformdienste vorzunehmen.

Ferner kann bei diesen beiden Diensten eine Integration, bzw. eine Anbindung in dem Sinne stattfinden, daß ein größtmöglicher Anteil an vorhandenem Plattformcode zur Lösung der Aufgabenstellung verwendet werden kann. Bei anderen Plattform-Basisdiensten wie z.B. dem MIB-Browser oder der Schwellwertüberwachung besteht hingegen keine Möglichkeit, ihre Funktionalität auch für CORBA-Objekte zu nutzen, da sie speziell auf SNMP zugeschnitten sind. Programme, die eine ähnliche Funktionalität für CORBA erbringen, müßten vollständig neu implementiert werden. Diese könnten dann

per Integration in die Oberfläche dem Anwender zur Verfügung gestellt werden. Da die Datenbanken der Plattform über offengelegte Schnittstellen verfügen, können die benötigten Informationen von dort ausgelesen werden.

Infrastrukturdienste für ein integriertes Ereignismanagement

Die Verarbeitung asynchroner Ereignismeldungen durch eine Managementplattform geschieht im wesentlichen in folgenden Schritten:

1. Von den Agenten der überwachten Systeme werden (im Fehlerfall oder bei Überschreitung vorher definierter Schwellwerte) Ereignismeldungen an die Managementplattform gesandt, die diese empfängt und nach festgelegten Regeln filtert. Hauptaufgabe der Filterung ist es, aus den eingehenden Ereignis-Rohdaten Managementinformation zu gewinnen. Filterkriterien sind beispielsweise die Art der Ressource, der Typ und der Zeitpunkt der Ereignismeldung oder die Häufigkeit des Auftretens.

2. Die gefilterte Managementinformation wird anschließend der Managementapplikation bzw. dem Administrator zur Verfügung gestellt. Die durch den Administrator ausgelösten Aktionen werden über die Plattform an die Ressourcen zur Ausführung übermittelt.

Unser Lösungskonzept besteht nun darin, die vorhandenen NetView-*Event Management Services (EMS)* für das Management von Ereignissen so zu erweitern, daß sowohl SNMP-Traps als auch CORBA-Events von einer zentralen Stelle empfangen werden können. Wir stützen uns dabei auf den CORBA Event Service [OMGSS 97], der Mittel für die asynchrone Kommunikation zwischen verteilten Objekten spezifiziert. Hierbei wird zwischen Objekten, die Ereignisse erzeugen (sog. **Supplier**) und empfangenden Objekten (sog. **Consumer**) unterschieden. Die Kommunikation zwischen diesen Objektarten erfolgt durch Aufruf einer Methode des Consumers durch den Supplier; zur Entkopplung der beiden werden sogenannte **Event Channels** eingesetzt, die ihrerseits sämtliche Ereignisse, die von Suppliern kommen, an diejenigen Consumer weiterleiten, die sich für die betreffenden Events registriert haben. Man erhält somit die Möglichkeit, ein Ereignis mehreren Consumern, bzw. umgekehrt die Ereignisse vieler Supplier einem Consumer zuzustellen. Dies ist vorteilhaft, da die Anzahl von Consumer-Objekten für einen Supplier transparent ist: Letzterer muß lediglich eine Objektreferenz kennen (nämlich die des Event Channels), obwohl sich die Anzahl der Consumer-Objekte zur Laufzeit ändern kann. Ferner unterstützt der Event Service *typisierte* sowie *generische* Ereigniskommunikation. Der Unterschied zwischen diesen Arten besteht darin, daß generische Ereignismeldungen lediglich aus einem Parameter vom IDL-Datentyp `any` bestehen, während eine typisierte Ereignismeldung eine beliebige Anzahl von Parametern enthalten darf, deren Datentypen ebenfalls frei wählbar sind.

Angesichts der Tatsache, daß wir auf die in den Ereignismeldungen übermittelten Informationen Filter anwenden möchten, haben wir uns für typisierte Ereigniskommunikation

entschieden. Wir sind somit in der Lage, Filterkriterien zu definieren, die eine Klassifikation hinsichtlich der Art der Ressource (Netzkomponente, Endsystem, Anwendung), der Ereigniskategorie (Fehler, Topologie, Status) sowie der Wichtigkeit zulassen. Letzteres bietet uns den Vorteil lastbezogener Verarbeitung, da beispielsweise wichtige Ereignisse von einem Supplier per *Push*-Kommunikation einem Consumer unmittelbar zugestellt werden können, während weniger bedeutsame Ereignisse per *Pull*-Kommunikation dann von einem Consumer abgerufen werden können, wenn dieser gegenwärtig keine vordringlicheren Aufgaben erledigen muß.

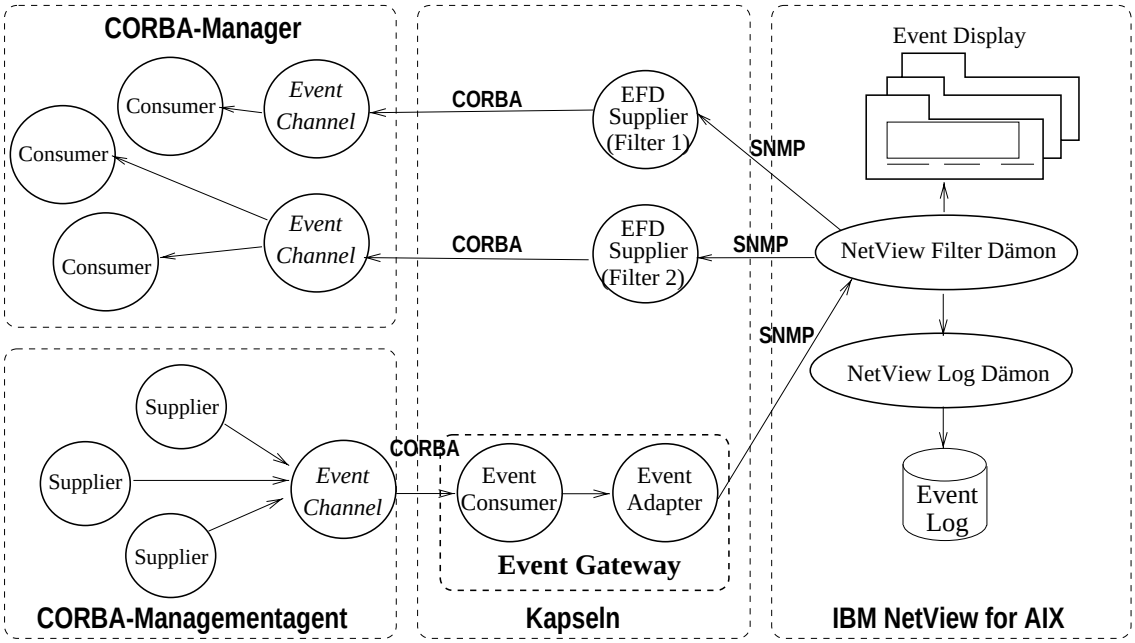


Abbildung 4.5: Weiterleitung asynchroner Ereignismeldungen

Da NetView selbst nicht in der Lage ist, CORBA-Events zu verarbeiten, müssen diese zunächst in SNMP-Traps umgewandelt werden. Wir haben dazu ein Event-Gateway implementiert (vgl. Abbildung 4.5), das die NetView-Ereignisdienste kapselt, um CORBA-Ereignismeldungen verarbeiten zu können. Dieses bietet eine `TypedConsumer`-Schnittstelle zum Empfang von CORBA-Ereignismeldungen, transformiert diese in SNMP-Traps und leitet sie an NetView zur weiteren Verarbeitung weiter.

Der Empfang und die Transformation laufen im einzelnen folgendermaßen ab: Das `EMS_Event_Consumer` Objekt registriert sich beim Event Channel für sämtliche Ereignisse aller Supplier. Die eigentliche Transformation der Ereignisse sowie die Weiterleitung an das Managementsystem geschieht durch das `Event_Adapter` Objekt, dessen Aufgabe darin besteht, eine *Enterprise-specific* Trap-PDU zu kreieren, mit den aus dem CORBA-Event übernommenen Parametern auszufüllen und diese an den Manager abzusenden. Diese Abbildung geschieht in Anlehnung an die in [OMG SNMP] gemachten Festlegungen und wird in einer Tabelle gespeichert. Naheliegenderweise müssen die

spezifischen Trap-Typen ebenfalls dem Manager bekanntgemacht werden, was entweder manuell oder (in unserem Fall) durch Konfiguration des NetView Trap-Dämons *trapd* per entsprechendem Funktionsaufruf geschieht.

Die IDL-Definition eines `Event_Adapter` Objekts lautet wie folgt:

```
interface Event_Adapter : SOMObject
{
 void create_pdu(in long tid, in string src);
 void add_arg_long(in long arg);
 void add_arg_string(in string arg);
 void send_pdu();
}
```

Eine Trap-PDU wird durch Aufruf der `create_pdu` Methode kreiert und mit dem Trap-Identifikator (`tid`) sowie der IP-Adresse des Quellsystems vorbelegt (`src`, ermittelt aus dem *host*-Parameter des CORBA-Events). Anschließend werden die im typisierten CORBA-Event enthaltenen Parameter mit den beiden `add_arg_()`-Methoden als „Variable-Bindings“ geschrieben. Schließlich wird die PDU mit `send_pdu()` an das Managementsystem abgesandt.

Wir haben diese IDL-Definition erwähnt, weil sie die Funktionsaufrufe des NetView SNMP-APIs [IBMNVPR 95] kapselt und somit als Beispiel eines *IDL Wrappers* angesehen werden kann. Da dieses API eine C-Programmierschnittstelle bietet, haben wir unter Zuhilfenahme des von der OMG standardisierten C-Language Mappings eine entsprechende Anbindung vornehmen können[3].

Nachdem die Ereignismeldungen empfangen wurden, können sie nun ebenso wie SNMP-Traps durch die NetView-Dienste gefiltert und protokolliert werden. Die Definition der Filterregeln kann mit den graphischen Werkzeugen des Managementsystems vorgenommen werden und bietet eine Vielzahl von Filterkriterien wie z.B. die Trap-ID, die IP-Adresse des Quellsystems, der Zeitpunkt des Auftretens, die Häufigkeit des Auftretens usw. Ebenfalls können die empfangenen und gefilterten Ereignismeldungen graphisch im *Event Display* des Managementsystems angezeigt werden (siehe dazu Abbildung 4.5). Es hat sich gezeigt, daß durch die Abstützung auf vorhandene Plattformdienste bereits mit verhältnismäßig geringem Aufwand gute Ergebnisse erzielt werden können (siehe dazu [Vogs 96]).

Wir hatten in der Anforderungsanalyse in Abschnitt 4.2.1 explizit gefordert, daß auch CORBA-basierte Managementapplikationen[4] in die Lage versetzt werden sollten, von der Managementplattform verarbeitete Ereignisse zugestellt zu bekommen. Der zweite Teil

[3] Unsere CORBA-Entwicklungsumgebung, das *IBM SOMobjects Developer Toolkit* erfordert, daß jede Objektklasse von der Basisklasse `SOMobject` abgeleitet wird, was kritisch hinsichtlich der Portabilität ist. Neueste CORBA-Toolkits weisen keine solchen Beschränkungen mehr auf.

[4] Dies können eigenständige verteilte Anwendungen sein oder Java-Applets, die in CORBA-fähigen WWW-Browsern ablaufen.

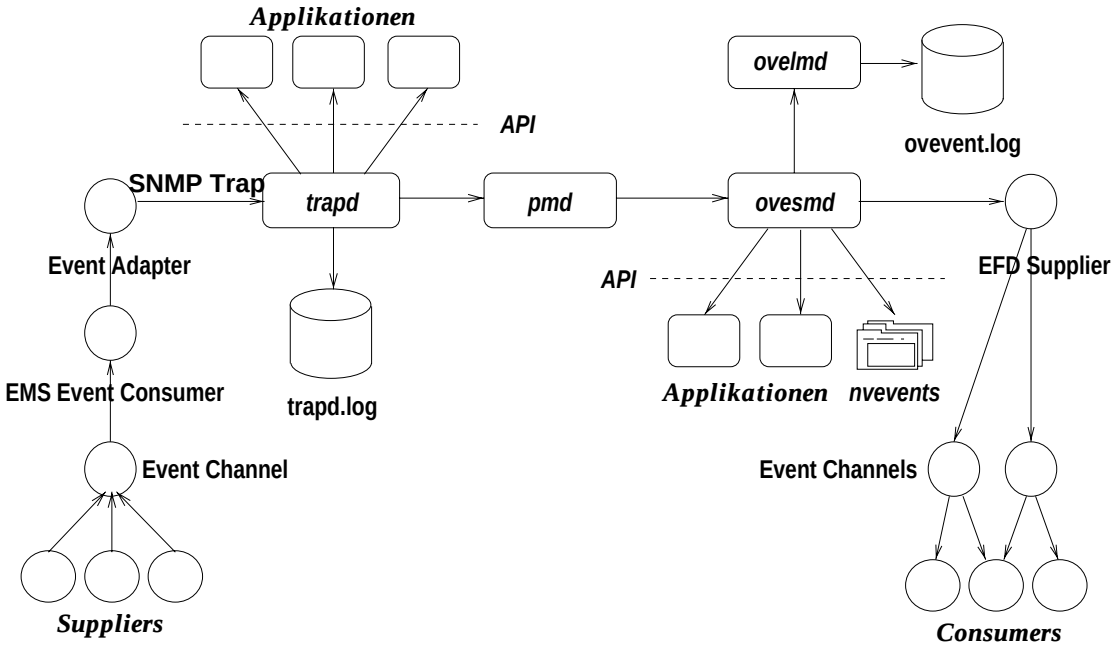


Abbildung 4.6: Module der Ereignisverarbeitung

der Aufgabe besteht nun darin, eine Infrastruktur zu definieren, die es ermöglicht, die vom Manager gefilterten Traps wieder in CORBA-Ereignismeldungen zurückzukonvertieren.

Hierbei agiert nun die Plattform als Event-Supplier und die CORBA-Applikation als Consumer, die wiederum durch Event Channels entkoppelt sind. Das Objekt, das den entsprechenden NetView-Dienst kapselt, wurde in Anlehnung an die OSI Event Report Management Function `EFD_Supplier` genannt, da seine Funktionalität identisch zu der eines OSI *Event Forwarding Discriminator (EFD)* ist. Die IDL-Definition dieser Objektklasse lautet wie folgt:

```
interface EFD_Supplier : SOMObject
{
 attribute string filter;
 void set_Dispatch(in Event_Dispatcher disp);
 oneway void activate();
}
```

Eine CORBA-Applikation registriert sich beim `EFD_Supplier`-Objekt für den Empfang gefilterter Ereignismeldungen einer bestimmten Art durch das Setzen des Attributs `filter` auf den gewünschten Filterstring. Hierfür können alle in NetView erlaubten Filterregeln verwendet werden. Bei der Initialisierung eines `EFD_Supplier`-Objekts durch den Aufruf seiner `activate()`-Methode wird der Inhalt des `filter`-Attributes ausgelesen und die Filterregel an den NetView Filter-Dämon *ovesmd* übergeben (siehe auch Abbildung 4.6). Jede Ereignismeldung, die die Filterregel passiert, wird in einen

CORBA-Event umgewandelt und über einen Event Channel an die registrierten Consumer (hier: Die jeweiligen Managementapplikationen) weitergeleitet. Die Auswahl des richtigen Event Channels ist Aufgabe des `Event_Dispatcher`. Für jede Filterart kann ein `EFD_Supplier`-Objekt nach Bedarf erzeugt werden, an dem beliebig viele Consumer über Event Channels angeschlossen sein können.

Obwohl die Weiterleitung und Verarbeitung der Ereignismeldungen in der Praxis gut funktionieren, hat unsere Implementierung (siehe hierzu die Ausführungen in [Vogs 96]) auch gezeigt, daß der Zwang zur Konvertierung der CORBA Events in SNMP-Traps die Achillesferse dieses Ansatzes darstellt: Einerseits müssen geeignete Abbildungen auf Trap-IDs erfolgen, die in separaten Tabellen vorgehalten werden, was bei einer großen Zahl von Ereignisarten einen nicht unerheblichen Verwaltungsaufwand mit sich bringt. Andererseits impliziert die Nutzung des Plattform-Filterdienstes die Beschränkung, daß die Filterkriterien stark SNMP-zentriert sind (z.B. Trap-ID, IP-Adresse der Quelle). Ein eigenständiger CORBA-Service, der die Definition leistungsfähiger Filterregeln zuläßt, ist mit dem *Notification Service* zwar angedacht, jedoch noch nicht abschließend standardisiert.

Nutzung des Plattform-Topologiedienstes für CORBA-Agenten

Die Topologieverwaltung überwachter Ressourcen zählt zu den Kernaufgaben von Managementplattformen. Sie gliedert sich in folgende Teilschritte:

1. Information über vorhandene Systeme und deren momentanen Zustand (up, down, usw.) wird ermittelt und in der Plattformdatenbank abgespeichert (Discovery-Funktion).
2. Ein übersichtliches Modell dieser Information (die Zustände der Ressourcen werden durch farbliche Kennzeichnung der entsprechenden Symbole dargestellt) wird anschließend an der graphischen Benutzeroberfläche der Plattform angezeigt.

Um der Dynamik der überwachten Ressourcen gerecht zu werden, wiederholt sich dieser Prozeß in regelmäßigen Zeitabständen, der gewöhnlich durch die Plattform initiiert wird (*Polling*). Zur Vorbeugung der dadurch entstehenden hohen Netzlast haben wir einen ereignisgesteuerten Ansatz verfolgt, der nur dann Ereignismeldungen an das Managementsystem verschickt, wenn sich Änderungen an der Topologie ergeben haben. Dies ist möglich, weil wir die CORBA-Agenten dahingehend instrumentiert haben, daß diese im Falle einer Änderung (Instantiierung, Zustandsänderung, Terminierung) geeignet typisierte asynchrone Ereignismeldungen aussenden. Wie Abbildung 4.7 verdeutlicht, liegt auch hier der Schwerpunkt unseres Integrationskonzeptes auf der Wiederverwendung des Plattform-Topologiedienstes *Generalized Topology Manager (GTM)*, der von uns analog zur Ereignisverarbeitung mit IDL-Wrappern gekapselt wurde. Dies ist die Grundvoraussetzung, um seine Nutzung durch die CORBA-basierte Discovery-Applikation (s.u.) überhaupt erst zu ermöglichen.

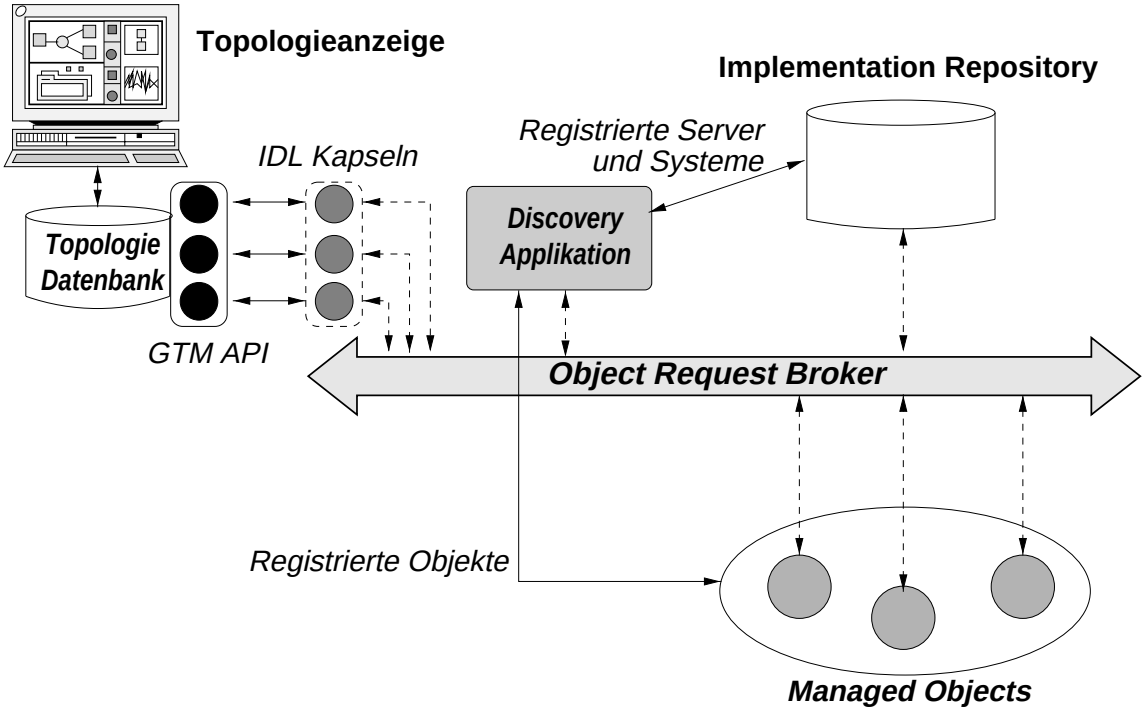


Abbildung 4.7: Komponenten der Topologieverwaltung

Der Vorteil der GTM-API beruht auf der Tatsache, daß sie dem Entwickler die graphische Darstellung der Managementobjekte abnimmt und die grundlegenden graphentheoretischen Datenstrukturen (Vertex, Arc, Graphs) bereitstellt. Der Kernbestandteil unserer CORBA-Topologieverwaltung besteht in der **Discovery-Applikation**, auf die wir als Beispiel für die Bildung einer plattformgestützten Informationsbasis im folgenden Abschnitt eingehen werden. Es sei an dieser Stelle noch erwähnt, daß von uns zwei Topologiemodelle sowohl für Endsysteme als auch für verteilte Anwendungen konzipiert und implementiert wurden, auf die wir jedoch aus Platzgründen nicht eingehen können. [Vogs 96] führt dies genauer aus.

Zusammenfassend haben unsere Erfahrungen bei der Implementierung der Topologieapplikation gezeigt, daß durch die Abstützung auf standardisierte bzw. von der Plattform zur Verfügung gestellte Infrastruktur-Dienste mit akzeptablem Aufwand eine brauchbare Lösung zur Topologieverwaltung erstellt werden konnte.

4.2.5 Schaffung einer plattformgestützten Informationsbasis

Die mit der Verwendung des plattformbasierten Integrationsansatzes einhergehende Vermeidung von SNMP erlaubt es uns, in einer „objektorientierten Welt“ zu bleiben: CORBA-Agentenobjekte müssen auf ihre Gegenstücke im Managementsystem abgebildet werden; eine Transformation der Informationsmodelle ist so nicht notwendig. Stattdessen muß durch eine Discovery-Applikation dafür gesorgt werden, daß neue CORBA-Ressourcen

vom Managementsystem zur Laufzeit erkannt und den darauf ablaufenden Managementapplikationen bekanntgemacht werden.

Die Managementinformationen bezüglich der zu verwaltenden Ressourcen können in zwei Klassen unterschieden werden:

- Information über die vorhandenen Ressourcen. Hierzu muß in den Datenbanken der Plattform ein abstraktes Modell des zu überwachenden Systems (d.h. eine MIB-Definition) gespeichert sein und aktuell gehalten werden.
- Information über die Instrumentierung der Ressourcen. Im Fall von CORBA-Managementobjekten befindet sich diese Information in den IDL-Beschreibungen der Objektklassen. Zur Gewinnung der Information über CORBA-Managementobjekte gibt es zwei Alternativen:
  - Es ist bereits zur Übersetzungszeit einer Managementapplikation bekannt, mit welchen MOs sie kommunizieren wird. Somit erfolgen die Aufrufe statisch über die IDL-Stubs der jeweiligen MOs. Dieser Fall hat keine praktische Bedeutung, da anwendungsbezogene Managementobjekte sehr dynamischer Natur sind und ihre Anzahl ständig variiert.
  - Zur Laufzeit kann die Information über Objektklassen und deren Instanzen aus den Interface- bzw. Implementation-Repositories gewonnen werden, damit anschließend auf den identifizierten Objekten Methodenaufrufe ausgeführt werden können.

Discovery-Applikation

Nachstehend folgt die Beschreibung der Implementierung einer Discovery-Applikation für CORBA-Managementobjekte (dies können sowohl verteilte Anwendungen als auch Endsysteme sein), die Aufschluß darüber gibt, wie die Anbindung an den Topologiedienst von NetView erfolgt. Die Datenbank der Plattform wird so zu einer integrierten Informationsbasis für SNMP- und CORBA-Managementobjekte.

Die Discovery-Applikation basiert auf dem NetView GTM, der, wie im vorigen Abschnitt erwähnt wurde, abstrakte Datenobjekte zur Darstellung von Ressourcen bereitstellt und über ein API zugänglich ist. Damit der GTM CORBA-Objekte geeignet darstellen kann, benötigt er von der Discovery-Applikation (siehe Abbildung 4.7) folgende Angaben: die insgesamt existierenden Applikationen oder Dienste, die Systeme (Hosts) und die Server (bzw. die Referenzen der Objektklassen, aus denen sie bestehen), die auf den Systemen laufen, sowie die Objektinstanzen, die auf einem Server aktiv sind. Diese Informationen wurden folgendermaßen ermittelt: das CORBA *Implementation Repository* enthält Angaben über verfügbare Systeme, auf ihnen laufende Server und eine Liste der durch letztere unterstützten Objektklassen. Die zu den jeweiligen Objektklassen existierenden Objektinstanzen werden durch den CORBA *LifeCycle Service* bereitgestellt. Über die Methoden der `Naming Context` Objekte (spezifiziert im CORBA Naming Service)

können alle darin enthaltenen Namensbindungen abgefragt werden; man kann damit die Namen und Objektreferenzen aller registrierten Objektinstanzen ermitteln.

Die Nutzung der Datenbanken des Managementsystems als integrierte Informationsbasis ergibt sich nicht zuletzt aus der Kapselung der Dienste-APIs. Der Schwerpunkt unseres Integrationskonzeptes liegt, wie ursprünglich gefordert, auf einem möglichst hohen Verfügbarkeitsgrad von Infrastruktur-Diensten, die bereits in der Plattform vorhanden sind. Dies gilt nicht zuletzt für die in den vorangehenden Abschnitten betrachteten Applikationen, die die Topologie von CORBA-Agenten ermitteln, bzw. die von ihnen ausgesendeten CORBA-Events auf SNMP-Traps (und umgekehrt) abbilden. Hierbei werden ebenfalls Infrastruktur-Dienste genutzt, die die Managementplattform bereitstellt, um die ermittelten Informationen einer CORBA-konformen Managementapplikation zur Weiterverarbeitung anzubieten. Die Plattform fungiert dabei, wie bereits vorher besprochen, als Brücke zwischen CORBA-Agenten und CORBA-Managementapplikationen.

4.2.6 Fazit: Eignung multiarchitektureller Manager

Die beiden in Abschnitt 4.2.2 beschriebenen Alternativen der Anbindung eines ORBs an eine Managementplattform haben durch die Heterogenität der Informationsmodelle die Einschränkung, daß auf Managementobjekten einer Protokolldomäne nur diejenigen Funktionen anwendbar sind, die das entsprechende Protokoll erlaubt; *Scoping* und *Filtering*, essentieller Bestandteil des OSI-Modells, kann weder für das Management von SNMP- noch von CORBA-Ressourcen verwendet werden. Ein anderer Aspekt, der aufzeigt, daß dieser Integrationsansatz niemals nahtlos sein kann, ist die Tatsache, daß zahlreiche Werkzeuge und Applikationen, die mit der Plattform mitgeliefert werden, explizit auf eine bestimmte Architektur zugeschnitten sind: Ein MIB-Browser, der auf SNMP zugeschnitten ist, kann nicht die Attributwerte von CORBA- oder OSI-konformen MOs anzeigen. Die Möglichkeit, jeweils auf andere Architekturen zugeschnittene Werkzeuge mit ähnlichem Funktionsumfang in die Plattformoberfläche zu integrieren, kann dabei nicht über den Verlust an Transparenz hinwegtäuschen. Es sei an dieser Stelle nochmals erwähnt, daß die Ausführungen, die sich hier am konkreten Produkt NetView orientieren, prototypischer Art für diese Klasse von Integration ist und folglich keine Unzulänglichkeiten des von uns gewählten Produkts sind.

Ein weiterer prinzipieller Gesichtspunkt, der die Problematik dieses Integrationsansatzes für offenes integriertes Enterprise Management verdeutlicht, besteht darin, daß auch im Falle einer geglückten Integration bereits auf oberster Ebene der Topologiehierarchie zwischen den zugrundeliegenden Managementarchitekturen unterschieden wird: Man findet, wie es in Abbildung 4.8 dargestellt ist, für jede Architektur (OSI/TMN, SNMP, CORBA) mindestens ein Symbol, unter dem dann die Ressourcen zu finden sind, die über diese Architektur verwaltet werden können. Genau dies sollte eigentlich verhindert werden, da man dem Betreiber die Anordnung MOs gemäß seiner Betriebsziele (d.h. nach organisatorischen oder geographischen Kriterien) ermöglichen möchte. Das Problem des Fortlebens

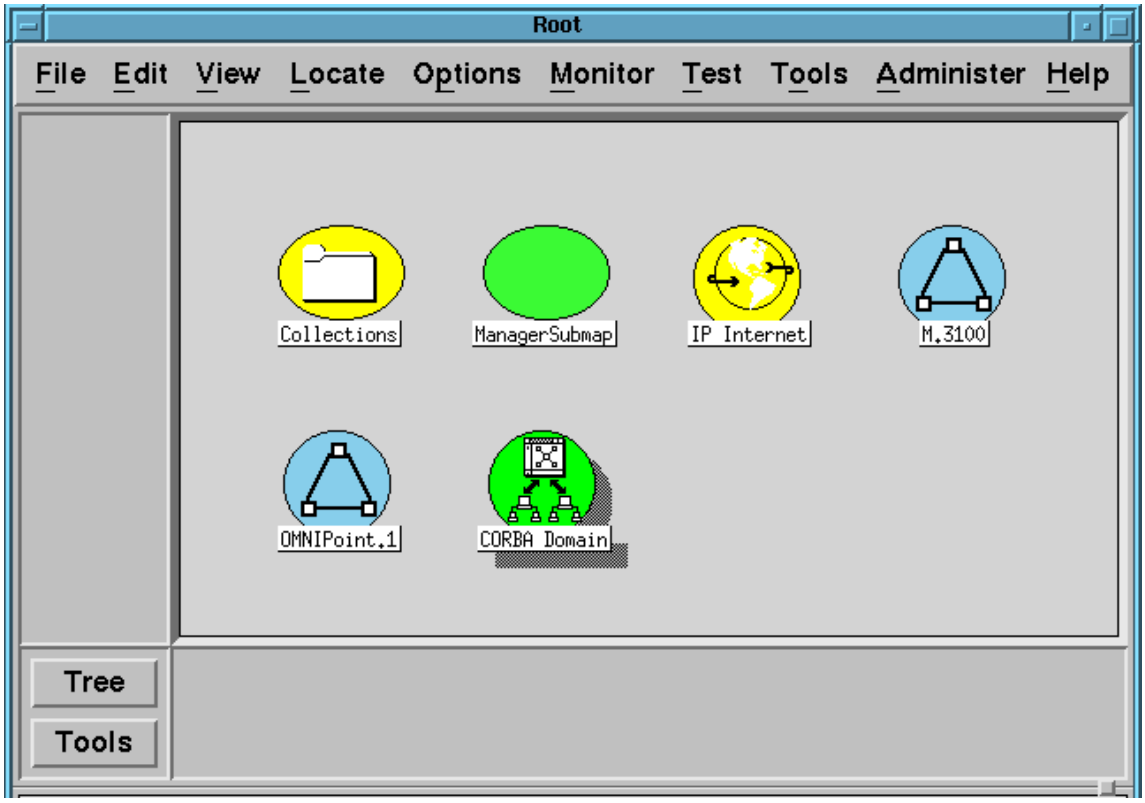


Abbildung 4.8: Root-Fenster der multiarchitekturellen Plattform IBM NetView

unterschiedlicher Architekturen unter einer gemeinsamen Oberfläche bleibt bei diesem Integrationsansatz bestehen.

Wie bereits in Abschnitt 4.2.3 eingehend begründet wurde, ist ein weiteres Problem bei der Implementierung multiarchitektureller Plattformen – nämlich die Abhängigkeit von bestehenden Plattformimplementierungen – inhärent mit diesem Integrationsansatz verbunden: Die Nutzung dieser produktspezifischen und nicht standardisierten Funktionen ist daher häufig nicht nur von Produkten abhängig, sondern auch in hohem Maße von Modifikationen dieser APIs durch den Hersteller.

Nichtsdestoweniger haben wir mit unserer Integration die Basis geschaffen, um trotz des gegenwärtigen Nichtvorhandenseins offener CORBA-Plattformen eine solche zu realisieren. Somit sind wir in der Lage, für die in Kapitel 5 vorgestellten Agenten eine CORBA-Managementumgebung zu schaffen. Gleichzeitig können wir die Vielfalt an bereits vorhandenen Plattformdiensten als einen temporären Ersatz für in der Standardisierung befindliche CORBA-Dienste ansehen und zur Lösung unseres Integrationsproblems einsetzen.

4.3 Multiarchitektureller Agent

Wir werden nun die zweite der drei Alternativen zur Kooperation von Managementarchitekturen vorstellen, die auf der Verwendung multiarchitektureller Agenten basiert.

Bevor wir den Nutzen und die Anwendbarkeit dieses Integrationsansatzes untersuchen, müssen wir zunächst diesen Begriff gegenüber einem auf den ersten Blick damit verwandten Konzept abgrenzen, nämlich dem des **Proxy-Agenten**.

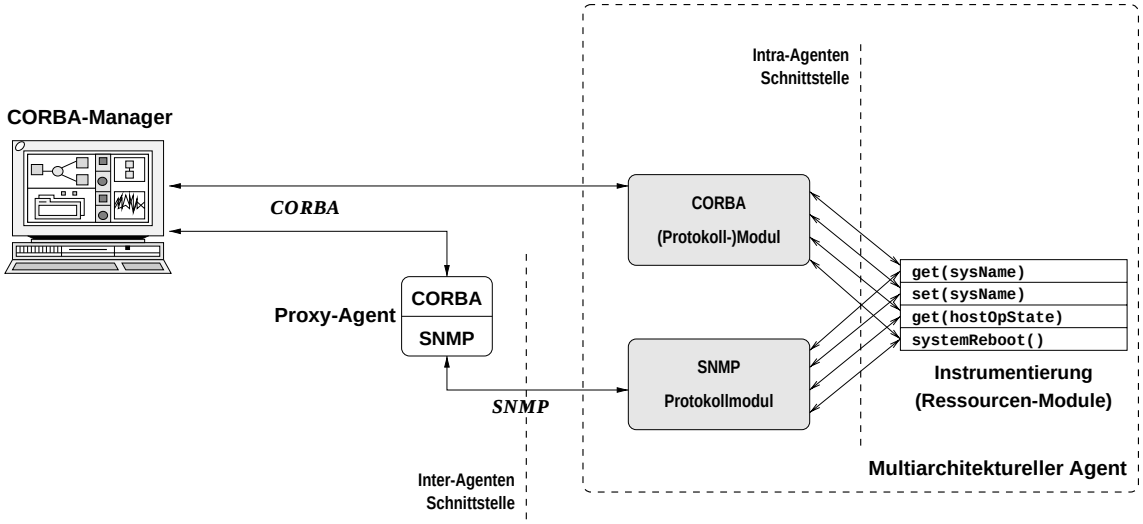


Abbildung 4.9: Multiarchitektureller Agent vs. Proxy-Agent

Die grundlegenden Architekturkonzepte von Managementagenten haben wir in Abschnitt 2.1 vorgestellt: Die Instrumentierung von Ressourcen erfolgt generell in Form sog. Ressourcen-Module, deren Schnittstellen in einer gewöhnlichen Programmiersprache (zumeist C oder C++) vorliegen. Die Managementinstrumentierung dieser Ressourcen-Module erfolgt durch Protokoll-Module, die einerseits eine Instanz der Protokollmaschine enthalten, andererseits die MIB-Struktur der administrierten Ressourcen verwalten, um die korrekte Adressierung der Managementinformation zu gewährleisten. Die Schnittstelle zwischen Ressourcen- und Protokoll-Modulen haben wir als Intra-Agenten-Schnittstelle bezeichnet. Ein multiarchitektureller Agent verfügt über *mehrere* Protokoll-Module zu *jeweils einem* Ressourcen-Modul, d.h. es erfolgt stets eine Abbildung an der Intra-Agenten-Schnittstelle. Grundsätzlich wird dabei immer die interne (proprietäre) Instrumentierung auf eine offene Managementarchitektur abgebildet.

Demgegenüber besteht die Aufgabe eines Proxy-Agenten darin, einen eigenständigen, d.h. bereits mit einem (ggf. offengelegten) Protokoll-Modul versehenen Agenten in einen fremden Managementkontext einzubinden. Der Agent kommuniziert demzufolge mit dem Proxy-Agenten über eine Inter-Agenten-Schnittstelle (siehe auch Abbildung 4.9). Im Gegensatz zum multiarchitekturellen Agenten kann ein Proxy-Agent insofern als Spezialfall eines Management-Gateways angesehen sehen, als ein Proxy-Agent jeweils in einer

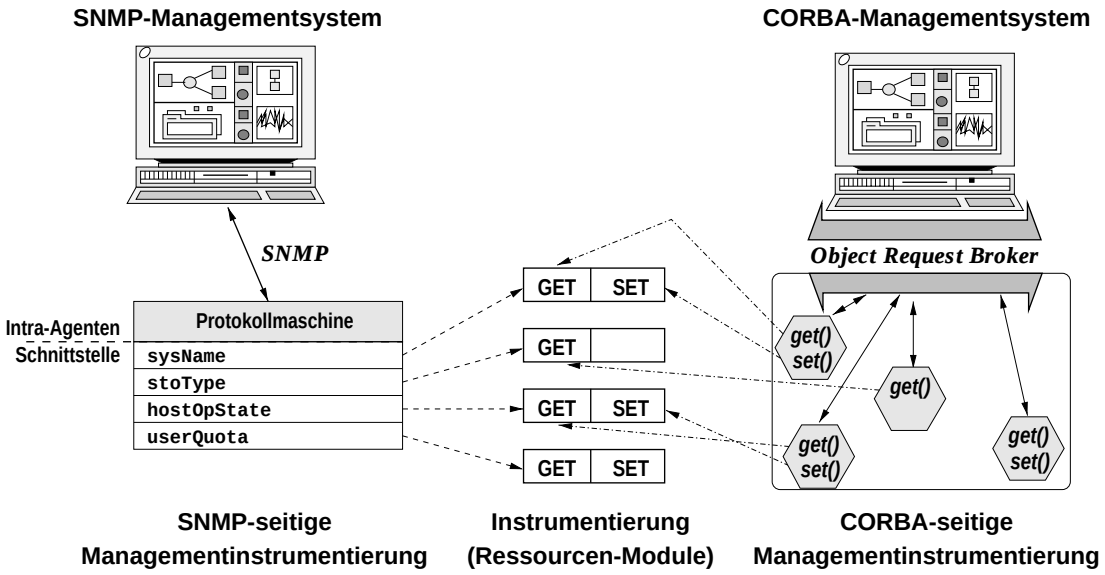


Abbildung 4.10: Multiarchitektureller CORBA- und SNMP-Agent

1:1-Beziehung mit dem zu integrierenden Agenten steht; beim Gateway ist dies jedoch üblicherweise eine 1:n-Beziehung.

Die praktische Durchführbarkeit des Integrationsansatzes auf der Basis multiarchitektureller Agenten hängt maßgeblich von der Art der Ressourcen ab: Offensichtlich besteht bei preiswerten Netzkomponenten, in denen sich der vollständige Agent bereits in der Firmware des Gerätes befindet, keinerlei Möglichkeit, eine unserer Definition des multiarchitekturellen Agenten genügende Integration vorzunehmen. Bessere Chancen für diesen Ansatz bestehen dagegen bei Ressourcenklassen, die (System-)Dienste oder Softwarekomponenten darstellen. Hier kann die jeweilige Programmierschnittstelle als Vereinigungsmenge der Ressourcenmodule aufgefaßt werden, auf die die entsprechende Managementinstrumentierung aufgesetzt werden kann. Als charakteristische Beispiele für derartige Managementinstrumentierungen gelten die in Abschnitt 3.1.3 vorgestellten Host Resources bzw. Application MIBs der IETF, welche die benötigte Managementinformation zum Teil anhand von Betriebssystemaufrufen ermitteln.

Die besten Realisierungsmöglichkeiten für eine effiziente Managementinstrumentierung sind naturgemäß dann gegeben, wenn die Ressourcen-Module im Quelltext vorliegen, was jedoch nur in den seltensten Fällen gegeben ist. Der Integrationsansatz des multiarchitekturellen Agenten bietet sich daher insbesondere für die Hersteller von Systemen bzw. Anwendungen an, die so auf einer gegebenen Instrumentierung parallel mehrere Managementarchitekturen unterstützen. Abbildung 4.10 stellt dies graphisch dar.

Für das von uns betrachtete Anwendungsszenario geht es nun darum, bestehende SNMP-Agenten unter den folgenden Randbedingungen um eine CORBA-konforme Zugriffsmöglichkeit zu erweitern: Es ist wünschenswert, daß die Vorgaben an die Managementinstrumentierung hinsichtlich der bereits unterstützten Architektur auch für das De-

sign der neu zu unterstützenden Managementarchitektur möglichst umfassend übernommen werden können. Dies würde eine signifikante Vereinfachung für die Entwicklung der neuen Managementinstrumentierung bedeuten, da man so bereits auf einer Grundmenge an Managementinformation aufsetzen könnte. Idealerweise sollte das Verfahren zur Transformation der Managementinstrumentierung offengelegt (d.h. standardisiert) und auf möglichst alle Ressourcenklassen universell anwendbar sein (d.h. es sollte *effektiv* sein). Ferner sollte die Spezifizität einer Managementarchitektur in bezug auf ihre Leistungscharakteristika zum Tragen kommen, wie zum Beispiel besonders performante Abfragemechanismen oder ein übersichtliches Design (d.h. das Verfahren zur Abbildung sollte zusätzlich *effizient* sein).

Im Grunde genommen geht es also hier um die Frage der allgemeingültigen Abbildbarkeit von Management-Informationsmodellen, da in ihnen die Struktur der Managementinstrumentierung festgeschrieben wird. Wir werden im nächsten Abschnitt im Rahmen der Management-Gateways demonstrieren, daß solche generischen Abbildungsvorschriften existieren, die eine *effektive* Umsetzung der Management-Informationsmodelle leisten. Andererseits werden wir auch feststellen, daß diese Umsetzung oft nicht *effizient* ist, da die bereits in Kapitel 3 analysierten Unterschiede zwischen den Management-Informationsmodellen tiefgreifend sind. Um trotz der vorhandenen Unterschiede eine effiziente Transformation zwischen unterschiedlichen Informationsmodellen vornehmen zu können, haben wir im Zuge unserer Untersuchungen ein Verfahren entwickelt, das dies leistet. Wir werden es in Abschnitt 5.1 vorstellen.

4.4 Management-Gateways

Dieser Abschnitt stellt nun die dritte Alternative zur Erreichung des Umbrella Managements vor. Es handelt sich hierbei um Management-Gateways, die, ähnlich wie die im vorigen Abschnitt angesprochenen Proxy-Agenten, zwischen Managern und Agenten agieren. Der konzeptionelle Unterschied zu letzteren besteht im wesentlichen darin, daß ein Proxy-Agent jeweils *einen einzelnen* Agenten, der in der Regel nur mit proprietären Mechanismen angesprochen werden kann, in den Managementkontext eines Managementsystems integriert, während Management-Gateways dies für eine *beliebige* Anzahl von Agenten tun. Ferner vermitteln letztere hauptsächlich zwischen offenen, standardisierten Managementarchitekturen.

Zuerst formuliert Abschnitt 4.4.1 die grundsätzlichen Anforderungen an Management-Gateways. Sie verdeutlichen, daß die Aufgabe von Management-Gateways weit über die bloße Umsetzung von Managementprotokollen hinausgeht, wenn die Integration möglichst umfassend und nahtlos sein soll[5]. Die hierzu notwendigen Schritte werden im

[5]Nichtsdestoweniger hat sich in der Fachwelt zur Beschreibung der Gateways eine „protokollorientierte“ Sichtweise durchgesetzt, d.h. man spricht von „CMIP/SNMP Gateways“, wenn Gateways zur Integration

einzelnen anhand der vier Teilmodelle des Netzmanagements in den Abschnitten 4.4.2 bis 4.4.5 erläutert. Jeder Schritt beschreibt dabei die grundsätzliche Problemstellung und geht anschließend auf unsere Erfahrungen bei der Implementierung zweier von uns entwickelter Prototypen ein. Hierbei handelt es sich um:

1. Ein CMIP/SNMP Gateway [Lang 96], dessen Ziel in der Integration von SNMP-Ressourcen in das OSI/TMN-Management besteht, das Managementsystem also TMN-basiert ist, während die Agenten ausschließlich eine SNMP-Protokollmaschine besitzen.

2. Ein CORBA/SNMP Gateway [Hoel 96], das einem CORBA-basierten Managementsystem den Zugriff auf SNMP-Ressourcen gestattet.

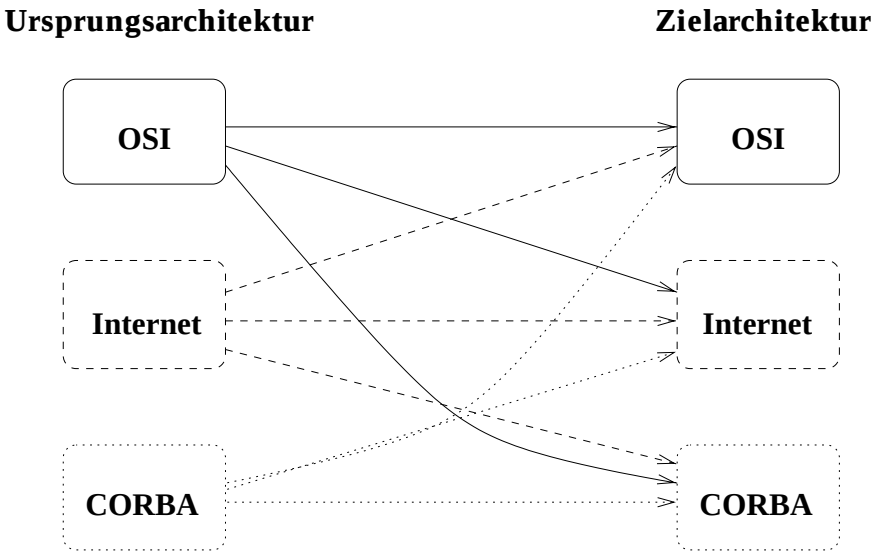


Abbildung 4.11: Abbildungsmöglichkeiten zwischen Managementarchitekturen

Neben den Umsetzungen des Internet-Managements nach OSI/TMN bzw. CORBA sind natürlich noch weitere Abbildungsmöglichkeiten zwischen Managementarchitekturen denkbar, die in Abbildung 4.11 dargestellt sind: Es sind dies die Übergänge für SNMP/CMIP, SNMP/CORBA sowie CORBA/CMIP bzw. CMIP/CORBA, wobei die erstgenannte Architektur sich jeweils auf die Zielarchitektur (d.h. die des Managementsystems) bezieht und letztere die Quellarchitektur (d.h. die der zu integrierenden Agenten) bezeichnet.

Wir beschränken uns aus folgenden Gründen auf die CMIP/SNMP- sowie CORBA/SNMP-Integrationsszenarien:

---

des Internet-Managements in OSI/TMN gemeint sind. Aus Konsistenzgründen werden wir ebenfalls diese Bezeichnungen verwenden.

- Es geht uns darum, die dem Umbrella Management zugrundeliegenden Konzepte aufzuzeigen. Hierzu ist es notwendig, zunächst das Vorgehen bei der Kopplung von zwei protokollbasierten Architekturen (in unserem Fall: OSI/TMN- und SNMP-basiertes Management) aufzuzeigen und anschließend die Interoperabilitätsaspekte einer protokollbasierten Managementarchitektur (Internet-Management) mit einer auf verteilten Objekten realisierten Architektur (CORBA) zu demonstrieren. Die restlichen vier Interoperabilitätsszenarien laufen prinzipiell nach einem ähnlichen Schema ab und unterscheiden sich nicht konzeptionell von den in dieser Arbeit behandelten Ansätzen. Insbesondere der Fall CMIP/CORBA hat durch die in Kapitel 3 angesprochene Migration von OSI/TMN zu CORBA eine wohl relativ geringe Praxisrelevanz, da die Einführung einer neuen Managementarchitektur primär auf Seiten einiger Managementsysteme geschieht und weniger bei einer unter Umständen sehr hohen Zahl von Agentensystemen.
- Das Ziel unserer Konzepte für das Umbrella Management besteht in einer möglichst universellen Anwendbarkeit. Aufgrund der hohen Verbreitung SNMP-basierter Managementagenten ist es naheliegend, das Internet-Management als Ressourcen-Architektur auszuwählen. Ferner reflektiert dies auch das Szenario zahlreicher Netzbetreiber, neue Managementarchitekturen in ein SNMP-basiertes Umfeld zu integrieren. Wir sehen daher von denjenigen Interoperabilitätsszenarien ab, die das OSI-Management als Agentenarchitektur haben (SNMP/CMIP und CORBA/CMIP).
- Der Nutzwert des Umbrella Managements wird besonders deutlich, wenn über das bloße Bereitstellen von Interoperabilität zusätzlich eine konzeptionell schwächere Managementarchitektur (hier: das Internet-Management) um Mechanismen einer leistungsfähigeren Architektur (hier: OSI/TMN bzw. CORBA) erweitert werden kann. Beispiele für den ersten Fall sind neue *Scoping* und *Filtering* Möglichkeiten für das Internet-Management; der Mehrwert von CORBA für das Internet-Management besteht unter anderem in automatischen Discovery-Mechanismen. Umgekehrt erscheint es aus diesen Gesichtspunkten heraus nicht angebracht, eine konzeptionell leistungsfähige Architektur in eine schwächere Architektur zu integrieren. Dies schließt neben dem oben angesprochenen SNMP/CMIP- insbesondere das SNMP/CORBA-Interoperabilitätsszenario aus.

Um insbesondere die Unterschiede zwischen der Integration zweier protokollbasierter Managementarchitekturen mit denen der Kopplung einer protokollbasierten und einer auf verteilten Objekten basierenden Architektur herauszustellen, werden wir die erforderlichen Transformationen jeweils anhand der einzelnen Teilmodelle vorstellen.

4.4.1 Anforderungen an Management-Gateways

Wir werden in diesem Abschnitt die grundlegenden Anforderungen an Management-Gateways erarbeiten, die uns später helfen werden, die Effektivität dieses Integrationsansatzes zu bewerten.

An Manager und Agent sollen keinerlei Modifikationen erfolgen

Diese Forderung besagt, daß der Übergang in die SNMP-Umgebung für das (OSI/TMN- bzw. CORBA-konforme) Managementsystem vollkommen transparent bleibt. Eine wichtige Implikation dieses Postulats besteht darin, daß für den Zugriff auf SNMP-Ressourcen keinerlei Änderungen am Code des Managementsystems geschehen. So schickt ein CORBA-Managementsystem CORBA-Requests an die „Stellvertreter-Objekte“ (*Proxy-Objekte*), die das Gateway bereitstellt und erhält CORBA-Responses von diesen zurück. Im Falle von OSI/TMN ist das Vorgehen analog. Für beide Fälle gilt, daß für das Managementsystem dabei die Umsetzung der Anfragen auf SNMP-Ressourcen nicht erkennbar ist.

Ferner dürfen beim Entwurf des Gateways keine Annahmen über die Erweiterbarkeit von Agenten gemacht werden: Die Erweiterung von Agenten, beispielsweise um eine CORBA-Schnittstelle (vgl. Abschnitt 4.3 über multiarchitekturelle Agenten), ist lediglich selten und nur auf hochperformanten Ressourcen möglich. Bei der Mehrheit der Agenten stehen weder Quellcode noch eine Programmierschnittstelle zur Verfügung. Deshalb dürfen am Agenten keine Modifikationen für die Kommunikation mit dem Gateway erfolgen; der Zugriff auf alle Agenten darf ausschließlich über die standardisierte SNMP-Schnittstelle erfolgen. Ebenfalls muß die Syntax und Semantik des Informationsaustausches aus Sicht des Agenten gleich bleiben, ungeachtet dessen, ob mit einem SNMP-Manager oder mit dem Gateway kommuniziert wird.

Darstellung der Agenten in einer für den Manager verständlichen Form

Dieses Postulat ist eine unmittelbare Konsequenz aus der ersten Forderung, da die Hauptaufgabe eines Gateways darin besteht, die Inhalte der Agenten-MIBs dem Manager zugänglich zu machen. Die entsprechenden CORBA- bzw. OSI-konformen Objektinstanzen sind Proxy-Objekte der Managementobjekte in der SNMP-Umgebung.

Aufgabe des Gateways ist es nun, die Managementobjekte in Form von Proxy-Objekten konsistent zum Manager hin abzubilden:

1. Jede Variableninstanz eines SNMP-Agenten wird von genau einem Attribut oder von einer Methode eines Proxy-Objektes repräsentiert. Letzterer Fall ist beispielsweise dann denkbar, wenn das Setzen eines Wertes in einem SNMP-Agenten die Ausführung einer Aktion anstößt (sogenannte „Pushbutton“-Variablen).

2. Es darf kein Proxy-Objekt existieren oder erzeugt werden, wenn es zu den Attributen des Objekts keine entsprechenden SNMP-Variableninstanzen gibt.

3. Wenn Inhalte der SNMP-Informationseinheiten in den Attributen von Proxy-Objekten gespeichert werden, müssen diese Attributwerte *zum Zeitpunkt der Abfrage* mit den Werten in den entsprechenden Managed Objects konsistent sein.

Weiterleitung von Operationen

Sämtliche Operationen, die das Managementsystem auf Proxy-Objekten ausführt, müssen in die SNMP-Umgebung weitergegeben werden. Voraussetzung ist dabei natürlich, daß das Gateway bei einem Zugriff erkennen kann, welches Proxy-Objekt – und damit welches Managed Object auf welchem SNMP-Agenten – das Zielobjekt des Zugriffs ist. Außerdem muß die Art des Zugriffs (lesen, schreiben, löschen, ...) für das Gateway erkennbar sein. Hieraus ergeben sich weitere Anforderungen:

- Eine Leseoperation auf ein Attribut eines Proxy-Objektes muß auf eine Leseoperation (eine *GetRequest-PDU*, eine *GetNextRequest-PDU* oder eine *GetBulk-PDU*) auf das bzw. die entsprechenden Managementobjekte abgebildet werden.
- Eine Schreiboperation auf ein Attribut eines Proxy-Objektes muß auf eine Schreiboperation (also eine *SetRequest-PDU*) auf das dazugehörige Managed Object abgebildet werden.
- Beim Löschen und Erzeugen eines Proxy-Objektes müssen die betroffenen Managed Objects gelöscht bzw. erzeugt werden. Dieser Fall betrifft hauptsächlich die Proxy-Objekte, die eine SNMP-Tabellenzeile repräsentieren, da nur Managementobjekte einer solchen SNMP-Tabellenzeile „gelöscht“ (d.h. als ungültig gekennzeichnet) oder „erzeugt“ (schreibender Zugriff auf eine in der SNMP-Tabelle noch nicht existierende Zeile) werden können.

Um Operationen auf Proxy-Objekten in die SNMP-Umgebung weiterzuleiten, ist es notwendig, Proxy-Objekte und deren Attribute auf entsprechende Managementobjektinstanzen abzubilden. Diese Abbildung findet im Gateway statt. Für den Zugriff auf eine konkrete Variableninstanz in der SNMP-Umgebung wird zur Generierung einer SNMP-Protokolldateneinheit generell folgende Information benötigt:

- Die IP-Adresse der Ressource, auf der der Agent läuft,
- der Objektidentifikator (OID) der Variable im Internet-Registrierungsbaum und
- ein *Zugriffsidentifikator*, d.h. eine Erweiterung des Objektidentifikators, der die Instanz des Objektes identifiziert. Bei skalaren Variablen ist dies eine „.0“, bei Tabellen ein zusammengesetzter Index-Objekttyp, der Instanzen der Elemente einer Tabellenzeile eindeutig kennzeichnet.

Diese Informationen müssen in der jeweils notwendigen SNMP-PDU enthalten sein und somit vom Gateway bereitgestellt werden.

Umgekehrt müssen Operationen, die in der SNMP-Umgebung stattfinden, zum Manager hin weitergeleitet werden.

- Eine asynchrone Ereignismeldung (`Trap-PDU`) eines SNMP-Agenten muß auf einen CORBA-Event bzw. eine CMIS-*Notification* abgebildet werden.

- Wird in der SNMP-Umgebung ein Managed Object neu erzeugt (wenn beispielsweise ein Agent gestartet wird), so müssen entsprechende Proxy-Objekte erzeugt werden. Analog müssen Proxy-Objekte entfernt werden, wenn die entsprechenden Managed Objects gelöscht wurden.

Allgemeine Anforderungen an Management-Gateways umfassen:

- Der Zugriff auf SNMP-Ressourcen sollte möglichst effektiv und effizient sein. Requests an Proxy-Objekte sollen quasi-parallel bearbeitet werden und sich nicht gegenseitig blockieren, wenn SNMP-Operationen ausgeführt werden. Ein Synchronisationskonzept sollte mögliche Verklemmungen verhindern.
- Ein Sicherheitskonzept soll zumindest die von SNMP gewährte Sicherheit bezüglich des unbefugten Zugriffs garantieren.
- Ein Management-Gateway soll hinsichtlich seiner Funktionalität (z. B. Führen von Statistiken, Schwellwertüberwachung) unabhängig von den Proxy-Objekten erweitert werden können, d.h. es sollte über einen Minimalumfang an generischer Managementfunktionalität verfügen.
- Die Funktionalität der Proxy-Objekte sollte erweiterbar sein. Hintergedanke ist die schrittweise Delegierung von Managementaufgaben an das Gateway – respektive die Proxy-Objekte – mit dem Ziel, das Managementsystem zu entlasten.

4.4.2 Umsetzung der Organisationsmodelle

Die Integration des Internet-Managements in das OSI-Management bereitet hinsichtlich des Organisationsmodells keinerlei Schwierigkeiten, da beide Architekturen auf dem Manager/Agent-Paradigma aufbauen.

Ferner gilt für alle in Abschnitt 3.1 angesprochenen Managementarchitekturen, daß die Umsetzung der Organisationsmodelle in ein CORBA-basiertes Enterprise Management ebenfalls unproblematisch ist, da ein Peer-to-Peer Ansatz (wie ihn CORBA verfolgt) im wesentlichen äquivalent zum Manager/Agent-Paradigma ist. Der hauptsächliche Unterschied besteht darin, daß die Rollen von Client und Server bei Manager/Agent-Modellen überwiegend *a priori* festgelegt und damit statisch sind, während dies bei kooperativen, „Peer-to-Peer“-Beziehungen dynamisch geschieht. Hinsichtlich der Kooperation des OSI- sowie des Internet-Managements mit CORBA ergibt sich aus diesem Grund keine Notwendigkeit, spezielle Verfahren zur Umsetzung der Organisationsmodelle zu entwickeln.

4.4.3 Überführung der Informationsmodelle

Wie in Abschnitt 2.1 ausgeführt wurde, stellt das Informationsmodell einer Managementarchitektur den Beschreibungsrahmen für Managementinformation und damit die Grundlage für die Verständigung von Manager und Agent bereit. Es beschreibt jede zu verwal-

tende Ressource mit allen benötigten Eigenschaften eindeutig. Folglich kann ein Managementsystem nur diejenigen Ressourcen überwachen und steuern, deren Beschaffenheit im Informationsmodell *seiner* Managementarchitektur beschrieben sind. Wie in Abschnitt 4.1 dargelegt wurde, ist die Annahme, daß sowohl Manager- als auch Agentensysteme derselben Managementarchitektur angehören, im Zeichen heterogener Managementarchitekturen nicht mehr haltbar. Die Sicherstellung der Umsetzbarkeit der Informationsmodelle ist somit die Basis für sämtliche Versuche, Übergänge zwischen unterschiedlichen Managementarchitekturen zu schaffen.

Natürlich ist diese Problematik bereits seit der Verfügbarkeit der OSI- und Internet-Managementarchitekturen bekannt; Konzepte, unterschiedliche Informationsmodelle zu integrieren, sind daher bereits seit der ersten Hälfte der neunziger Jahre verfügbar (vgl. dazu [KaSe 93], [Pavl 93], [NMF 114]). Mit dem Aufkommen weiterer, neuer Architekturen erhält diese Problematik jedoch eine neue Dimension, da theoretisch jede Quellarchitektur auf eine Zielarchitektur abgebildet werden müßte, wie es Abbildung 4.12 verdeutlicht.

Ursprungsarchitektur \ Zielarchitektur	OSI	Internet	CORBA
OSI	✕	**IIMC**	**JIDM**
Internet	**IIMC**	✕	**JIDM**
CORBA	**JIDM**	✕	✕

Abbildung 4.12: Vorhandene Algorithmen zur Umsetzung der Informationsmodelle

Die nachfolgenden Abschnitte gehen zuerst auf prinzipielle Überlegungen zur Umsetzung von Informationsmodellen unterschiedlicher Architekturen ein. Um die Lösungswege zu verdeutlichen, werden anschließend die Verfahren anhand von zwei konkreten Beispielen dargestellt: Zuerst wird ein im Rahmen der **ISO-Internet Management Coexistence (IIMC)**-Initiative des Network Management Forums standardisierter Algorithmus zur Überführung von Internet-SMI in das OSI-Informationsmodell vorgestellt; anschließend gehen wir auf ein in der Standardisierung befindliches und unter dem Namen **Joint Inter-Domain Management (JIDM)** bekanntgewordenes Verfahren zur Transformation von Internet-MIBs in CORBA-konforme Objektbeschreibungen ein. Abbildung 4.12 ordnet die jeweils behandelten Interoperabilitätsszenarien diesen beiden Gruppen zu. Auffällig ist hierbei, daß es bisher kein Verfahren gibt, welches die Integration von CORBA-Agenten in das Internet-Management gestattet. Die Begründung hierfür lautet,

daß ein solches Interoperabilitätsszenario in der Praxis kaum vorkommt und der geringe Nutzen eines solchen Verfahrens den Entwicklungsaufwand nicht rechtfertigt.

Beispiel 1: Abbildung von Internet-SMI in das OSI-Informationsmodell

Die großen Unterschiede in den Informationsmodellen der OSI- bzw. Internet-Architektur zeigen[6], daß das Aufgabenspektrum eines Management-Gateways weit über die bloße Umsetzung von Protokolldateneinheiten unterschiedlicher Managementprotolle hinausgeht: Schließlich soll (wie oben gefordert wurde) ein CMIP/SNMP-Gateway einem OSI-Manager den Zugriff auf SNMP-Ressourcen *auf vollkommen transparente Weise* ermöglichen, d.h. das Managementsystem greift auf SNMP-fähige Ressourcen auf dieselbe Art zu, wie dies beim Abrufen von Informationen aus OSI-Agenten der Fall wäre. Folglich ist es Aufgabe des Gateways, eine OSI-Sicht auf SNMP-basierte Ressourcen herzustellen. Die Abbildung von Internet-SMI in das OSI-Informationsmodell ist hierzu unerläßlich.

Das Ergebnis dieser Abbildung ist eine in GDMO übersetzte Internet-MIB, d.h. sämtliche Internet-MOs werden in OSI-MOs überführt. Diese neuen OSI-MOs haben die Aufgabe, dem Managementsystem gegenüber in der Rolle als Proxy-Objekte für die SNMP-Ressourcen zu agieren. Damit wird dem Managementsystem durch das Gateway eine OSI-Sicht auf die Internet-Managementinformation bereitgestellt.

Bei der Abbildung der Informationsmodelle gibt es zwei prinzipielle Designvarianten, die wir in den folgenden beiden Abschnitten darstellen und bewerten.

1. Alternative: Direkte Übersetzung mit dem IIMC-Algorithmus

Die sogenannte *direkte Übersetzung* transformiert Internet-MIBs *unmittelbar* in neue GDMO Klassen, wobei folgende einfache Regeln gelten:

- Gruppen einer SNMP-MIB werden zu OSI-MOCs. So werden z.B. die Gruppen interfaces, ip zu entsprechenden gleichlautenden MOCs. Im Fall der system-Gruppe der MIB-II tritt jdeoch das Problem auf, daß eine gleichlautende Umsetzung zu einem Namenskonflikt führt, da die Wurzel des OSI Management Information Tree denselben Namen trägt. Folglich ist für diesen Sonderfall eine geeignete Umbenennung vorzusehen. In unserer Implementierung haben wir für die GDMO-Darstellung der MIB-II den Namen internetSystem eingeführt.
- Tabellenzeilen in Internet-SMI werden ebenfalls auf OSI-MOCs abgebildet. Die Idee beruht auf der Tatsache, daß SNMP-Tabellen grundsätzlich dann gebildet werden, wenn zu einer einer Ressourcenklasse mehrere Instanzen existieren; somit wird beispielsweise die Tabellenzeile ipAddrEntry zu einer gleichlautenden OSI-MOC.
- Alle anderen skalaren MIB-Variablen werden zu Attributen der jeweiligen OSI-MOCs. Die SNMP-Variable sysDescription aus der Gruppe system wird folglich

[6] vgl. hierzu die Ausführungen in den Abschnitten 3.1.1 bzw. 3.1.3

zu einem Attribut der MOC internetSystem, die aus der Gruppe system entstanden ist.

- Die Object Identifier (OID), die eine Internet-MIB-Variable eindeutig referenzieren, werden zum Suffix des Knotens im OSI-Registrierungsbaum, der das OSI-Äquivalent zur Internet-MIB darstellt.
- Die Einträge des *Description* Feldes jeder MIB-Variable werden in das *Behaviour* Feld des korrespondierenden OSI-Attributes eingetragen, da in beiden Fällen die Semantik von Managementobjekten lediglich in natürlicher Sprache definiert ist.

Aus Platzgründen ist es an dieser Stelle nicht möglich, die Details des IIMC-Algorithmus zur Überführung von Internet-MIBs in OSI-MOCs in voller Länge darzustellen. Der interessierte Leser findet diese in Anhang B.1. Der Algorithmus wird dort anhand der MIB-II [rfc1213] näher erläutert.

Es ist offensichtlich, daß die direkte Übersetzung keine bereits in existierenden MOCs definierten Parameter und Methoden durch Vererbung übernehmen kann, da jegliche in den SNMP-MIBs enthaltene Managementinformation in neue, unmittelbar von „top“ abgeleitete Klassen überführt wird. Die Vererbungshierarchie bleibt somit flach und der Grad an generischer Managementinformation entsprechend gering. Vorteilhaft ist, daß die Übersetzung nach einem vom NM-Forum standardisierten Algorithmus [NMF 26] erfolgt. Sie ist somit deterministisch und daher automatisierbar. In der Tat sind die in Abschnitt B.1 detailliert vorgestellten Beispiele durch einen Compiler [Reil 93] erzeugt worden, dessen momentaner Reifezustand allerdings manuelle Eingriffe durch den Entwickler in den erzeugten Code erforderte. Eine detaillierte Beschreibung dieser Problematik ist in [NMF 29] enthalten.

2. Alternative: Abstrakte Übersetzung:

Die *abstrakte Übersetzung* zielt demgegenüber darauf ab, die Semantik einer Internet-MIB möglichst weitgehend zu erhalten. Dies geschieht durch Abbildung der Elemente einer Internet-MIB auf diejenigen OSI-MOCs, die durch geeignete Vererbung aus bereits existierenden GDMO-Klassen entstehen.

Als Basis der Vererbungshierarchie können beispielsweise die **Generic Managed Object Classes (GMOC)** [ISO 10165-5] für die Modellierung der Internet-Ressourcen benutzt werden. Damit können SNMP-fähige Ressourcen *vollständig*, d.h. auch in Bezug auf Vererbungseigenschaften in das OSI-Management eingebunden werden. Ein solcher Ansatz findet sich in [AbCH 93]. Der Einsatz von GMOCs wirft jedoch eine Reihe von Problemen auf, die nicht zuletzt daraus resultieren, daß innerhalb des OSI-Informationsmodells lediglich eine kleine Anzahl generischer und sehr abstrakter MOCs definiert sind. Obwohl es Versuche gab (z.B. [Beie 93]), Teile der MIB-II durch abstrakte Übersetzung in die Vererbungshierarchie des OSI-Informationsmodells auf Grundlage der GMOCs einzubinden, hat sich gezeigt, daß diese Art der Übersetzung von Internet-MIBs

vor allem für Ressourcen, welche nicht zum OSI-Schichtenmodell konform sind, problematisch ist. Dies äußert sich nicht zuletzt darin, daß sämtliche GMOCs ausschließlich die Elemente des OSI-Referenzmodells beinhalten (z.B. Dienstzugangspunkt, verbindungsorientierte Protokollmaschine usw.) und damit keinerlei Anknüpfungspunkte für generische MOCs bieten, welche die Komponenten einer verteilten Anwendung darstellen. Auch die gegenwärtig in Bearbeitung befindlichen MOCs für Protokollmaschinen des Systems Management Protocols [ISO 10165-9] ändern an dieser Situation nichts Grundlegendes, sind jedoch ein erster Schritt in die Richtung eines OSI-basierten Anwendungsmanagements[7]. Das zentrale Problem liegt jedoch darin, daß die abstrakte Übersetzung nichtdeterministisch ist, da unterschiedliche Anordnungen der Vererbungshierarchie denkbar sind. Eine Automatisierung der abstrakten Übersetzung von Internet-MIBs in OSI-MOCs ist daher nicht machbar.

Wir haben uns daher aus diesen Gründen entschlossen, für die Entwicklung unseres Gateway-Prototypen die *Direkte Übersetzung* zu verwenden, um damit die Offenheit dieser Systeme zu garantieren.

Beispiel 2: Umsetzung von Internet-MIBs in das CORBA-Objektmodell: Der JIDM-Algorithmus

Die Abbildung bestehender Objektbeschreibungen in das CORBA-Objektmodell setzt analog zur Transformation von Internet-SMI nach GDMO die algorithmische Überführung der Spezifikationssprachen voraus, in denen die Managementobjekte beschrieben sind. Im vorliegenden Fall handelt es sich um die Übersetzung der in der Internet-Managementarchitektur verwendeten ASN.1-Templatesprache in OMG IDL. Hierfür existiert ein Algorithmus [JIDM 97], der im Rahmen der Aktivitäten der *Joint X/Open NM-Forum Inter-Domain Management Task Force (JIDM)* entworfen wurde und sich zur Zeit in der Standardisierungsphase befindet. Aufgrund der Automatisierbarkeit des Verfahrens der direkten Übersetzung wurde daher auch in diesem Fall diese Transformationsvariante gewählt. Überdies kennt CORBA bisher keine generischen MOCs, die als Basis einer Vererbungshierarchie dienen könnten.

Wichtigster Gesichtspunkt bei der Überführung von Agenten einer Managementarchitektur in eine andere ist die unterschiedliche Mächtigkeit der jeweiligen Informationsmodelle. Während das Internet-Informationsmodell sämtliche Aspekte eines Agenten (Parameter und Aktionen) in Form von skalaren Datentypen bzw. Tabellen beschreibt und – im Gegensatz zu CORBA – keine Mechanismen wie beispielsweise Vererbung oder Datenabstraktion kennt, werden im CORBA-Objektmodell Agenten in Form von Objektklassen sowie deren zugehörigen Attributen und Methoden definiert. Die Transformation der in SNMPv2 vorhandenen Datentypen, Makros und asynchronen Ereignismeldungen in die CORBA-Entsprechungen geschieht folgendermaßen:

[7] Der Entwurf einer solchen Vererbungshierarchie für verteilte kooperative Managementsysteme ist der Inhalt von Kapitel 5.

1. Jedes einzelne Internet SMI-Modul wird einem IDL-Modul zugeordnet. In diesem werden alle durch die Übersetzung entstandenen Schnittstellen, Typen und Konstanten abgelegt. Wenn es mindestens ein NOTIFICATION-TYPE-Makro im SNMP Informationsmodul gibt, werden zusätzlich zwei IDL-Schnittstellen SnmpNotifications und PullSnmpNotifications für die typisierte *push*- respektive *pull*- Event-Kommunikation erzeugt.
2. ASN.1-Typen werden auf IDL-Datentypen abgebildet; insbesondere werden die Typspezifikationen der TEXTUAL-CONVENTION-Makros zu IDL-Datentypen.
3. MODULE-IDENTITY-, OBJECT-IDENTITY- und OBJECT-TYPE-Makros werden auf IDL-Konstanten vom Typ ASN1_ObjectIdentifier abgebildet.
4. Für jedes Tabellenobjekt und für jede Gruppe des MIB-Moduls wird eine Schnittstelle definiert. Dabei wird
   - zu jedem Spalteneintrag einer Tabelle und
   - zu jeder MIB-Variable einer Gruppe

   ein IDL-Attribut in der entsprechenden Schnittstelle deklariert, wobei Identifikator, Typ und Zugriffsrechte der Attribute aus den OBJECT-TYPE Makros der dazugehörigen Variablen abgeleitet werden. Dieses Vorgehen ist identisch zur oben beschriebenen Erzeugung von GDMO-Templates.
5. Zu jedem NOTIFICATION-TYPE Makro werden ferner drei Funktionen deklariert:
   - im Interface SnmpNotifications die Funktion <notification_name>, wobei <notification_name> der Identifikator des NOTIFICATION-TYPE-Makros ist,
   - im Interface PullSnmpNotifications die beiden Funktionen try_<op> und pull_<op>. <op> ist der Name der korrespondierenden Funktion in SnmpNotifications. Diese sind notwendig, um sowohl das *Push*- als auch das *Pull*-Modell zur Zustellung asynchroner Ereignismeldungen zu unterstützen.

Auch hier haben wir aus Platzgründen die Anwendung dieses Algorithmus am Beispiel der MIB-II sowie die dazu notwendigen Erläuterungen in Anhang B.2 verlagert. Der JIDM-Algorithmus ist Bestandteil der OMG *CORBA/TMN Interworking*-Aktivitäten und befindet sich gegenwärtig in der Abschlußphase der Standardisierung durch die OMG (siehe dazu [OMG SNMP]).

4.4.4 Abbildung der Kommunikationsmodelle

Nachdem wir nun vorgestellt haben, wie die Informationsmodelle der einzelnen Architekturen ineinander überführt werden können, stellt sich nun die Frage, wie diese Managementinformation ausgetauscht werden kann. Hierzu müssen die Kommunikationsmodelle geeignet aufeinander abgebildet werden. Das Management von SNMP-Agenten durch

CORBA- bzw. OSI-konforme Managementsysteme ist erst dann machbar, wenn **beide** Abbildungen erfolgt sind, d.h. sowohl die Managementinformation als auch die Kommunikationsmechanismen.

Zur Erreichung einer möglichst umfassenden Integration sind zwei grundlegende Designalternativen möglich:

Zustandsloses (stateless) Gateway

Der Grundgedanke eines zustandlosen Gateways besteht darin, daß Operationen auf Proxy-Objekten *unmittelbar* an die entsprechenden MOs der Agenten weitergeleitet werden. Hierzu muß das Gateway gegebene OSI- bzw. CORBA-MOIs auf die dazugehörigen SNMP-OIDs abbilden, die Art der Operation umsetzen und die SNMP-PDU generieren. Im Gateway müssen somit die Definitionen der Objekt*klassen* und der vorhandenen Instanzen vorliegen; es erfogt jedoch keinerlei (Zwischen-) Speicherung von Informationen bezüglich der aktuellen *Wertebelegung* von MO-Instanzen. Das Gateway agiert somit im wesentlichen in der Rolle eines „PDU-Konverters“. Ein Request vom Manager an das Gateway löst in jedem Fall einen Request vom Gateway an den ausgewählten Agenten aus. Die hierbei ermittelten Informationen sind in jedem Fall aktuell, auch wenn diese Abfrage eine gewisse Bearbeitungszeit erfordert und einen erhöhten Netzverkehr mit sich bringt, da die Belegungen jeder betroffenen MIB-Variable einzeln ermittelt werden.

Zustandsbehaftetes (stateful) Gateway

Der Ansatz eines zustandsbehafteten Gateways setzt hier an und zielt auf eine Optimierung des Gesamtsystems Manager/Gateway/Agent hinsichtlich Antwortzeit und Netzverkehr ab: Erreicht wird dies durch die Speicherung von *Instanzeninformationen* bereits im Gateway. Aufgrund der Tatsache, daß das Gateway Caching von MO-Instanzeninformation durchführt, löst ein Request des Managers auf die Proxy-Objekte des Gateways nicht in jedem Fall auch eine Anfrage vom Gateway an das Agentensystem aus. Hierdurch ergeben sich offensichtlich Vorteile in bezug auf die Antwortzeit und den Netzverkehr, da nur dann aktuelle Werte vom Agentensystem angefordert werden, wenn dies „notwendig“ erscheint.

Das grundlegende Problem liegt nun darin, zu entscheiden, wann der „richtige“ Zeitpunkt ist, um eine Auffrischung der gespeicherten Daten vorzunehmen, da die Dynamik der MIB-Attribute bekanntlich starken Schwankungen unterliegt: So liegt die Änderungsfrequenz des Namens bzw. der Adresse eines Endsystems nahezu bei Null während die Wertebelegung eines Zählers für eingegangene PDUs sich kontinuierlich ändert. Eine Möglichkeit zur Lösung dieses Problems besteht nun darin, bereits in der MIB für jedes Attribut Informationen bezüglich seiner Dynamik festzuhalten und die MIB-Beschreibung entsprechend zu erweitern. Ein solcher Ansatz liegt dem in [HAB 91] entwickelten Konzept zugrunde. Das unmittelbare Problem eines solchen Ansatzes besteht nun darin, daß eine solche Erweiterung zwangsläufig außerhalb der gegenwärtig standardisierten MIB-Definitionen liegt, heutige MIB-Compiler solche Definitionen nicht akzeptieren und somit

diese erweiterten MIB-Beschreibungen in kein gegenwärtiges Managementsystem übernommen werden können. Ferner spiegelt dieser Ansatz eine „Bottom-up"-Sichtweise wider, da die Information bezüglich der Dynamik dieser Werte ausschließlich vom Hersteller einer Ressource festgelegt werden können. Dies widerspricht jedoch dem Ziel eines *betreibergerechten* Managements, das idealerweise einem Netzadministrator erlauben würde, die Abfragefrequenz für einzelne Attribute seinem Bedarf entsprechend zu konfigurieren. Dieser Bedarf hängt wiederum stark von den Zielvorgaben des Unternehmens ab, welche sich jedoch bisher nicht automatisch auf einzelne MIB-Attribute abbilden lassen. Bei einem Umfang von mehreren hundert MIB-Variablen pro Endsystem stellt jedoch die manuelle Konfiguration einzelner Attribute ein aus Skalierbarkeitsüberlegungen kritisches Problem dar.

Eine andere Sichtweise verfolgen die in [DiBo 97] und [DivB 97] beschriebenen neuen Ansätze, die auf eine automatische Klassifikation der MIB-Attribute in Abhängigkeit ihrer Aktualität abzielen. Hierbei wird anhand der Änderungshistorie von Werten empirisch auf die zukünftige Änderungsfrequenz geschlossen, wobei jedoch die Gefahr besteht, daß im Falle einer plötzlich auftretenden Fehlersituation nicht rechtzeitig reagiert werden kann, weil die Abfrage des entsprechenden Attributs mangels vorheriger Dynamik zum notwendigen Zeitpunkt unterbleibt. Aufgrund des geringen Alters dieser Algorithmen und dem damit einhergehenden Mangel an prototypisch ermittelten Erfahrungswerten muß die Praktikabilität dieses Ansatzes gegenwärtig unbeantwortet bleiben. Nicht zuletzt erfordert die Ermittlung der Dynamik von MIB-Attributen die Bildung von Zeitreihenanalysen, was wiederum eine hohe Verarbeitungslast auf dem Gateway impliziert.

Wir haben uns bei unseren Prototypen für einen dritten Weg entschieden, der keine Modifikationen an den MIBs erfordert, einen relativ guten Kompromiß zwischen Verarbeitungslast und Netzverkehr darstellt und überdies verhältnismäßig einfach zu implementieren ist. Grundlage unserer Überlegungen ist eine Heuristik, nach der die Pollingfrequenz anhand des Datentyps eines MIB-Attributes festgelegt wird. Werte zu Datentypen, welche (sehr dynamische) Zähler repräsentieren, werden nicht auf dem Gateway zwischengespeichert, Strings erhalten einen relativ langen Abfragezeitraum zugewiesen, Integer-wertige Attribute liegen in der Mitte. Darüber hinaus sind die datentypbezogenen Polling-Zeiträume vom Benutzer konfigurierbar, d.h. das Gateway selbst besitzt eine MIB, mit der es konfiguriert wird. Wir werden gegen Ende dieses Kapitels auf die Konsequenzen der Gateway-Instrumentierung zurückkommen.

Implementierungsbeispiel 1: CMIP/SNMP Gateway

Aufgabe eines CMIP/SNMP Gateways ist es, einem OSI-konformen Managementsystem zu ermöglichen, mit dem *vollständigen* CMIS-Funktionsumfang auf SNMP-Agenten zugreifen zu können. Dies erfordert erstens eine Umsetzung der Managementoperationen und zweitens eine Abbildung der auszutauschenden Managementinformationen. Für den Austausch von Managementoperationen zwischen diesen beiden Architekturen bedarf es

einer Protokollkonvertierung, die vom Management-Gateway[8] durchzuführen ist.

Prinzipieller Aufbau eines CMIP/SNMP Gateways

Die aus [NMF 28] übernommene Abbildung 4.13 illustriert die Doppelrolle des Gateways, das gegenüber dem OSI-Manager eine Agentenrolle und gegenüber dem SNMP-Agenten einen Managerrolle einnimmt. Die Bezeichnungen der Schnittstellen und Referenzpunkte zwischen den einzelnen Komponenten folgen den Festlegungen des TMN-Referenzmodells [ITU M.3010].

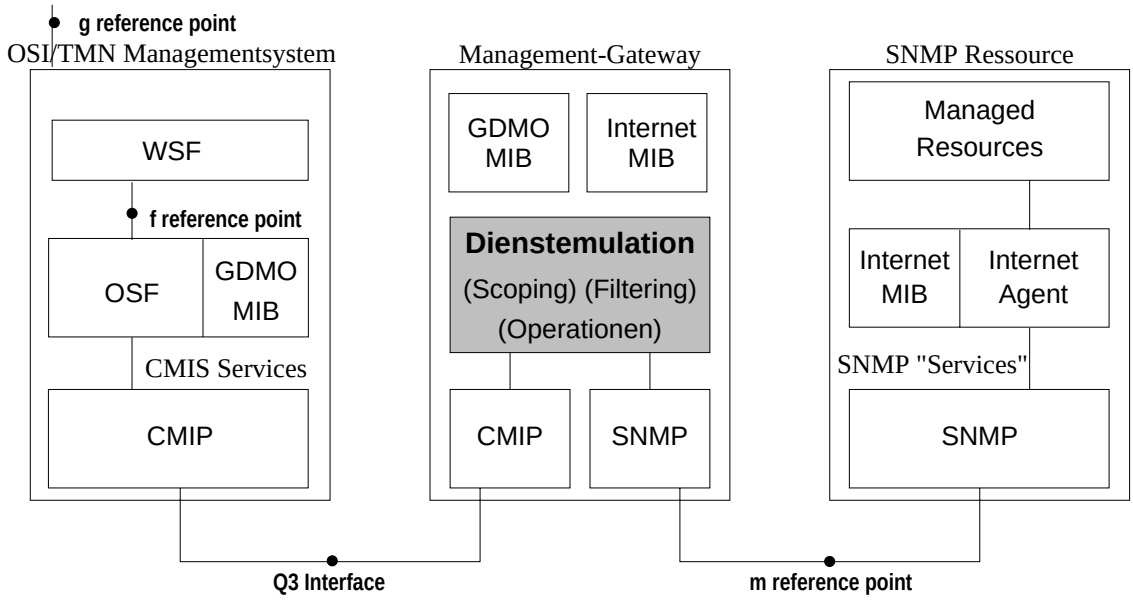


Abbildung 4.13: Prinzipieller Aufbau eines CMIP/SNMP Management-Gateways

Die durch das Gateway erfüllten Aufgaben lassen sich in zwei zentrale Kategorien unterteilen:

- **Name Mapping** bezeichnet die Aufgabe, nach Erhalt einer CMIS-Anfrage auf eine bestimmte MOI den korrespondierenden *Object Identifier* zu berechnen. Dies ist notwendig, da die Zugehörigkeit der Ressource zum Internet-Management für das Managementsystem transparent ist.
- Mit **Service Mapping** wird die Transformation von CMIS-Diensten (die Scoping und Filtering beinhalten können) in entsprechende SNMP-PDUs bezeichnet. Nach Eintreffen eines CMIS-Requests müssen diejenigen MOIs identifiziert werden, die

[8] In der TMN-Terminologie wird ein Gateway, das einem Managementsystem (OSF) gestattet, Ressourcen anderer Protokollwelten zu überwachen und zu steuern, häufig *Q-Adapter* genannt (vgl. dazu die Ausführungen zu TMN in Abschnitt 3.1.1.

im Scope des Requests liegen und auf die folglich die Operation angewandt werden muß. Daraufhin muß geprüft werden, ob deren Attribute die Filterkriterien erfüllen. Ist dies der Fall, wird die Operation auf den betroffenen MOIs ausgeführt. Tabelle 4.1 gibt einen Überblick, wie die CMIS-Operationen auf SNMP-PDUs durch das Gateway abgebildet werden.

Asynchrone Ereignismeldungen, die von SNMP-Agenten an das Gateway gesandt werden, müssen durch das Gateway den Proxy-Objekten zugeordnet und durch diese an das Managementsystem gesandt werden. CMIS-Operationen wie M-CREATE und M-DELETE sind bekanntlich nur auf mehrfach instantiierbaren Objekten ausführbar (in SNMP sind dies Tabellen) und werden dort durch das Ändern der sogenannten *Row Status* Variable realisiert. M-CANCEL-GET ist die einzige CMIS-Operation, die keine Entsprechung in SNMP hat; ihre Funktionalität muß daher mit anderen Mitteln nachgebildet werden, beispielsweise durch Implementierung einer *Abort*-Operation im Gateway, das daraufhin die Verarbeitung des Requests abbricht und die bisher ermittelten Ergebnisse verwirft.

CMIS-Dienst	SNMP-PDU
M-GET	SNMP-GET
M-SET	SNMP-SET bzw. SNMP-GET
M-CREATE	SNMP-SET
M-DELETE	SNMP-SET
M-CANCEL-GET	–
M-EVENT-REPORT	SNMP-TRAP

Tabelle 4.1: Abbildung der CMIS-Operationen auf SNMP-PDUs

Architektur des CMIP/SNMP Gateway-Prototyps

Die in den vorangehenden Abschnitten identifizierte Doppelrolle eines CMIP/SNMP Management-Gateways (sowohl OSI/TMN-Agent als auch SNMP-Manager) impliziert, daß wir uns zur Implementierung unseres Prototypen auf Entwicklungssysteme für OSI/TMN-Agenten abstützen können. Wir müssen nunmehr dafür sorgen, den OSI/TMN-Agenten um Funktionen für das oben angesprochene *Name*- und *Service Mapping* zu erweitern und eine SNMP-Programmierschnittstelle zum Auslösen der entsprechenden Protokolloperationen sowie zur Entgegennahme von SNMP-Traps zu integrieren.

Unsere Wahl der Entwicklungsumgebung fiel auf die *IBM TMN Products* ([IBMTMNGI 96], [FeHN 96]), da dieses System einige für unsere Aufgabenstellung wichtige Merkmale aufweist: Die *NetView TMN Support Facility* [IBMTMNSF 96] ist eine vollständige OSI/TMN-Managementplattform, d.h. sie beinhaltet sowohl eine OSF als auch eine WSF und stellt dem Entwickler Programmierschnittstellen zur Entwicklung

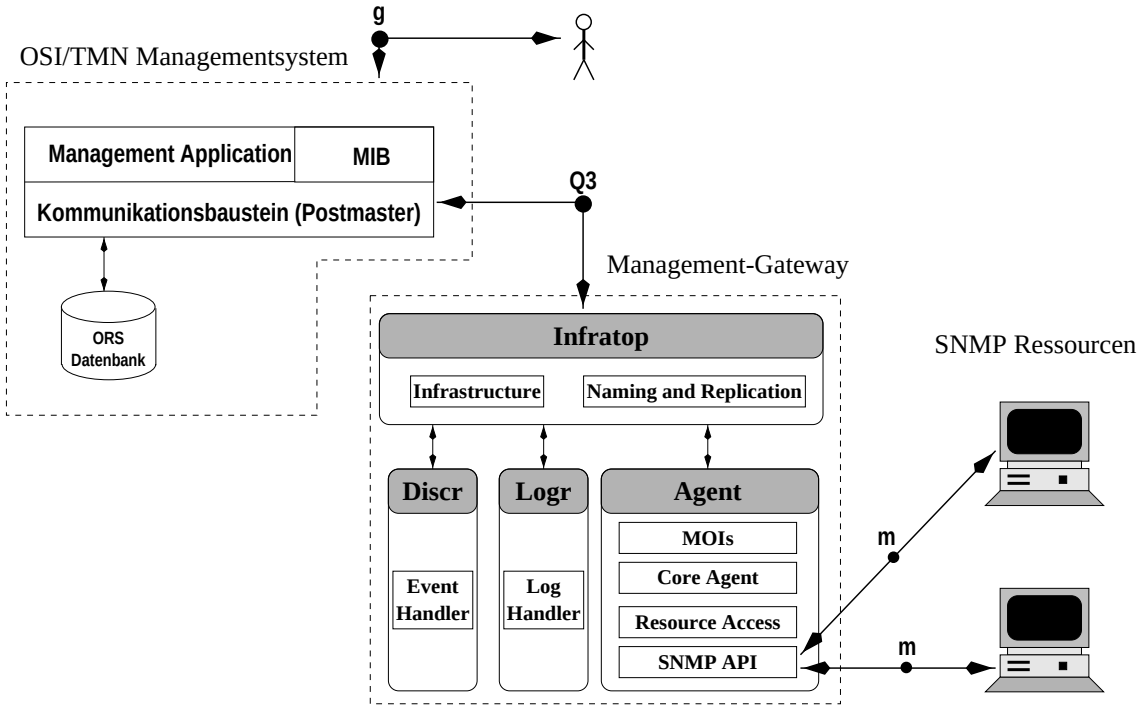


Abbildung 4.14: Architektur des CMIP/SNMP Gateways

von TMN-Applikationen bereit. Das Produkt unterstützt die generischen Objektkataloge M.3100 [ITU M.3100] und OMNIPoint und stellt ein Rahmenwerk zur Entwicklung OSI/TMN-konformer Managemenetagenten bereit. Die eigentliche Entwicklungsumgebung wird unter dem Namen *TMN WorkBench for AIX* [IBMTMNWB 96] vermarktet. Sie besteht aus Werkzeugen zur Modellierung von Managementinformation wie MOC-Browsern und -Editoren mit deren Hilfe neue Objektklassen werkzeugunterstützt von bestehenden MOCs abgeleitet werden können. Zu diesem Zweck steht eine Vielzahl an Basis-MOCs in maschinenlesbarer Form bereit. OSI/TMN-konforme Agenten werden von der Entwicklungsumgebung aus GDMO/ASN.1-Objektbeschreibungen in C++ Programmrümpfe übersetzt, die dann in Form sog. *Callback*-Methoden vorliegen. Letztere beinhalten die eigentliche Managementinformation, d.h. die konkrete Managementinstrumentierung. Diese Methoden werden genau dann vom Agentensystem aufgerufen, wenn Operationen auf den dazugehörigen MIB-Variablen erfolgen.

Der wesentliche Vorteil der IBM TMN Entwicklungsumgebung besteht nun darin, diese *Callback*-Methoden unter Zuhilfenahme der in der TMN/C++ API standardisierten Abbildungsregeln automatisch zu erzeugen, um so den Entwickler vollständig von den CMIS-Spezifika abzuschirmen. Die eigentliche Kommunikation auf der Basis des Protokolls CMIP geschieht vollkommen transparent für den Entwickler, was gegenüber der Alternative XMP/XOM eine signifikante Vereinfachung bedeutet. Die Nutzung dieser Vorteile durch die Abstützung auf die IBM-Entwicklungsumgebung legt jedoch ebenfalls die pro-

grammtechnische Architektur unseres Gateways fest, wie sie in Abbildung 4.14 dargestellt ist.

Nach seiner Instantiierung besteht das Gateway aus vier unterschiedlichen Prozessen:

- Der **Infratop**-Prozeß setzt sich aus zwei Komponenten zusammen, die als *Infrastructure* bzw. *Naming and Replication* bezeichnet werden. Dieser Prozeß stellt den Kern eines Agenten dar, da er sowohl CMISE als auch ACSE realisiert und darüber hinaus auch den Management Information Tree verwaltet sowie die Gültigkeit von Name Bindings für M-CREATE bzw. M-DELETE Anfragen überprüft. Ferner ist der Infratop für die Verarbeitung von Scopes und Filtern verantwortlich, die im wesentlichen darin besteht, eine Anfrage an alle im Scope liegenden MOIs zu replizieren. Während dies einerseits eine wesentliche Verringerung des Entwicklungsaufwandes bedeutet, werden wir im folgenden sehen, daß wir uns damit auch eine signifikante Einschränkung bezüglich der dynamischen Erweiterbarkeit unseres Gateways einhandeln.
- **Discr** ist die zur OSI *Event Report Management Function* [ISO 10165-5] konforme Komponente, die eine flexible Ereignisfilterung und -weiterleitung auf der Basis von EFDs zuläßt.
- **Logr** stellt als Implementierung der OSI *Log Control Function* [ISO 10165-6] Dienste zur persistenten Speicherung von Ereignismeldungen bereit.
- Der **Agent** schließlich beinhaltet die MOIs sowie die Dienste zum Zugriff auf die konkrete Ressourceninstrumentierung. Wir haben hier eine SNMP-Programmierschnittstelle eingebunden, um so die entsprechenden PDUs erzeugen zu können.

Funktionsweise des Gateways

Die Funktionsweise des Gateway-Prototyps ist nachfolgend am Beispiel eines *scoped* und *filtered* M-GET.request dargestellt (siehe auch Abbildung 4.15):

1. Vom OSI-Managementsystem wird durch den CMIS/ACSE-Stack ein M-GET.request mit den gewünschten Scope-, Filter- und Synchronisationsparametern auf eine Basis-MOC abgesendet.
2. Das Gateway empfängt diese PDU als M-GET.indication, extrahiert die Parameter und gibt diese an die *Naming and Replication*-Komponente weiter.
3. Dort werden die Scopes, Filter und Synchronisationsbedingungen ausgewertet. Hier geschieht die Prüfung, ob eine Instanz der ausgewählten Basis-MOC überhaupt vorhanden ist.
4. Ist dies nicht der Fall, wird eine M-GET.response, die die Fehlermeldung *noSuchInstance* enthält, an das Managementsystem zurückgeschickt.

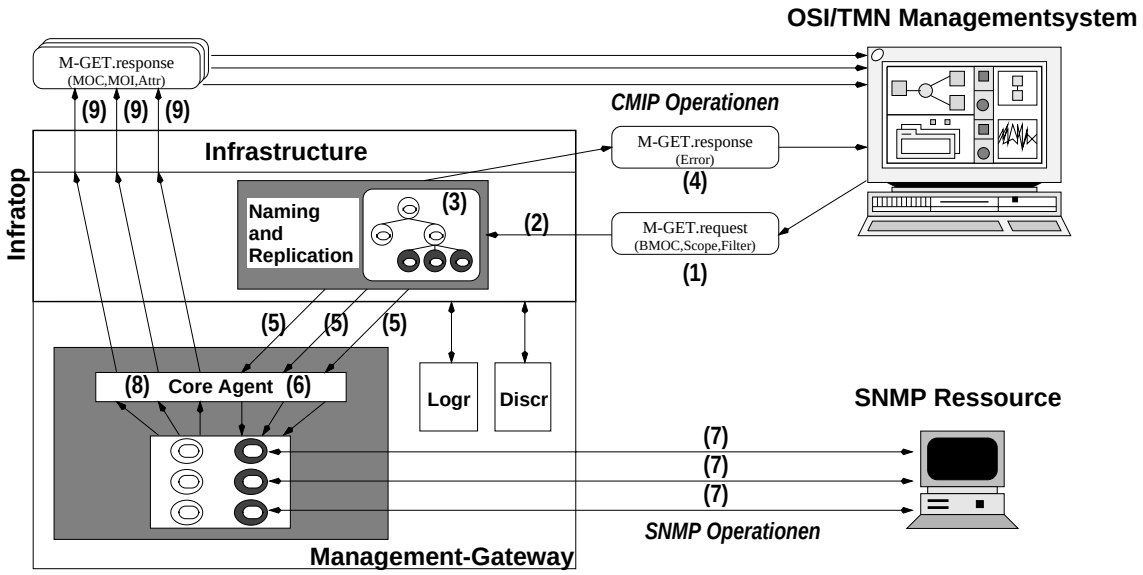


Abbildung 4.15: Ablauf eines M.GET-requests mit Scoping und Filtering

5. Existiert eine Instanz der Basis-MOC, werden die Scopeparameter dahingehend ausgewertet, daß jede im Scope befindliche Instanz einer MOC eine Kopie der GET-Operation zugestellt bekommt. Dies geschieht durch Übergabe der replizierten Operation an den Core Agent.

6. Dieser stößt die Bearbeitung der Operation auf den entsprechenden MO-Instanzen an, indem die Callback-Methoden der jeweiligen Attribute getriggert werden.

7. Diese Callback-Methoden enthalten Funktionen zur Bestimmung der SNMP-OID der jeweiligen Attribute. Anschließend erfolgt für die betroffenen MIB-Variablen jeweils eine SNMP-GET Anfrage, deren Antwort die Wertebelegungen der betrachteten MIB-Variablen enthält.

8. Der Core Agent nimmt die Ergebnisse entgegen und gibt diese seinerseits an die Protokollinfrastruktur des Gateways weiter.

9. Die Infrastruktur verpackt die Ergebnisse in M-GET.response PDUs und schickt diese in Form von *linked replies* an das Managementsystem zurück, wo sie weiterverarbeitet werden können.

Von SNMP-Agenten ausgesandte Traps müssen vom Gateway in OSI-Notifikationen übersetzt und zu den entsprechenden EFDs weitergeleitet werden. Diese übernehmen schließlich den Transport zu den jeweiligen OSI-Managern.

Wir haben zur Verdeutlichung des Vorgangs in Abbildung 4.16 einen Ausschnitt der Operatorkonsole des OSI-Managementsystems dargestellt, in dem sich unterhalb der Knoten TelcoNet (das gesamte Kommunikationssystem) und TelcoSys (die über das Gateway administrierte SNMP-Domäne) die eigentlichen Ressourcen wiederfinden, die in unserem

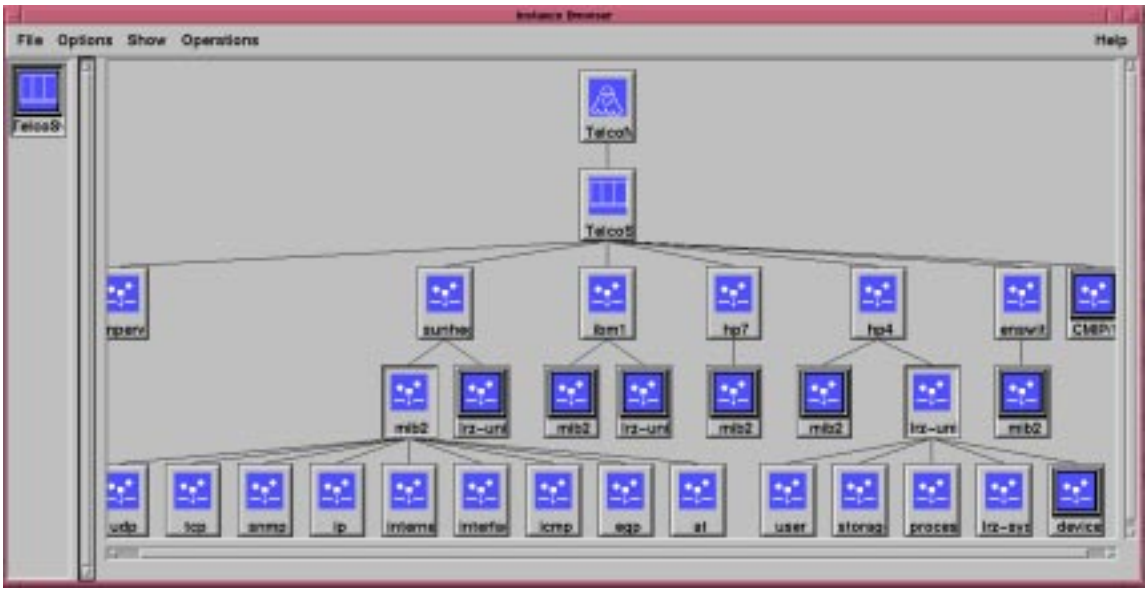


Abbildung 4.16: Management Information Tree des Gateway-Prototypen

Fall vier Workstations (sunhegering2, ibm1, hp7, hp4) und einen Ethernet-Switch (enswitch1) umfassen. Die Workstations verfügen jeweils über die MIB-II (mib2) und eine weitere von uns entwickelte UNIX Workstation MIB (lrz-unix), die in Abschnitt 5.1.2 vorgestellt wird. Die Bestandteile der einzelnen MIBs (9 Gruppen der MIB-II, 5 Gruppen der Unix Workstation MIB) ergeben sich aus der untersten Zeile der Darstellung. Um die Abbildung übersichtlich zu halten, haben wir die lediglich die MIB-II des Systems sunhegering2 sowie die UNIX Workstation MIB des Systems hp4 im Detail ausgewählt. Eine expandierte Darstellung erreicht man durch Selektion des jeweiligen Symbols. Die aus Abschnitt 4.4.3 bekannte Abbildung von SNMP-Gruppen auf OSI-MOCs wird hier sichtbar. Ferner zeigt sich, daß die Gruppe system, die sowohl in der MIB-II als auch in der UNIX Workstation MIB existieren, geeignet umbenannt werden mußte (internetSystem bzw. lrz-system), um einen Namenskonflikt mit der Basis-MOC system des OSI-MIT zu vermeiden. Die vorher angesprochenen Parameter zur Konfiguration des Gateways (z.B. hinsichtlich der Pollingintervalle) werden über die MOI CMIP/SNMP dem Management zur Verfügung gestellt, die am rechten Rand der Abbildung sichtbar ist.

Das Auslösen einer Abfrage auf einem oder mehreren Managed Objects geschieht durch Selektion des entsprechenden Symbols und Vervollständigen der notwendigen Angaben in einem weiteren Fenster. Hierbei kann u.a. der Scope einer Operation angegeben werden sowie die für diese Operation geltenden Filterregeln. Außerdem kann eine Synchronisationsbedingung (atomic/best effort) angegeben werden. Schließlich werden diejenigen Attribute ausgewählt, deren Wertebelegung angezeigt werden soll. Innerhalb der Filterregeln können die Attribute mit Vergleichsoperatoren (bezüglich Zahlen und Strings) versehen werden; diese Terme lassen sich dann mit logischen Operatoren verknüpfen. Man erhält so

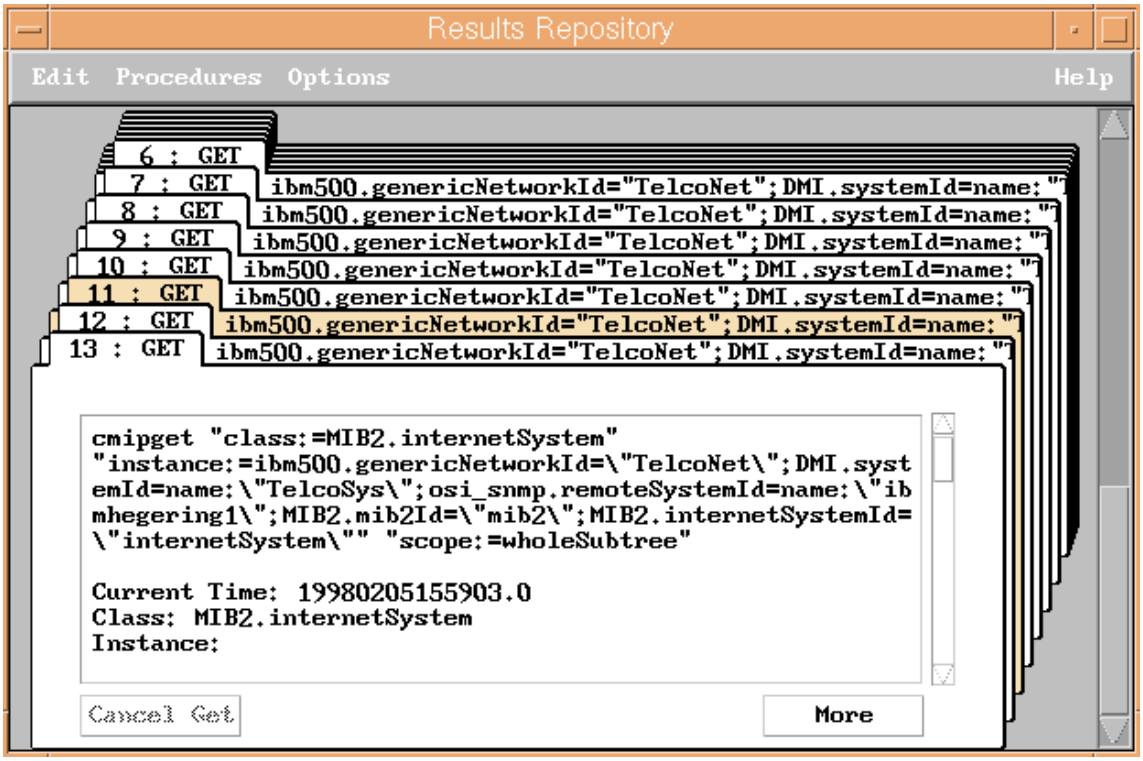


Abbildung 4.17: Anzeigefenster für Anfrage auf die MIB-II System-Gruppe

die Möglichkeit, zusammengesetzte Abfragen zu definieren, die starke Ähnlichkeit mit der Datenbank-Abfragesprache SQL haben. Sämtliche Abfragen lassen sich auch persistent speichern bzw. laden. Denkbare Abfragen wären zum Beispiel: „Ermittlung des Systemnamens aller Workstations, deren Root-Partition zu mehr als 80% belegt ist“ oder „Löschen aller Prozesse eines Benutzers, welche seit mehr als 24 Stunden laufen“. Solche zusammengesetzten Abfragen kommen der Ausdruckskraft von Management-Zielvorgaben bereits sehr nahe und bieten so den Netzadministratoren ausgezeichnete Überwachungs- und Eingriffsmöglichkeiten, die aufgrund der verteilten Auswertung auf den Agentensystemen ebenfalls ausgesprochen gut skalieren. Auf Seiten des Managementsystems werden somit nur die Abfragen formuliert und die Ergebnisse der Abfragen in Empfang genommen. Hierdurch besteht die Möglichkeit, auch mehrere unterschiedliche Abfragen parallel durchzuführen. Abbildung 4.17 stellt die graphische Anzeige einer Anfrage auf die system-Gruppe der MIB-II (hier aus den oben genannten Gründen als internetSystem bezeichnet) auf dem System *ibmhegering1* graphisch dar. Das daraufhin zurückgelieferte Ergebnis ist in Abbildung 4.18 dargestellt.

Ein deutliches Indiz für die Leistungsfähigkeit des Gateway-basierten Integrationsansatzes ist die nahtlose Integration des SNMP-Managements in OSI, da es unser Gateway-Prototyp ermöglicht, diese ursprünglich nur bei OSI verfügbaren Abfragen auch auf solchen Systemen auszuführen, die lediglich SNMP-Agenten besitzen. Es ist unmittelbar ein-

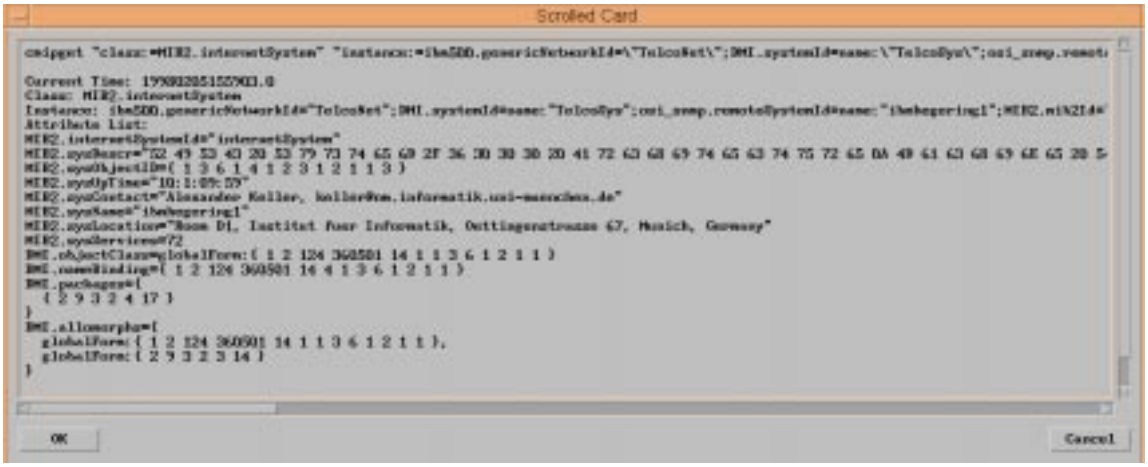


Abbildung 4.18: Ergebnis der Anfrage auf die MIB-II System-Gruppe

sichtig, daß solche Abfragen in einer reinen SNMP-Umgebung nicht möglich sind, da in SNMP-PDUs jeweils nur die Wertebelegungen konkreter Variablen abgefragt werden können und eine Filterung lediglich auf Seiten des Managementsystems möglich ist. Aus Skalierbarkeitsgründen ist dies in der Praxis oft undurchführbar.

Um eine Vorstellung der Antwortzeiten unseres Gateways zu vermitteln, werden wir nun einige in Abhängigkeit der Komplexität der Abfrage empirisch ermittelte Meßwerte vorstellen. Es sei jedoch an dieser Stelle betont, daß wir die reine Bearbeitungszeit durch das Gateway nicht ermitteln konnten, sondern uns vielmehr auf die (für den Benutzer sichtbare) gesamte Zeitspanne zwischen Anfragezeitpunkt und Ergebniszustellung konzentriert haben, die jedoch offensichtlich stark von der Ressourcenanzahl sowie der Komplexität der Abfrage abhängt.

- Für die in Abbildung 4.16 angegebenen 5 Ressourcen mit der MIB-II sowie der UNIX Workstation MIB benötigt die Erstellung des MIT beim Start des Managementsystems ca. 90 Sekunden.
- Die Abfrage der Wertebelegungen einer SNMP-Gruppe auf einem System (wie in Abbildung 4.17 dargestellt) variiert zwischen 1 Sekunde (für die system-Gruppe) und 15 Sekunden (Auslesen der aktuellen Prozeßliste).
- Die Abfrage einer MIB-Variable über alle 5 Systeme mit Scope und Filter benötigt zwischen 10 und 12 Sekunden.
- Komplexe Abfragen auf mehrere MIB-Variablen mit zusammengesetzten Filtern auf allen 5 Systemen dauert bestenfalls 30 Sekunden und im „worst case“ bis zu 3 Minuten.

Es ist deutlich zu erkennen, daß der hohe Komfort der OSI-Abfragen zu einem erhöhten Datenaufkommen zwischen Gateway und SNMP-Agenten führt, da die Verarbeitung und

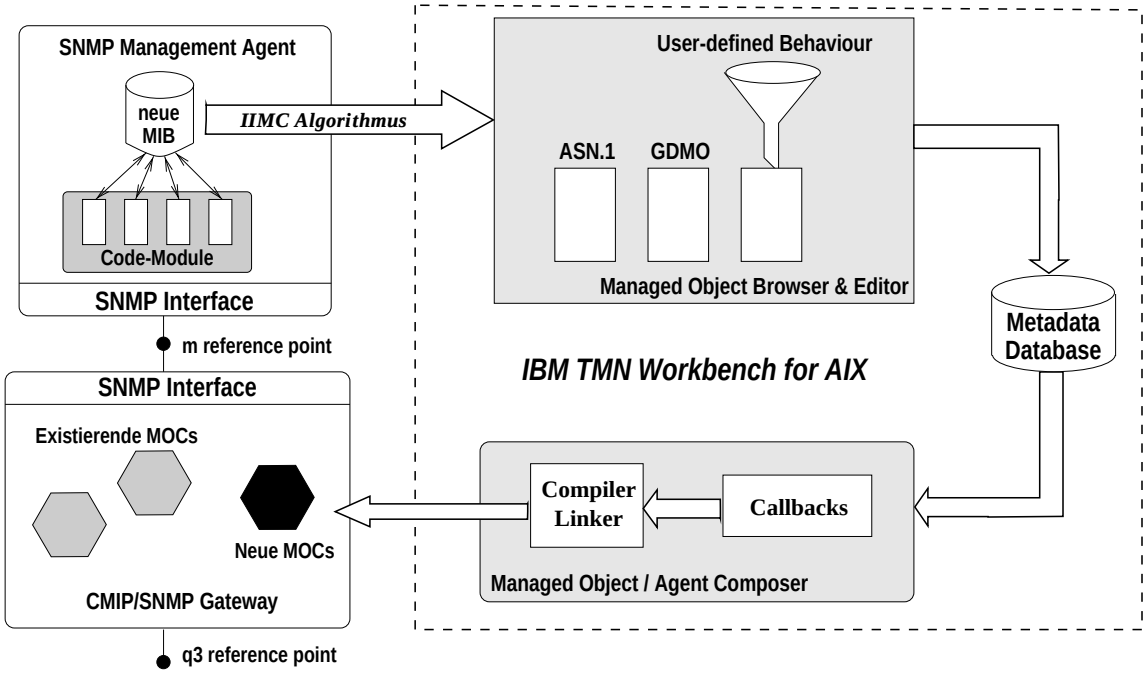


Abbildung 4.19: Integration von MIBs für neue Ressourcen in das Gateway

Auswertung der Anfragen vollständig durch das Gateway erfolgt.

Erweiterung des Gateways um neue Ressourcen

Wir werden uns nun der Problematik zuwenden, wie unser CMIP/SNMP-Gateway um neue Ressourcen-MIBs für SNMP-fähige Geräte erweitert werden kann.

Zuerst muß, wie in Abbildung 4.19 dargestellt ist, die MIB-Definition einer neuen Ressource mit Hilfe des IIMC-Algorithmus vom Internet-Informationsmodell in GDMO übersetzt werden. Anschließend erfolgt innerhalb der Entwicklungsumgebung durch das TMN/C++ API die Umsetzung der GDMO-Objektklassenbeschreibungen in C++ Callback-Klassen, deren Methoden vom Entwickler in entsprechende SNMP-Aufrufe umgesetzt werden müssen. Nachdem dies geschehen ist, wird das gesamte Gateway neu übersetzt und gebunden, was in unserem Falle der beiden unterstützten MIBs auf einer IBM RS/6000 Workstation (Modell 3BT) mit 128 MB Hauptspeicher ca. 60 Minuten dauert. Das Ergebnis ist ein CMIP/SNMP-Gateway, dessen Codegröße ohne Abstützung auf *shared libraries* knapp 40 Megabyte beträgt. Folglich ist die Erweiterung um neue MIBs gegenwärtig relativ aufwendig, da das vollständige Gateway für jede neu hinzukommende MIB neu übersetzt werden muß.

Implementierungsbeispiel 2: CORBA/SNMP Gateway

In Analogie zum CMIP/SNMP Gateway vermittelt ein CORBA/SNMP Gateway zwischen einem CORBA-konformen Managementsystem und mehreren SNMP-Agenten (siehe Ab-

bildung 4.20). In der CORBA-Architekturdomäne ist das Gateway eine Anwendung, die über einen ORB mittels CORBA-Requests von einem CORBA-Managementsystem angesprochen werden kann. Aus Sicht der SNMP-Agenten hingegen ist das Gateway ein SNMP-Manager. Die wesentlichen Aufgaben des Gateways liegen auch hier in der transparenten Weiterleitung von CORBA-Requests auf SNMP-PDUs sowie die Übermittlung der Ergebnisse. Eine Besonderheit der Kopplung von CORBA mit SNMP liegt darin, daß CORBA-Objekte bereits explizit vom Managementsystem adressiert werden und das Auslesen bzw. Setzen von MIB-Attributen durch Aufrufe der diesen Attributen zugeordneten get()- bzw. set()-Methoden geschieht. Ein anderes Spezifikum von CORBA ist, wie bereits vorher erwähnt, daß die Struktur von CORBA-Requests im Gegensatz zu den PDUs eines Managementprotokolls variabel ist. Eine zentrale Komponente zur Ermittlung der in Frage kommenden MOI sowie zur Weiterleitung der Anfragen[9] ist somit nicht erforderlich. Ein weiterer Aspekt von CORBA besteht darin, daß sich eine Anfrage grundsätzlich nur auf *eine* Objektinstanz bezieht; ein leistungsfähiger Scoping und Filtering-Mechanismus, wie ihn das OSI-Management kennt, fehlt hier. Somit entfällt auch die Möglichkeit der verteilten Auswertung von Anfragen bezüglich mehrerer MOIs auf den Agenten bzw. dem Gateway. Schließlich müssen SNMP-Traps vom Gateway angenommen und einem CORBA-basierten Managementsystem über die in Abschnitt 4.2.4 diskutierten *Event Channels* zugestellt werden.

Architektur des CORBA/SNMP Gateway-Prototyps

Aufgrund der Komplexität des Gesamtsystems liegt es nahe, das Gateway in verschiedene Komponenten zu zerlegen, die jeweils eine bestimmte Aufgabe erfüllen. Im einzelnen können folgende Komponenten identifiziert werden:

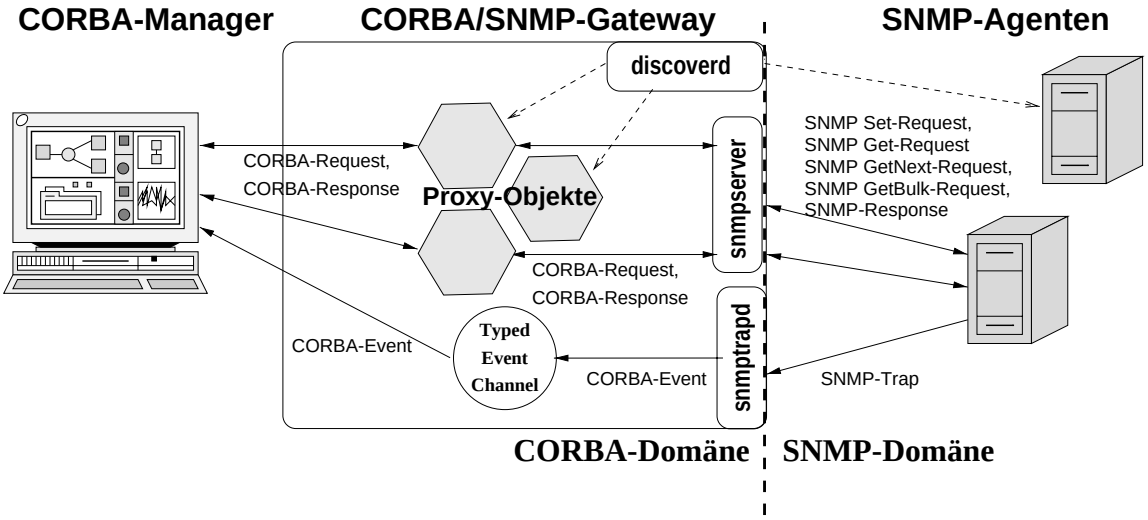


Abbildung 4.20: Architektur eines CORBA/SNMP Management-Gateways

[9]Dies war beim CMIP/SNMP-Gateway die *Naming and Replication* Komponente.

Proxy-Objekte sind die Instanzen der in IDL beschriebenen Objektklassen, die die SNMP-MIB des Agenten repräsentieren und durch den JIDM-Algorithmus erzeugt wurden. Diese Proxy-Objekte sind Teil des Gateways. Der Manager greift auf die Proxy-Objekte zu, indem er die Methoden der Proxy-Objekte mittels CORBA-Requests aufruft, worauf diese mit CORBA-Responses antworten. Wie für alle CORBA-Clients üblich, kann der Manager über die statische und/oder die dynamische Aufrufschnittstelle Methoden der Proxy-Objekte aufrufen. Die statische Aufrufschnittstelle ist für Managementzwecke allerdings wenig geeignet, da dazu die Client-Stubs aus der IDL-Beschreibung der Proxy-Objekte zu den Client-Programmen der Managementanwendung zur Übersetzungszeit gebunden werden müssen. Jedesmal, wenn neue Proxy-Objektklassen hinzukommen und verwendet werden sollen, müßte bei Verwendung der statischen Aufrufschnittstelle die Managementanwendung also neu übersetzt werden. Dies ist natürlich nicht praktikabel.

Für die dynamische Aufrufschnittstelle benötigt die Managementanwendung Objektreferenzen auf die Proxy-Objekte, auf die zugegriffen werden soll. Diese Referenzen werden dem Managementsystem wahlweise durch den Naming Service oder durch Events (etwa vom Gateway, wenn dort ein Proxy-Objekt erzeugt wurde) zur Verfügung gestellt. Beim Methodenaufruf gibt der Manager in einem Request die Objektreferenz des adressierten Proxy-Objektes, die aufzurufende Methode sowie die dazugehörigen Parameter an. Anhand der Objektreferenz lokalisiert der ORB das Proxy-Objekt und ruft dort die gewünschte Methode auf.

Der **snmpserver** ist die Komponente zur Kommunikation mit der SNMP-Umgebung. Hauptaufgabe des snmpserver ist das Aussenden von SNMP-Request-PDUs und das Empfangen der dazugehörigen Response-PDUs. Außerdem muß der snmpserver den Inhalt der Response-PDUs an die richtigen Proxy-Objekte weiterleiten können. Hierzu wird durch den JIDM-Algorithmus zu jeder erzeugten MOC sowie zu jedem Attribut dieser MOC die entsprechende SNMP OID als Konstante in die IDL-Definition des Moduls eingetragen (vgl. dazu auch Anhang B.2).

Der **snmptrapd** ist diejenige Gatewaykomponente, die für den Empfang der von den Agenten stammenden SNMP Trap-PDUs sowie deren Abbildung auf CORBA-Events. Zusammen mit dem snmpserver bildet der snmptrapd den Teil des Gateways, der gegenüber den SNMP-Agenten in der Managerrolle agiert.

Der letzte Bestandteil des Gateways ist der Discovery-Dämon **discoverd**, dessen Aufgabe im Auffinden neuer SNMP-Ressourcen besteht, wobei dies analog zu den üblichen Discovery-Mechanismen kommerzieller SNMP-Managementplattformen geschieht (Auslesen von ARP-Caches und Routing-Tabellen der Ressourcen, Subnetz-Discovery durch Versenden von ICMP-PDUs, Durchlaufen der Ressourcen-MIBs mit SNMP get-next Operationen usw.). Im Falle eines Auffindens neuer Ressourcen ist diese Komponente demzufolge insbesondere für die Erzeugung neuer Proxy-Objekte zuständig.

In den folgenden Abschnitten wird diskutiert, wie diese Komponenten zur Erreichung eines flexiblen und erweiterbaren Gesamtsystems zusammenwirken.

Funktionsweise des CORBA/SNMP Gateway-Prototyps

Abbildung 4.21 zeigt eine Beispielkonfiguration des Gateways zur Laufzeit. In der SNMP-Umgebung läuft ein SNMP-Agent auf dem Rechner *sun6*. Von der MIB-II des Agenten sind beispielhaft die `system`-Gruppe und die Routing-Tabelle (`ipRouteTable`) der MIB-II IP-Gruppe abgebildet. Der Wert der Variable `sysDescr` ist "SunOS 4.0.1"; die Routing-Tabelle besteht aus insgesamt zwei Zeilen. Ein Zeileneintrag (d.h. die SNMP-Instanzen, die zu dieser Zeile gehören) wird mit dem Wert des Spaltenobjektes `ipRouteDest` eindeutig identifiziert. Der Agent wird über die IP-Adresse 129.188.215.16 und die Portnummer 161 angesprochen.

Im Gateway wurden durch den Discovery-Dämon die in der Abbildung grau unterlegten Proxy-Objekte erzeugt, die der Agenten-MIB entsprechen. Für die `system`-Gruppe wurde im Gateway eine Instanz der IDL-Schnittstelle systemGroup generiert. In dieser Schnittstelle wurde für jedes Managed Object der SNMP-Gruppe `system` mit dem JIDM-Algorithmus ein Attribut definiert. Auf die Instanz eines Managed Object kann über das gleichnamige Attribut des Proxy-Objektes zugegriffen werden. Für das Managed Object `sysDescr` stehen dazu zwei Methoden, get_sysDescr und set_sysDescr, zur Verfügung. Bei einem Aufruf dieser Methoden werden zusätzlich die Werte von zwei Attributen benötigt, um die richtige SNMP-Instanz zu adressieren: Während das Attribut destination des Proxy-Objektes der Klasse systemGroup die IP-Adresse des Agenten beinhaltet, ist das Attribut indexinfo mit dem Zugriffsidentifikator der SNMP-Variablen belegt, die das Proxy-Objekt repräsentiert (hier: **.0**). Die Objektidentifikatoren der SNMP-Variablen wurden zuvor als Konstante im Interface Repository abgelegt.

In unserem Beispiel wird für jede Zeile der Routing-Tabelle eine Instanz der IDL-Klasse ipRouteTableEntry erzeugt. Analog zum oben geschilderten Fall ist bei beiden Proxy-Objekten das Attribut destination mit der IP-Adresse des SNMP-Agenten belegt. Die Attribute indexinfo sind hingegen mit dem *Wert* des Spaltenobjektes belegt, das einen Zeileneintrag der Routing-Tabelle eindeutig identifiziert (hier: der Wert des Indextyps `ipRouteDest`). Das erste Proxy-Objekt erhält den Index (.128.129.215.13) der ersten Tabellenzeile, das andere den der zweiten Zeile (.234.111.116.64). Hierdurch sind die Proxy-Objekte eindeutig einer Zeile der Routing-Table zuzuordnen. Die Elemente einer Zeile können mit den Attributzugriffsfunktionen des entsprechenden Proxy-Objektes ausgelesen bzw. gesetzt werden. Die zentrale Gateway-Komponente zum Erzeugen von SNMP-PDUs ist das **snmpserver**-Objekt. Die Methoden snmp_get() und snmp_set() erzeugen und versenden SNMP Request-PDUs und empfangen die dazugehörigen SNMP Response-PDUs. Die CORBA-Laufzeitumgebung sorgt dafür, daß ein Aufruf einer Methode eines CORBA-MOs auf das Proxy-Objekt weitergeleitet wird (in der Abbildung durch die gestrichelte Linie dargestellt). Die Schnittstellen der CORBA-MOs (bzw. Proxy-Objekte) können durch ORB-Dienste ermittelt werden.

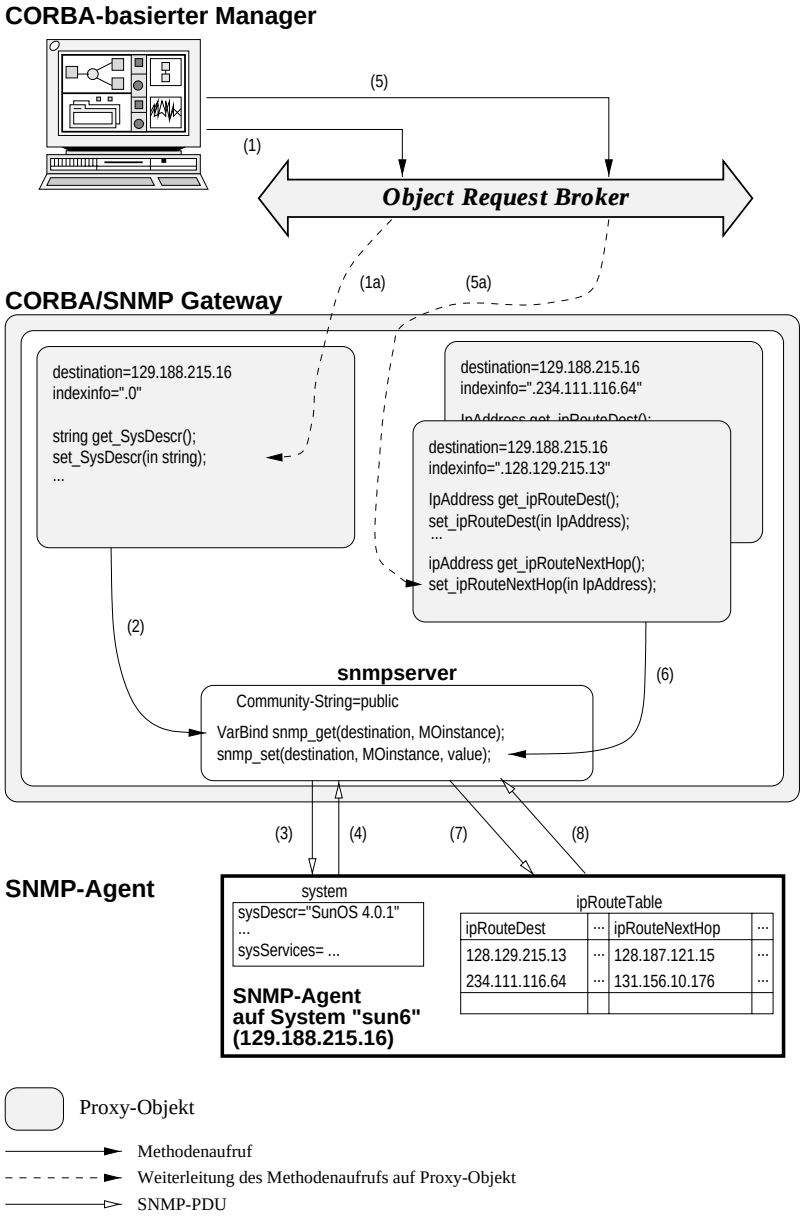


Abbildung 4.21: Funktionsweise des CORBA/SNMP-Gateways

SNMP Get/Set-Anfragen

Zunächst sei angenommen, daß der Manager vom Proxy-Objekt der Klasse `systemGroup` den Wert des Attributs `sysDescr` auslesen will.

1. Dies geschieht durch Aufruf der Methode get_SysDescr. Der Aufruf wird dann von der CORBA-Laufzeitumgebung an das Proxy-Objekt weitergeleitet (Schritt (1a) in Abbildung 4.21).

2. Innerhalb der aufgerufenen Methode erfolgt ein Aufruf der Methode snmp_get des **snmpserver**-Objektes, von dem jedes Proxy-Objekt eine Objektreferenz speichert.

Der Parameter MOinstance dieser Methode setzt sich zusammen aus dem Objektidentifikator von `sysDescr` und dem Wert des Attributs indexinfo. Der Objektidentifikator einer MIB-Variable steht im Interface Repository und wird aus Effizienzgründen bereits im Proxy-Objekt gespeichert. An ihn wird der Zugriffsidentifikator der SNMP-Variable angehängt. Konkret ergibt sich für den Parameter MOinstance der Wert 1.3.6.1.2.1.1.1.0. Das **snmpserver**-Objekt erzeugt daraufhin die *Variable Binding List* für die GetRequest-PDU, die er zusätzlich mit dem Community-String versieht.

3. Die mit allen Angaben versehene SNMP-PDU wird an den Agenten verschickt, der den Wert der angegebenen Objektinstanz ermittelt (in unserem Beispiel "SunOS 4.0.1") und mit einer GetResponse-PDU wieder an das Gateway zurückgesandt.

4. Aus der eingehenden GetResponse-PDU extrahiert das **snmpserver**-Objekt den Wert der SNMP-Variable `sysDescr` und gibt ihn dem Proxy-Objekt weiter (Rückgabewert der Methode snmp_get). Ferner überprüft das Proxy-Objekt, ob eine Fehlersituation aufgetreten ist, d.h. ob eine Exception vom **snmpserver**-Objekt ausgelöst wurde. Ist dies nicht der Fall, kann das Ergebnis an den Manager zurückgegeben werden.

5. Für ein zweites Beispiel, das den Zugriff auf SNMP-Tabellen illustriert, nehmen wir an, daß der Eintrag für `ipRouteNextHop` der ersten Zeile der Routing-Tabelle geändert werden soll. Pakete mit der Zieladresse 128.129.215.13 sollen fortan nicht mehr an die Adresse 128.187.121.15 sondern an die Adresse 131.156.10.176 weitergeleitet werden. Diesen neuen Wert für das Attribut gibt der Manager beim Aufruf der Methode set_IpRouteNextHop des Proxy-Objektes an. Um herauszufinden, auf welchem Objekt er die Methode aufrufen muß, hat der Manager vorher das Attribut ipRouteDest abgefragt. Wiederum sorgt die CORBA-Laufzeitumgebung dafür, daß die entsprechende Methode des Proxy-Objektes aufgerufen wird (Schritt (5a) in Abbildung 4.21).

6. Der Parameter MOinstance für den Aufruf von snmp_set wird auf dieselbe Weise wie oben angegeben berechnet und setzt sich aus der OID (1.3.6.1.2.1.4.21.1.7) sowie indexinfo (.128.129.215.13) zusammen. Der Parameter value hingegen enthält die neue IP-Adresse 131.156.10.176.

7. Diesmal wird vom **snmpserver**-Objekt eine SetRequest-PDU erzeugt und an den SNMP-Agenten versandt.

8. Die vom SNMP-Agenten zurückgeschickte SetResponse-PDU dient primär der Feststellung evtl. aufgetretener Fehler. Falls die SNMP-Variable nicht gesetzt werden konnte (da beispielsweise der Community-String falsch war) wird dies in der Response-PDU durch den Fehler `authorizationError` angezeigt. Das snmpserver-Objekt löst die der Fehlermeldung entsprechende Standard-Exception

NO_PERMISSION aus. Dasselbe tut das Proxy-Objekt, sobald aus der Methode snmp_set zurückgekehrt wird.

Anwendung von GetNextRequest- und GetBulk-Anfragen

Für den schreibenden Zugriff auf SNMP-Agenten kann beim Aufruf der Methode snmp_set() jeweils nur eine SetRequest-PDU an den Agenten gesandt werden. Für den lesenden Zugriff hingegen stehen in SNMP neben der GetRequest-PDU zwei weitere Protokolldateneinheiten zur Verfügung, nämlich GetNext und GetBulk. Mit ihnen können, wie wir in Abschnitt 3.1.3 erläutert hatten, Tabellen durchlaufen bzw. auf größere Datenmengen von SNMP-Ressourcen effizienter zugegriffen werden. Der Manager kann beispielsweise alle Attributwerte der einem Tabellenzeileneintrag entsprechenden Objekte gleichzeitig abfragen. Um den gesamten Inhalt der Tabelle vom SNMP-Agenten zu holen, wird SNMP-seitig nur eine einzige GetBulk-PDU benötigt. Aufgrund der Tatsache, daß der JIDM-Algorithmus jede Tabellenzeile auf voneinander unabhängige Proxy-Objekte abbildet, kann von der GetBulk-Optimierung kein Gebrauch gemacht werden, da die die Tabellenzeilen darstellenden Objekte voneinander unabhängig die Methoden des **snmpserver**-Objektes aufrufen, um die SNMP-PDUs zu generieren. Dieses kann aber weder feststellen, daß zwischen angeforderten SNMP-Variablen ein Zusammenhang besteht, noch kann es a priori wissen, wie viele SNMP-Variablen von den Proxy-Objekten noch angefordert werden, die in einer GetBulk-PDU zusammengefaßt werden könnten. Jeder Methodenaufruf wird einzeln abgearbeitet, was bedeutet, daß mit einer GetRequest-PDU immer nur eine einzige SNMP-Variable abgefragt wird. Folgende Maßnahmen müßten getroffen werden, um GetBulk-PDUs zu unterstützen:

- Erweiterung der Proxy-Objekte dahingehend, daß mehrere (unterschiedliche) Attributzugriffsfunktionen gleichzeitig aufgerufen werden können. Bisher war ein Proxy-Objekt erst nach der Rückkehr aus einer Attributzugriffsfunktion wieder bereit, Methoden abzuarbeiten.
- Modifizieren der Parameter der Methode snmp_get() damit Proxy-Objekte mehr als eine SNMP-Variable übergeben können.
- Im **snmpserver**-Objekt werden Anforderungen von Proxy-Objekten nach Zielagent sortiert „gesammelt“ und mehrere Variablen gleichzeitig in einer GetRequest-PDU oder einer GetBulk-PDU abgefragt. Dies stellt einen Versuch dar, angeforderte MIB-Variablenwerte mit möglichst wenigen SNMP-PDUs zu erhalten.

Hierfür müßte insbesondere das von uns verfolgte Konzept des zustandslosen Gateways aufgegeben werden, da Anfragen entsprechend des Zielsystems in Warteschlangen eingereiht werden müßten, um gemeinsam in einer GetBulk-PDU angefordert werden zu können. Ferner müßte ein Timeout-Mechanismus implementiert werden, der veranlaßt, daß nach Ablauf einer bestimmten Zeit die Warteschlangeninhalte in SNMP-PDUs verpackt und abgeschickt werden. Da uns der Mehrwert im Vergleich zu den Implementie-

rungskosten nicht gerechtfertigt erschien, haben wir von einer Verwendung der GetBulk-Operation abgesehen.

Aus verwandten Gründen kann auch die GetNext-Operation nur eingeschränkt verwendet werden: Durch eine GetNextRequest-PDU werden Informationen vom Agenten zurückgegeben, die nicht dem Proxy-Objekt zugeordnet werden dürfen, das die Methode snmp_get() aufgerufen hat. Wenn beispielsweise die Methode get_ipRouteNextHop() des Proxy-Objektes, das die erste Zeile der Routing-Tabelle repräsentiert (in Abbildung 4.21 das Proxy-Objekt mit indexinfo=.128.129.215.13), aufgerufen wird und daraufhin eine `GetNextRequest-PDU` verschickt wird, so wird der entsprechende Spalteneintrag der nächsten Tabellenzeile, also 131.156.10.176 vom Agenten zurückgegeben. Da mit der GetNextRequest-PDU Tabellen durchlaufen werden können, von denen die Zeilenindizes nicht bekannt sind, wird diese PDU im Gateway nur vom **Discovery-Dämon** eingesetzt.

Weiterleitung asynchroner Ereignismeldungen

Für die Weiterleitung von SNMP-Traps benutzen wir analog zu Abschnitt 4.2 typisierte Event Channels, von denen jeder einem bestimmten SNMP-Trap zugeordnet ist. Die Event Channels selbst werden in einen Naming Graph eingetragen[10], wodurch eine CORBA-basierte Managementapplikation die Event Channels auffinden und sich bei diesen zum Empfang von Ereignismeldungen registrieren kann.

Das **snmptrapd**-Objekt, das an Port 162 SNMP-Trap-PDUs empfängt, registriert sich bei allen Event Channels als *Supplier*, da sämtliche SNMP-Traps hier eingehen und in typisierte Events umgewandelt werden. Er spielt somit dieselbe Rolle wie der in Abschnitt 4.2.4 beschriebene EFD_Supplier und die Verfahren zur Zustellung der Ereignismeldungen an die registrierten CORBA-basierten Managementapplikationen, die in der Rolle des *Consumers* agieren, laufen somit identisch ab. Wir werden daher zur Vermeidung von Wiederholungen auf eine Diskussion der Mechanismen zur Ereignisweiterleitung verzichten.

Erweiterung des Gateways um neue Ressourcen

Die von einem Sprachcompiler erzeugten IDL-Klassendefinitionen einer SNMP-MIB (siehe dazu Abschnitt 4.4.3 bzw. Anhang B.2) werden in das **Interface Repository** des ORB eingetragen (vgl. Abbildung 4.22), also nicht direkt in das Managementsystem, wie es bei SNMP-Managern bzw. SNMP-Plattformen notwendig ist. Für jede neu hinzukommende oder geänderte Agenten-MIB wiederholt sich dieser Vorgang, der im wesentlichen daraus besteht, die aus dem JIDM-Algorithmus erzeugten IDL-Objektbeschreibungen mit dem Code des Gateways neu zu übersetzen und zu binden, um die neuen MO-Beschreibungen dem Gateway bekanntzugeben. Es gibt bisher von der OMG keinen standardisierten Mechanismus, um dies zur Laufzeit zu tun, weshalb wir bei der Implementierung der Discovery-Komponente unseres Gateways einen proprietären Mechanismus der von uns verwendeten SOM/DSOM-Entwicklungsumgebung verwendet haben. Es handelt

[10] Dies ist möglich, da Event Channels selbst Objekte sind.

sich hierbei um das *DSOM Metaclass Framework*, das es uns gestattet, neue Objektdefinitionen zur Laufzeit in das Interface Repository einzuspielen und deren Implementierung mit dem Gateway zu verknüpfen (siehe Abbildung 4.22). Im Rahmen des OMG Meta-Object Service ist ein solches Verfahren angedacht, jedoch bisher noch nicht standardisiert worden.

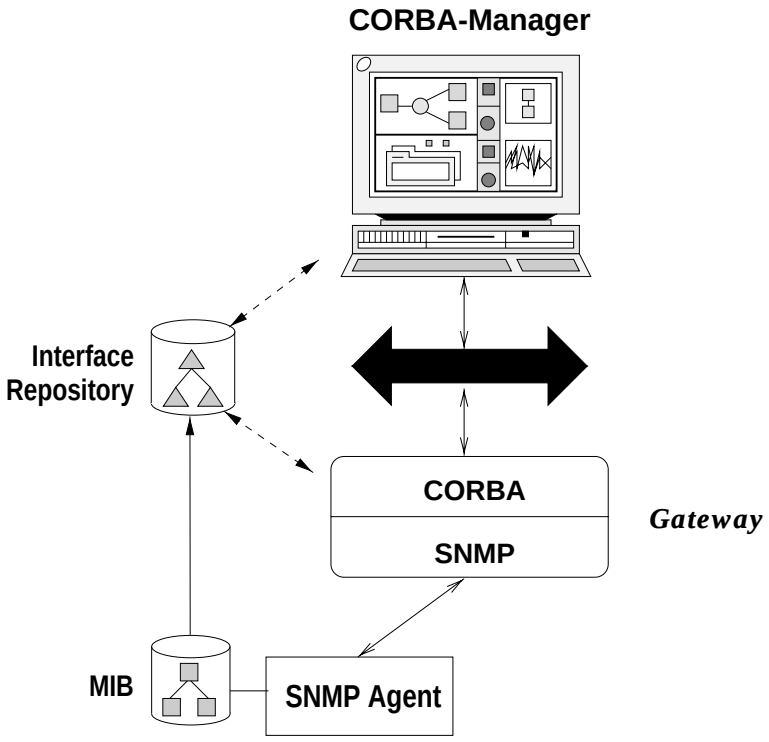


Abbildung 4.22: Bekanntgabe neuer Ressourcen

In jeder CORBA-Umgebung existiert ein verteiltes Interface Repository, auf das alle CORBA-Anwendungen Zugriff haben. Mit dem einmaligen Laden einer in IDL übersetzten Agenten-MIB ins Interface Repository steht diese MIB (bzw. die entsprechenden Schnittstellendefinitionen) dann automatisch sowohl der Managementanwendung als auch dem Gateway zur Verfügung.

4.4.5 Transformation der Funktionsmodelle

Die Transformation der Funktionsmodelle zielt auf die Wiederverwendbarkeit von vordefinierten, ggf. delegierbaren Managementdiensten in heterogenen verteilten Systemen ab. Diese in etablierten Managementarchitekturen vorhandenen Dienste (z.B. die Systems Management Functions im OSI-Management), haben wir in [KeNe 95] mit der Dienstmenge verglichen, die von CORBA angeboten wird. Wie bereits in Abschnitt 3.1.2 festgestellt wurde, ist gegenwärtig der Anteil an spezifisch auf das Management ausgerichteter Funktionalität in CORBA noch relativ gering. Ziel dieses Vergleichs war es daher,

herauszufinden, welche Managementfunktionalität für das CORBA-basierte Management verteilter kooperativer Managementsysteme benötigt wird und inwieweit bereits in etablierten Managementarchitekturen vorhandene Funktionalität übernommen werden kann.

Wir sind dabei zu folgenden Ergebnissen gekommen:

- Insbesondere zwischen den OSI *Systems Management Functions* und den diversen *CORBAservices* bestehen einige Gemeinsamkeiten; von einem vollständigen Grad der Überdeckung kann jedoch keine Rede sein. Es ist daher notwendig, Übergänge zwischen den Funktionsmodellen zu schaffen.
- Soll eine weitgehende Kooperation der verschiedenen Managementarchitekturen möglich sein, müssen möglichst nahtlose Übergänge zwischen ihnen geschaffen werden, bei denen möglichst wenig der angebotenen Funktionalität verloren geht. Die Managementdienste der einen Architektur sollten auch von der anderen Architektur aus verwendbar sein. Dies ist umso besser und einfacher machbar, je „ähnlicher“ sich die bereitgestellten Funktionen sind.
- Liegen *vergleichbare Infrastruktur-Dienste* in beiden Architekturen vor, müssen sie aufeinander abgebildet werden. Vor allem zwischen dem OMG-Ansatz und dem OSI-Management bestehen im Bereich der Management-Dienste große Ähnlichkeiten. Mit diesen Ähnlichkeiten ist es prinzipiell möglich, Übergänge zwischen diesen Ansätzen innerhalb der Architekturen zu schaffen und damit die Interoperabilität der Architekturen insgesamt zu verbessern. Können die Dienste verschiedener Architekturen in einer ähnlichen Art und Weise durch die Anwendungen benutzt werden, wird auch die Skalierbarkeit des Managements für komplexe DV-Infrastrukturen insgesamt besser.
- Falls die Managementdienste der einen Architektur *keine Entsprechung* in der anderen betrachteten Architektur haben, kann man durch die Überführung der Beschreibungen dieser Dienste in das entsprechende Informationsmodell die Managementfunktionalität einer Architektur von einer anderen Managementarchitektur aus nutzen. Damit könnte man z.B. in der OSI-Architektur existierende Mechanismen zur Schwellwertüberwachung von CORBA-Objekten aus verwenden.

Das Problem der Abbildung der Funktionsmodelle kann im wesentlichen auf das Problem der Abbildbarkeit der jeweiligen Informations- und Kommunikationsmodelle reduziert werden. Dies ist möglich, da Managementfunktionalität sowohl in OSI als auch in CORBA in derselben Notation spezifiziert wird wie Managementinformation. Im Fall von OSI handelt es sich hierbei um GDMO, im Falle von CORBA um IDL. Da diese Dienste mit Hilfe des Management-Informationsmodells definiert wurden, könnte man aus diesen Beschreibungen IDL-Schnittstellen generieren, evtl. vorhandene Implementierungen damit kapseln und so z.B. OSI-Managementfunktionen auch per CORBA verfügbar machen. Um Managementfunktionalität architekturübergreifend nutzen zu können, ist es notwendig, die Dienstschnittstellen in die jeweilige Notation zu transformieren (dies geschieht mit

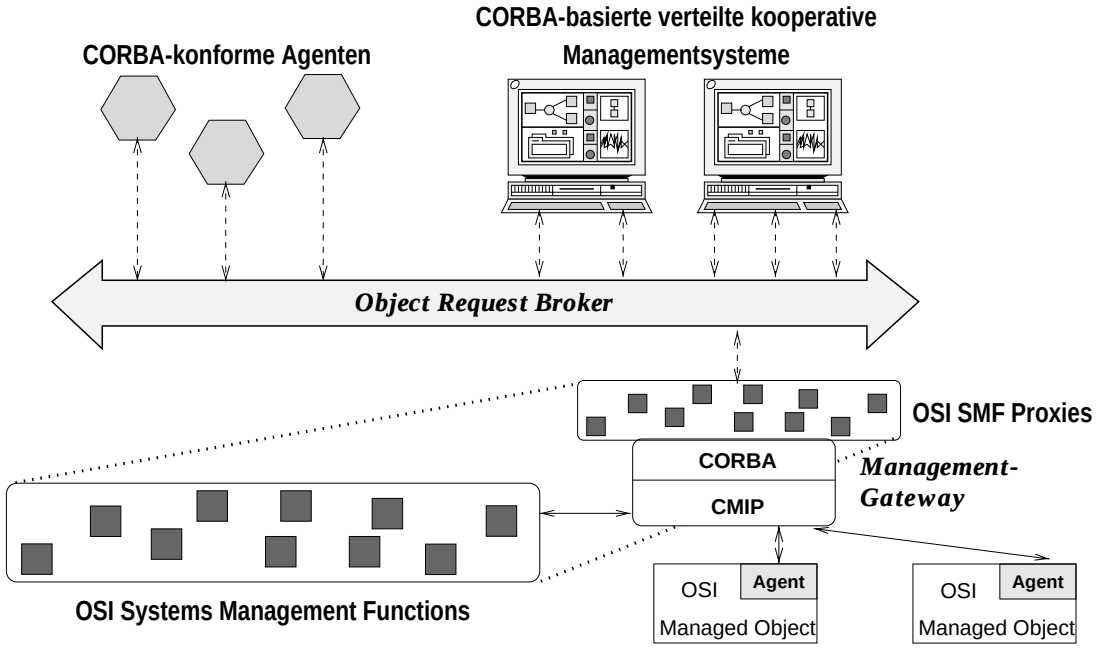


Abbildung 4.23: Nutzung von Diensten anderer Managementarchitekturen

den IIMC- bzw. JIDM-Algorithmen) und Mechanismen zum Zugriff auf diese Schnittstellen zur Verfügung zu stellen. Abbildung 4.23 stellt diesen Zusammenhang graphisch dar.

Unsere Analyse des Status Quo hinsichtlich der Managementarchitekturen (siehe Abschnitt 3.1 bzw. [KeNe 95]) hat aufgezeigt, daß insbesondere die OSI/TMN-Managementarchitektur über das umfangreichste Dienstangebot verfügt und somit in erster Linie ein Kandidat zur Erweiterung von CORBA bezüglich neuer Managementdienste ist. Das Internet-Management hingegen verfügt gegenwärtig über keinerlei generische Managementdienste, was sich jedoch mit dem Aufkommen von DISMAN möglicherweise ändern wird. Jedoch bleibt der DISMAN-Dienstumfang zumindest in seiner jetzigen Fassung hinter der Menge an bereits standardisierten CORBAservices zurück, weshalb sich die Nutzung von Internet-Managementdiensten durch CORBA auch in absehbarer Zeit nicht lohnen wird.

4.5 Zusammenfassung

Dieses Kapitel hat anhand der drei möglichen Integrationsansätze aufgezeigt, daß es durchaus möglich und aussichtsreich ist, Übergänge zwischen Managementarchitekturen im Sinne eines Umbrella Managements zu schaffen. Diese Übergänge können allerdings, auch das hat sich bei unseren Implementierungen gezeigt, wegen der Unterschiedlichkeit der Architekturen zwangsläufig nicht nahtlos sein. Ein charakteristisches Beispiel hierfür

ist die Tatsache, daß die im Internet-Management nicht vorhandene Fähigkeit zur automatischen Ermittlung der unterstützten MIBs neuer Ressourcen bedingt, daß die zugehörigen MIBs einem Manager bzw. Gateway immer „von Hand“ bekanntgegeben werden müssen.

Während der Darstellung der Interoperabilitätsproblematik von Managementarchitekturen mehrere Veröffentlichungen (z.B. [GlHa 96], [KaSe 94]) gewidmet sind, so liegen bis zum heutigen Tage kaum konkrete Betriebserfahrungen mit Implementierungen vor, die eine abschließende Bewertung der Alternativen gestatten. Beispielsweise befaßt sich [GlHa 96] mit dem auf multiarchitekturellen Agenten (dort als „Q-Addition“ bezeichnet) basierenden Ansatz sowie Gateway-basierter Integration („Q-Adaptation“), läßt jedoch eine abschließende Bewertung offen.

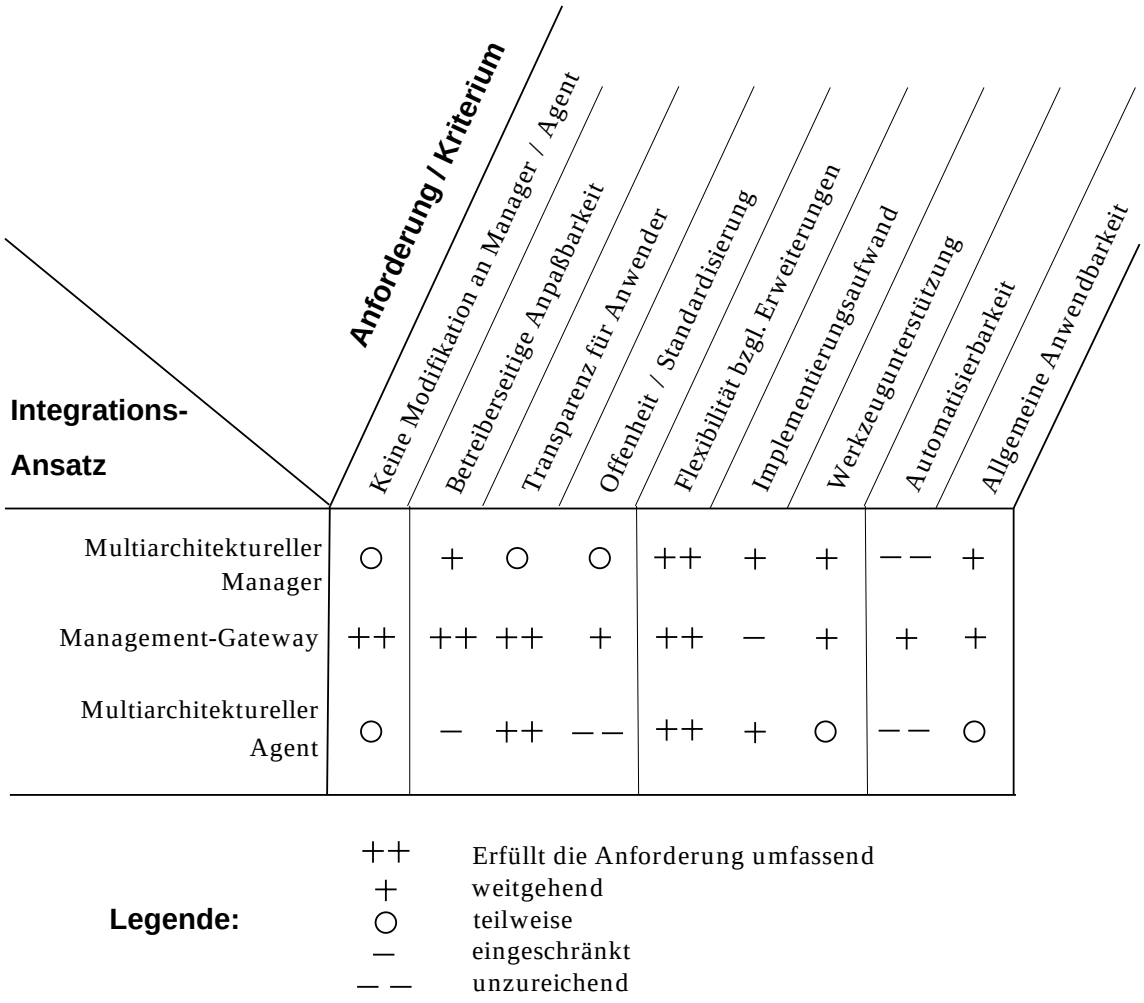


Abbildung 4.24: Bewertung der Alternativen für das Umbrella Management

Demgegenüber haben wir anhand unserer Prototyp-Implementierungen den Versuch unternommen, eine solche Bewertung hinsichtlich der von uns zu Beginn dieses Kapitels formulierten Anforderungen vorzunehmen, die in Abbildung 4.24 dargestellt ist. Es fällt

dabei auf, daß der Gateway-basierte Ansatz die meisten positiven Eigenschaften für sich verbuchen kann, da insbesondere die Mechanismen zur Umsetzung der Informations- und Kommunikationsmodelle demnächst standardisiert werden und somit vollständig das Kriterium der Offenheit erfüllen. Der wohl wichtigste Vorteil von Management-Gateways besteht darin, weder Modifikationen an Managern noch an Agenten vornehmen zu müssen, was sich in der Unabhängigkeit von konkreten Produkten und somit in einem breiten Anwendungsspektrum äußert.

Die Durchführung von Leistungsvergleichen gestaltet sich angesichts der stark unterschiedlichen Randbedingungen, denen diese Ansätze unterliegen, als ausgesprochen kompliziert, da die Wahlfreiheit bereits dadurch eingeschränkt wird, ob man von einer Hersteller- (Managementsystem oder Agent), Anwender- oder Systemintegratorenrolle ausgeht. Unzweifelhaft besitzt ein multiarchitektureller Agent sowohl die besten Transparenzeigenschaften für den Netzbetreiber als auch das mit Abstand beste Leistungsprofil, weil hier nicht zwischen Managementarchitekturen vermittelt wird, sondern dieselbe Instrumentierung lediglich um eine weitere Schnittstelle ergänzt wird. Diese Vorteile werden jedoch durch die Tatsache neutralisiert, daß der Zugriff auf die Instrumentierung einer Ressource (und damit die Durchführbarkeit dieses Ansatzes) oft nur dem Ressourcenhersteller vorbehalten ist und folglich weder dem Netzbetreiber (bzw. einem von ihm beauftragten Systemintegrator) noch dem Hersteller eines Managementsystems offensteht.

Unsere Messungen der Antwortzeiten eines Management-Gateways lassen insbesondere erkennen, daß der Preis zur Durchführung komplexer und leistungsfähiger (Scoping und Filtering) Operationen oft durch entsprechend lange Wartezeiten (z.T. mehrere Minuten) erkauft wird, während einfache Zugriffe in Sekundenbruchteilen ablaufen. Der klassische Tradeoff zwischen Komfort bzw. Flexibilität und Komplexität (hinsichtlich Zeit und Platz) bestätigt sich auch an dieser Stelle.

Wir konnten bei der Entwicklung unserer CMIP/SNMP bzw. CORBA/SNMP Gateway-Prototypen ebenfalls feststellen, daß sich mit dem TMN/C++ API die Entwicklung von OSI/TMN-Agenten auf nahezu identische Weise wie die von CORBA-Agenten gestaltet: GDMO-Darstellungen werden von der Werkzeugumgebung durch ein „C++ language mapping“ automatisch in entsprechende C++ Klassenrümpfe umgewandelt, welche vom Entwickler um die eigentliche Instrumentierung ergänzt werden müssen. Analog zu CORBA kommt auch hier das Prinzip zum Tragen, den Entwickler von den Spezifika der Kommunikationsinfrastruktur (in diesem Fall: CMIP) vollständig abzuschirmen, was die Komplexität für den Entwickler drastisch senkt. Erkauft wird dieser Komfort jedoch durch den verhältnismäßig großen Umfang eines ausführbaren Agenten, der in unserem Falle (MIB-II und UNIX-MIB) unter Verwendung von *shared libraries* deutlich über 5 MB betrug und in der standalone-Variante bei knapp 40 MB lag. Es bleibt zu hoffen, daß die weitere Verbesserung der von uns verwendeten relativ jungen Entwicklungsumgebung hier zu signifikanten Effizienzsteigerungen führt.

Auch wenn die Bewertung des multiarchitekturellen Managers diese Ausprägungsform des Umbrella Managements wenig reizvoll erscheinen läßt, so bietet sie für die vor-

liegende Arbeit einen entscheidenden Mehrwert: Durch die Erweiterung einer SNMP-Managementplattform zu einem multiarchitekturellen CORBA/SNMP-Manager haben wir die Möglichkeit geschaffen, CORBA-basierte Agenten unmittelbar von einer kommerziellen Plattform aus zu administrieren, was wir uns bei der Vorstellung unserer Agentenprototypen in Kapitel 6 zunutze machen werden. Dieser Zusatznutzen wird durch die Tatsache verdeutlicht, daß bis zum heutigen Tage noch keine offene CORBA-Managementplattform auf dem Markt erhältlich ist. Ferner zeigt unser Prototyp auf, wie durch geeignete Modularisierung bisherige monolithische Managementsysteme CORBA-konform und folglich verteilbar gemacht werden können. Wir werden diesen Aspekt am Ende von Kapitel 6 vertiefen.

Zusammenfassend läßt sich sagen, daß wir mit dem Umbrella Management die technische Grundlage zur Etablierung eines *vollständig* CORBA-basierten Enterprise Managements gelegt haben, da wir durch ersteres in die Lage versetzt werden, die klassischen Managementarchitekturen fast nahtlos zu integrieren. Demzufolge können wir bei der Konzeption des von uns im folgenden Kapitel entworfenen Objektmodells verteilter kooperativer Managementsysteme von einer *homogenen*, CORBA-basierten Umgebung ausgehen, ohne dadurch die Anwendbarkeit unserer Methodik in *heterogener* Umgebung zu gefährden.

Eine weitere Erkenntnis aus unseren Erfahrungen bei der Implementierung der CMIP/SNMP- sowie der CORBA/SNMP-Gateways besteht darin, daß auch Management-Gateways überwacht und gesteuert werden müssen. Wir haben dies in Abschnitt 4.4.4 anhand der Konfiguration zustandsbehafteter Gateways bezüglich der Polling-Intervalle motiviert. Gateways sind aufgrund ihrer Agent/Manager-Doppelrolle ein Spezialfall verteilter kooperativer Managementsysteme, wie wir es in Abschnitt 2.1 definiert hatten. Folglich wird das im folgenden Kapitel entwickelte Objektmodell neben gewöhnlichen Managementsystemen insbesondere auch Management-Gateways umfassen.

KAPITEL 5

Ein Objektmodell für das Management verteilter kooperativer Managementsysteme

Nachdem wir im vorigen Kapitel mit der Diskussion der unterschiedlichen Ausprägungsformen des Umbrella Managements sowohl das konzeptionelle als auch das implementierungstechnische Fundament für ein intergriertes Enterprise Management gelegt haben, werden wir uns nun im folgenden Kapitel der Instrumentierung verteilter kooperativer Managementsysteme widmen. Wie in der Einleitung der vorliegenden Arbeit motiviert wurde, umfaßt diese zwei Problembereiche, die wir als den **Informationsaspekt** sowie den **Funktionsaspekt** kategorisiert hatten. Ersterer legt den für effektives Management notwendigen Umfang an Managementinformation fest, die von einer Ressource zur Verfügung gestellt werden muß; letzterer fokussiert auf die möglichst generischen und flexibel kombinierbaren Managementdienste, welche die von der Instrumentierung zur Verfügung gestellten Parameter hinsichtlich der Zielvorgaben eines Betreibers aufbereiten.

Entsprechend ist dieses Kapitel strukturiert: Wir werden zunächst in Abschnitt 5.1 einen werkzeugunterstützten Ansatz zur Gewinnung neuer CORBA-konformer Agenten aus bestehenden SNMP-Implementierungen vorstellen, der die mit der Verwendung von CORBA einhergehenden Vorteile moderner Softwareentwicklungssysteme demonstriert. Die Vorstellung dieses Ansatzes erfolgt an einem einfachen Beispiel: Es handelt sich dabei um einen Agenten für das integrierte Management UNIX-basierter Endsysteme, der am Lehrstuhl entwickelt und seitdem kontinuierlich verbessert wurde. Seine Behandlung wird nicht zuletzt durch unsere Erfahrungen gerechtfertigt, daß auch heute noch im Betriebssystem auftretende Fehlersituationen zu einem wesentlichen Anteil für Störungen der darauf aufsetzenden Managementsysteme verantwortlich sind.

Im Anschluß daran werden wir eine Methodik vorstellen, die darauf abzielt, generische MOCs für Managementsysteme zu entwerfen, um bereits so nahe wie möglich an der Wurzel der Vererbungshierarchie ein Maximum an Managementinformation und -diensten zu definieren. Wie wir in Abschnitt 1.3 festgestellt haben, ist dies in gegenwärtigen (objektorientierten) Managementarchitekturen nicht gegeben. Der Vorteil einer hohen Informati-

onsdichte an der Spitze der Vererbungshierarchie bietet uns neben der Eigenschaft einer übersichtlichen Darstellung unseres Management-Objektmodells verteilter kooperativer Managementsysteme insbesondere den Mehrwert, konkrete Vorgaben an den Minimalumfang einer geeigneten Instrumentierung machen zu können. Letztere können beispielsweise als Referenz für die Bewertung zukünftiger (kommerzieller) Lösungen dienen, um den bereitgestellten Umfang an Managementinformation für Managementsysteme in die Praxis umzusetzen.

Der darauf folgende Abschnitt 5.3 konkretisiert diese Aspekte, indem wir unser Objektmodell für verteilte kooperative Managementsysteme (eine MIB für Managementsysteme) vorstellen, das es uns gestattet, Mid-level-Managementsysteme und heutige Managementplattformen so zu instrumentieren, damit sie von anderen Managementsystemen überwacht und gesteuert werden können. Aus Kapitel 4 haben wir darüber hinaus die Erkenntnis gewonnen, daß sich aus der Manager/Agent-Doppelrolle von Management-Gateways die Notwendigkeit des Managements solcher Übergangspunkte ergibt. Wir werden daher ebenfalls ein Objektmodell für Management-Gateways entwickeln, was aufgrund des von uns verwendeten objektorientierten Designs ohne großen Aufwand möglich ist. Insbesondere dem Management von Management-Gateways ist bislang von der Fachwelt keinerlei Aufmerksamkeit zuteil geworden.

5.1 Transformation bestehender Agentenmodelle

Dieser Abschnitt präsentiert einen Ansatz zur Migration von bestehendem modular aufgebauten SNMP-Agentencode in eine CORBA-Umgebung und schildert die Schritte dessen konkreter Anwendung anhand eines praxisnahen und einfachen Beispiels. Mit dieser Aufgabenstellung sind primär die Hersteller von Agentensystemen konfrontiert, da von ihnen bereits heute die Unterstützung von CORBA durch den Markt gefordert wird, andererseits jedoch die bestehende Instrumentierung des Agenten nicht modifiziert werden sollte.

Bei dem von uns gewählten Beispiel handelt es sich um einen Agenten für das integrierte Management von UNIX-Workstations, der mit unserer Aufgabenstellung insofern zusammenhängt, als die Funktionsfähigkeit von Managementsystemen unmittelbar von der Verfügbarkeit des Betriebssystems abhängig ist. Außerdem existieren klare und allgemein akzeptierte Vorstellungen, welche Managementinformation von einem UNIX-Systemmanagementagenten bereitgestellt werden müssen (vgl. hierzu [GuNe 95] sowie die Ausführungen zum IBM *System Information Agent (SIA)* in Abschnitt 3.3.1). Wir können uns somit zunächst auf die Darstellung des Transformationsverfahrens sowie des Nutzens einer umfassenden Werkzeugunterstützung konzentrieren und den Designaspekt momentan noch zurückstellen. In den späteren Abschnitten 5.2 und 5.3 wird der Schwerpunkt unserer Untersuchungen auf dem Design liegen.

5.1.1 Vorgehensmodell für die Transformation

Aufgrund der Tatsache, daß CORBA – als die von uns ausgewählte Basisarchitektur – für beliebige verteilte Anwendungen konzipiert wurde, können wir bei unserem Lösungsweg jüngste Entwicklungen im Bereich objektorientierter Software-Engineering-Werkzeuge nutzen, um einen effizienten Entwicklungsvorgang zu garantieren. Die Struktur dieses Teilkapitels orientiert sich an dem in Abbildung 5.1 skizzierten Vorgehensmodell, das wir nachfolgend vorstellen. Das Ziel unseres Ansatzes besteht im Design objektorientierter Kapseln für eine bestehende Instrumentierung, womit der bereits existierende SNMP-Agent um eine CORBA-Schnittstelle erweitert wird. Er wird dadurch zu einem **multiarchitekturellen Agenten**, auf dessen grundlegende Eigenschaften wir in Abschnitt 4.3 eingegangen sind. Die praktische Anwendbarkeit unseres Ansatzes wurde anhand des von uns implementierten Prototypen nachgewiesen.

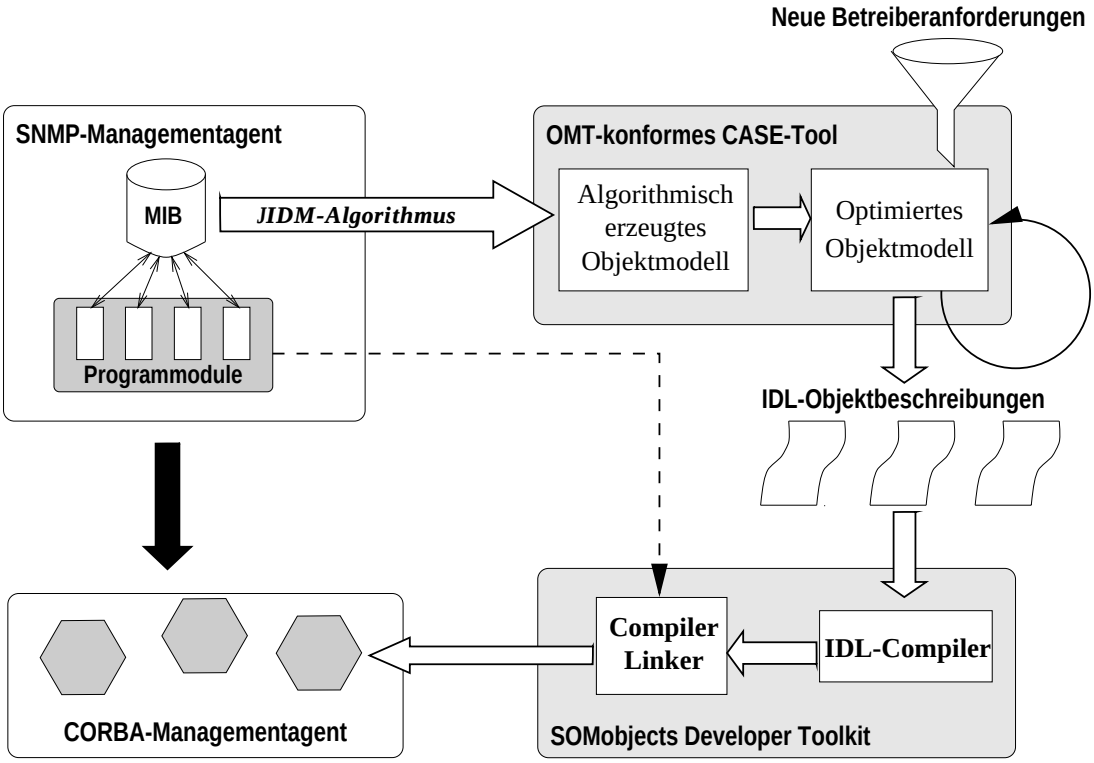


Abbildung 5.1: Überblick über den Transformationsvorgang

Abschnitt 5.1.2 beschreibt die Eigenschaften des am Lehrstuhl entwickelten SNMP-Agenten zum Management von UNIX-Workstations. Der JIDM-Algorithmus zur Umsetzung des Internet-Informationsmodells in CORBA-Objektbeschreibungen wurde bereits in Abschnitt 4.4.3 beschrieben. Das Ergebnis dieses Transformationsvorgangs ist ein algorithmisch erzeugtes Objektmodell, das jedoch noch in einigen Punkten optimiert werden muß. Die Gründe dafür sowie die Schritte und Werkzeuge zur Optimierung des vorhandenen Objektmodells sind in Abschnitt 5.1.3 beschrieben. Hierbei ist es von großer Be-

deutung, auf Werkzeuge wie CASE-Tools zurückgreifen zu können, die den zyklischen Prozeß der objektorientierten Analyse und des Designs geeignet unterstützen. Sie schaffen auch die Grundlage, um das bestehende Modell um neue Anforderungen von Seiten der Betreiber nahtlos zu erweitern. Nachdem die Modellierung abgeschlossen ist, kann die Implementierung erfolgen, die in Abschnitt 5.1.4 dargestellt wird. Hierbei hat sich herausgestellt, daß die Schnittstellen zwischen den einzelnen Werkzeugen zum Teil noch verbesserungsbedürftig sind und man von einem reibungslosen Zusammenspiel aller am Entwicklungsprozeß beteiligten Komponenten in manchen Fällen noch etwas entfernt ist.

5.1.2 Ein SNMP-Agent für das Management von UNIX-Workstations

Zum heutigen Zeitpunkt muß trotz zahlreicher Bemühungen der Hersteller die Problemstellung, integriertes Management von UNIX-Workstations von einer Managementplattform aus sicherzustellen, noch immer als ungelöst betrachtet werden: Es existieren zwar Werkzeuge, die eine Vielzahl von Parametern eines Endsystems erfassen und mit Hilfe eines standardisierten Managementprotokolls an die Managementplattform weiterleiten [IBMSysMon94]; aktive Eingriffsmöglichkeiten durch den Administrator fehlen jedoch völlig. Andererseits existieren Produkte, die zwar Eingriffe zur Steuerung des Systems erlauben; die Kommunikation mit der Managementplattform geschieht jedoch lediglich auf der Grundlage eines herstellerspezifischen, d.h. proprietären Protokolls, dessen Schnittstellen nicht offengelegt sind [ITOcg 97]. Zusätzlich sind beide Ansätze von einer Bottom-up-Vorgehensweise geprägt, die nur selten die tatsächlichen Bedürfnisse der Netzbetreiber berücksichtigt. Von einer umfassenden, integrierten Managementlösung kann daher zur Zeit noch nicht die Rede sein.

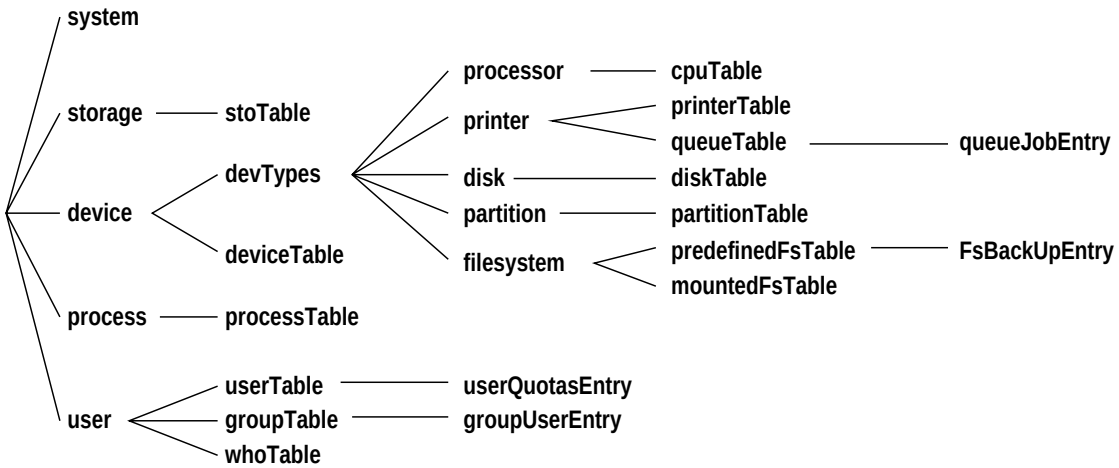


Abbildung 5.2: Eine MIB für das Management von UNIX-Workstations

Im Gegensatz zu kommerziell erhältlichen Werkzeugen wurde in bereits vor einiger Zeit am Lehrstuhl durchgeführten Arbeiten (beschrieben u.a. in [Guts 95], [Krie 94]) ein Top-down-Ansatz verfolgt, der das typische Aufgabenspektrum eines UNIX-Systemadministrators abdeckt: Es wurden Möglichkeiten für das Einrichten und Löschen von Benutzerkennungen und -gruppen sowie deren Quoten für den Zugriff auf Betriebsmittel (Plattenplatz, Druckseiten) ebenso vorgesehen, wie Funktionen zum Erfassen und ggfs. Stoppen der momentan aktiven Prozesse oder zum Mounten/Unmounten von Dateisystemen. Es ist somit nicht nur passives Monitoring von UNIX-Workstations möglich, sondern auch das aktive Eingreifen in den Betrieb des Systems. Als Managementprotokoll wird SNMP in der Version 2 verwendet.

Im Rahmen der vorgenannten Arbeiten wurde eine Systemmanagement-MIB entwickelt und der dazugehörige Agent auf verschiedenen Betriebssystemplattformen (IBM AIX, HP-UX, SunOS, Solaris) implementiert. Die MIB ist in Abbildung 5.2 schematisch dargestellt; sie umfaßt 195 MIB-Variablen und 15 Tabellen und stellt (unter anderem) ein Modell folgender Komponenten eines UNIX-Systems zur Verfügung:

- Speicher (Hauptspeicher, Swap-Bereich)
- Geräte (Prozessoren, Drucker, Platten und deren Dateisysteme usw.)
- Prozesse
- Benutzer (Kenndaten, Gruppen, Quoten usw.)

Im Falle der Benutzer- und Gruppenverwaltung tritt jedoch durch eine Einschränkung des Internet-Informationsmodells folgendes Problem auf: in der Regel sind auf einem UNIX-System mehrere Benutzer eingetragen, die ihrerseits wieder zu mehreren Gruppen gehören. Tabellen innerhalb von Tabellen sind jedoch explizit durch den entsprechenden Internet-Standard [rfc1902] untersagt. „Zu den Internet-Standards konforme“ Managementsysteme akzeptieren jedoch MIBs, die diese Anomalie aufweisen, klaglos.

Als sehr vorteilhaft für die Kapselung des SNMP-Agentencodes hat sich der modulare Aufbau des SNMP-Agenten erwiesen, bei dem auf eine strikte Trennung zwischen Protokoll- und Ressourcen-Modulen geachtet wurde. Als Intra-Agenten-Schnittstelle wird das **Distributed Protocol Interface (DPI)** [rfc1592] eingesetzt, der Vorgänger des in Abschnitt 3.1.3 beschriebenen **AgentX** Implementierungskonzepts, das die Kommunikation zwischen Master- und Subagenten spezifiziert. Der Zugriff auf die MIB-Variablen erfolgt jeweils über eine Prozedur. Dies bedeutet, daß jede MIB-Variable in einem eigenen Modul implementiert worden ist und über eine bzw. zwei Schnittstellen verfügt, je nachdem, ob auf die Variable nur lesend oder auch schreibend zugegriffen werden kann. Die ausschließlich lesbare MIB-Variable `sysName` wird durch Aufruf der Prozedur get_sysName() ermittelt; eine schreib- und lesbare Variable wie `sysLocation` wird durch die Prozeduren get_sysLocation() und set_sysLocation() implementiert. Insgesamt besteht der SNMP-Agent aus 195 get- und 150 set-Prozeduren. Der ursprüngliche Grund für diese Modularisierung lag in der besseren Erreichbarkeit von Plattformunabhängigkeit, da man zwischen

Betriebssystem-spezifischen Systemaufrufen und Aufrufen, die für alle unterstützten Betriebssysteme identisch sind, besser unterscheiden konnte.

Ergebnis der algorithmischen Transformation: Ein erstes Objektmodell

Der bereits in Abschnitt 4.4.3 vorgestellte JIDM-Algorithmus zur algorithmischen Umformung von SNMP- in CORBA-konforme Objektbeschreibungen bildet die Grundlage für die Gewinnung einer ersten Version des Objektmodells von UNIX-Endsystemen. Hierzu sind folgende Anmerkungen zu machen:

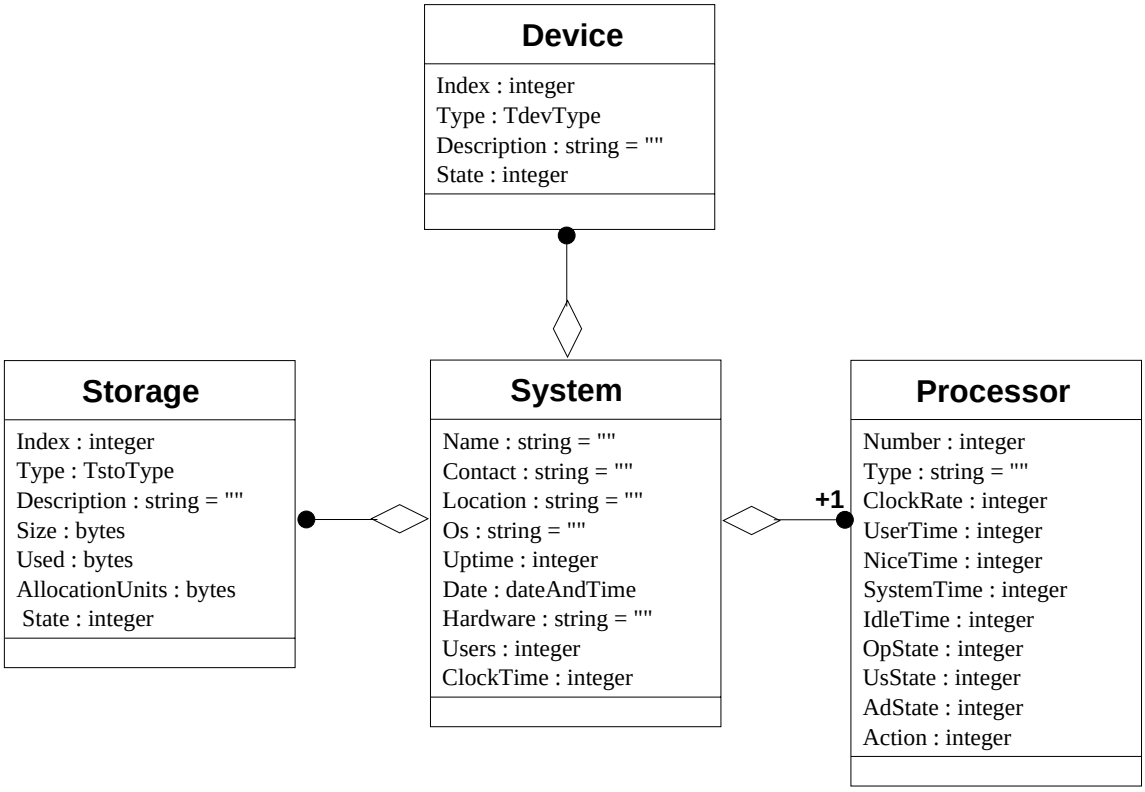


Abbildung 5.3: Algorithmisch erzeugtes Objektmodell in OMT-Notation (Teilansicht)

1. Der JIDM-Algorithmus dient in erster Linie dazu, Managementinformation für Management-Gateways bereitzustellen; er soll einem CORBA-basierten Manager ermöglichen, SNMP-konforme Managementagenten durch ein dazwischen liegendes CORBA/SNMP Management-Gateway derart zu steuern, daß der Manager die SNMP-Agenten als CORBA-Agenten sieht. Dieses Szenario impliziert unter anderem, daß sämtliche zur Verwaltung der SNMP-Information notwendige Daten wie die Indizes der Tabellenzeilen in Attribute der CORBA-Objektklassen umgewandelt werden müssen. Während dies für Management-Gateways zweifellos notwendig ist, ist

ein solches Vorgehen für das objektorientierte Design des Agenten nicht wünschenswert, da beispielsweise die Indizes von SNMP-Tabellenzeilen bereits implizit in den Instanzenidentifikatoren der entsprechenden CORBA-Objektklassen enthalten sind.

2. Die Zielsprache des JIDM-Algorithmus ist, wie oben erwähnt, OMG IDL. Für das weitere Design und die Erweiterung des UNIX-Managementagenten ist es jedoch zwingend erforderlich, die gewonnenen Objektbeschreibungen in eine Notation zu überführen, die es gestattet, die Beschreibungen mit kommerziellen CASE-Tools nachzubearbeiten. Das uns zur Verfügung stehende CASE-Werkzeug, auf dessen Vorteile wir im folgenden Abschnitt genauer eingehen werden, verwendet die **Object Modeling Technique (OMT)** nach Rumbaugh [RBPE 91]. Die Umsetzung der IDL-Syntax in OMT ist daher erforderlich, was jedoch trotz des beachtlichen Umfangs und der damit einhergehenden Komplexität der zugrundeliegenden MIB vollständig automatisierbar ist, da die sogenannte *Code-Reengineering-Komponente* des CASE-Tools in der Lage ist, aus IDL-Klassenbeschreibungen[1] ein OMT-konformes Objektmodell zu generieren. Die Struktur und die Datentypen der Objektklassen sowie deren Attribute sind bereits durch den JIDM-Algorithmus festgelegt.

Ein weiteres Problem besteht darin, daß das Internet-Informationsmodell keine Angaben über die Kardinalität der Beziehungen zwischen einzelnen Objektklassen vorsieht, da es zwar objekt*basiert* ist, nicht jedoch objekt*orientiert*. Demzufolge fehlen Angaben bezüglich vorhandener Enthaltenseinsbeziehungen vollständig, wurden jedoch bereits bei der Überführung in die OMT-Notation von Hand eingefügt. Abbildung 5.3 stellt einen Ausschnitt aus dem so gewonnenen Objektmodell dar.

5.1.3 Gewinnung eines geeigneten Objektmodells

Zusätzlich zu den im vorigen Abschnitt angesprochenen Konvertierungen gibt es natürlich noch eine Reihe offener und für das Design des CORBA-Agenten substantielle Kritikpunkte, die nachfolgend aufgezählt sind. Es sei an dieser Stelle daran erinnert, daß das Ziel der Arbeit darin besteht, einen Agenten zu erhalten, der *möglichst vollständig* objektorientierten Prinzipien entspricht.

- **Zwischen den einzelnen Klassen fehlt (bis auf die Enthaltenseinsbeziehung zur Systemklasse) jegliche Hierarchie**
  Eine Enthaltenseinshierarchie ist nur ansatzweise (eben mit der Systemklasse) realisiert, eine Vererbungshierarchie fehlt gänzlich. Damit sind zwei wesentliche Möglichkeiten der Objektorientierung noch nicht ausgeschöpft. Polymorphismus wird damit zwangsläufig auch nicht unterstützt. Dies sind unmittelbare Konsequenzen aus dem Internet-Informationsmodell, das keinerlei Beziehungen zwischen Objektklassen (weder Vererbung noch Enthaltensein) kennt.

[1] Zusätzlich existieren Reengineering-Module für Programmiersprachen wie C++, Java und Smalltalk.

- **Die noch vorhandenen Typvariablen widersprechen objektorientierten Grundsätzen**
  Beispiele für solche Typvariablen in obigen MIB-Ausschnitten sind `Type` in der `Storage`-Klasse oder `Type` in der `Device`-Klasse. Damit ist es zwar möglich, verschiedene Arten der Objektklasse `Storage` (durch entsprechendes Setzen der Typvariablen) zu bilden, die Objektstruktur ist jedoch für die unterschiedlichen Arten von Speicherobjekten (Hauptspeicher, Festplatten, Magnetbänder, Disketten) dieselbe; die stark differierenden Eigenschaften der einzelnen Typen werden somit nicht berücksichtigt. Hier wird das Fehlen einer Vererbungshierarchie im Internet-Informationsmodell deutlich.

- **Die Problematik der „Pushbutton"-Variablen bleibt bestehen**
  Das Problem, daß die Wertzuweisung an eine Variable unmittelbar eine Operation auslöst, ist auch in diesem ersten Objektmodell vorhanden. Dies resultiert aus der Tatsache, daß SNMP keine Action-Protokolldateneinheit kennt und diese Funktionalität durch die Set-PDU nachgebildet werden muß.

- **Die Variablentypen beschränken sich nur auf ASN.1-Grundtypen**
  Auch Variablen mit stark eingeschränktem Wertebereich, wie z.B. `OpState`[2] oder `AdState`[3] in der Klasse `Processor`, besitzen im ersten Objektmodell einen einfachen integer-Datentyp. Während dieser Punkt auf den ersten Blick als ein „kosmetisches" Problem erscheint, so sind Überlegungen bezüglich der Einschränkung der Wertebereiche von Attributen hinsichtlich späterer Konformitätstests wichtig.

Das objektorientierte Prinzip der Datenabstraktion (*Encapsulation*) ist in diesem Stadium bereits berücksichtigt: Wäre diese erste Version als Basis für die Implementierung herangezogen worden, so hätte der IDL-Compiler Rahmendateien erzeugt, in denen die einzelnen Attribute als private gekennzeichnet, also nur über spezielle get-/set-Operationen (deren Rahmen ebenfalls erzeugt würden) zugreifbar gewesen wären. Es ist somit nicht möglich, auf die Attribute über andere als die bei der Implementierung vorgesehenen Wege zuzugreifen, was dem Konzept der Datenabstraktion entspricht.

CASE-basierte Werkzeugunterstützung

Die im vorangehenden Abschnitt beschriebenen Mängel implizieren massive Eingriffe in die Struktur des ersten Objektmodells. Ebenso sollten zusätzliche neue Anforderungen von Seiten des Betreibers in die Konzeption des verbesserten Objektmodells einfließen. Die damit verbundenen Modifikationen sollten jedoch mit vertretbarem Aufwand für den Entwickler erfolgen; ebenso sollte die Einarbeitungszeit in akzeptablen

[2] `OpState` (Operational State) kann die Werte 1 (enabled) oder 2 (disabled) annehmen; ein Aufzählungstyp reicht hier aus.

[3] `AdState` (Administrative State) kann die Werte 1 (unlocked), 2 (locked) oder 3 (shutting down) annehmen; auch hierfür ist ein Aufzählungstyp besser geeignet.

Grenzen bleiben. Wir haben uns daher entschieden, ein am Markt erhältliches CASE-Tool für das Re-Engineering des Managementagenten einzusetzen, das sowohl konform zur objektorientierten Analyse- und Designmethodik OMT ist als auch die problemlose Übernahme bereits vorliegender IDL-Schnittstellenbeschreibungen gewährleistet. Hinsichtlich der Exportmöglichkeiten besteht ein wesentliches Kriterium darin, daß der Codegenerator des CASE-Tools in der Lage sein sollte, die Objektklassen in der Schnittstellenbeschreibungssprache IDL auszugeben, damit diese anschließend von der CORBA-Entwicklungsumgebung weiterverarbeitet werden können. Die Anforderungen an die Güte der Modellierung werden von kommerziellen CASE-Tools, wie zum Beispiel *Software through Pictures* [StP 34], erfüllt, mit dem die geforderten Optimierungen am Objektmodell in unserem Fall durchgeführt wurden. Hierbei ist hervorzuheben, daß für die unterschiedlichen Phasen der Softwareerstellung leistungsfähige Editoren bestehen, deren Navigationsmöglichkeiten den zyklischen Prozeß der Analyse und des Designs berücksichtigen [Post 94]. Die Erstellung graphischer Modelle und deren Nachbearbeitung wird ebenso unterstützt wie die Dokumentation des Projektes. Features wie die Möglichkeit, Attributen bereits bei der Modellierung Eigenschaften wie „nur lesbar“ oder Default-Werte zuzuweisen, erweisen sich beim Übergang zur Implementierung ebenfalls als vorteilhaft. Insbesondere hilfreich ist die übersichtliche Darstellung des vollständigen Objektmodells; die vergleichbare SNMP-MIB umfaßt demgegenüber 35 Seiten.

Optimierung des Objektmodells

Aus den am Anfang dieses Abschnitts gemachten Beobachtungen lassen sich folgende *Regeln für die Optimierung des Objektmodells* ableiten:

1. In mehreren Klassen vorkommende identische Attribute und Operationen werden in einer (ggf. neu einzuführenden) Superklasse zusammengefaßt.

2. Typenbezeichner werden entfernt. An die Stelle dieser Bezeichner treten neue Subklassen.

3. „Pushbutton“-Variablen werden zu Methoden der jeweiligen Klasse.

4. Durch die Einführung objektorientierter Hierarchien (Vererbung und Enthaltensein) soll der Forderung nach einer möglichst realitätsgetreuen Modellierung von Objektbeziehungen begegnet werden.

5. Es werden neue Variablentypen für Attribute mit bestimmten Wertebereichen (z.B. Aufzählungstypen) definiert.

Die Konsequenzen der Anwendung dieser Regeln sind nachfolgend detailliert beschrieben.

1. Neue Superklassen für gemeinsame Attribute und Operationen

Auffällig an der ersten Version des Objektmodells ist, daß mehrere Klassen Attribute mit identischer Bedeutung (aber zum Teil abweichender Bezeichnung) haben. Stattdessen bietet es sich an, gemeinsame Attribute in eine Superklasse aufzunehmen, betreffende Klassen von dieser Superklasse abzuleiten und dafür diese Attribute in den Subklassen zu streichen. Typische Beispiele für gemeinsame Attribute mehrerer Klassen sind Identifikatoren, Namensbezeichnungen und Statusvariablen. Diese treten u.a. in den Objekten `Printer`, `Storage` und `Processor` auf. Anstatt nun eine neue Superklasse einzuführen, wird der Klasse `Device` (umbenannt in `Generic_Device`) die Rolle der Superklasse zugewiesen. Damit ändert sich die Aufgabe der `Device`-Klasse: Sie dient nun als Wurzel der Vererbungshierarchie mehrerer Systemkomponenten. Dies erleichtert auch spätere Erweiterungen des Objektmodells, bei denen eine neue Systemkomponente von `Generic_Device` abgeleitet werden kann, wodurch schon eine gewisse Grundfunktionalität bereitgestellt werden kann (und zwar die, die in den meisten Komponenten gleich ist).

2. Neue Subklassen anstelle von Typvariablen

Dieser Optimierungsansatz ist die Antwort auf den zweiten Kritikpunkt an der ersten Version des Objektmodells. Durch vorhandene Typvariablen ist es zwar möglich, Objekte verschiedener Typen einer Klasse zu instantiieren. Dies steht jedoch in klarem Widerspruch zum objektorientierten Klassenkonzept, wonach alle Instanzen einer Objektklasse dieselben Eigenschaften aufweisen sollten. Insbesondere ist es innerhalb des vorliegenden Objektmodells also nicht möglich, verschiedenen Typen einer Objektklasse unterschiedliche Eigenschaften (Attribute, Operationen etc.) zuzuweisen, da jedes Objekt der Klasse (gleich welchen Typs) denselben Aufbau hat. So kann man zwar Objekte verschiedenen Typs der Klasse `Device` erzeugen (durch entsprechendes Belegen der Variablen `Type`); für jede dieser Komponenten stünden dann jedoch nur drei Variablen (`Index`, `Description` und `State`) zur Verfügung, die, separat betrachtet, nur über beschränkte Aussagekraft verfügen. Dies ist ein weiterer Grund, weshalb die virtuelle, also nicht instantiierbare Klasse `Generic_Device` zur Wurzel des Vererbungsbaumes wird. Soll nun ein Objekt für eine Systemkomponente erzeugt werden, so wird nicht etwa die `Generic_Device`-Klasse instantiiert, die nun als Containerklasse für allgemeingültige Attribute dient, sondern eine entsprechende Subklasse.

Ähnlich verhält es sich mit der `Storage`-Klasse. Hier können zwar die Typen `Ram`, `VirtualMemory`, `FLCache` und `SLCache`[4] erzeugt werden; es ist jedoch nicht möglich, den einzelnen Typen auch unterschiedliche Eigenschaften zuzuordnen. Deshalb werden die neuen Klassen `permanentStorageDevice` und `volatileStorageDevice` eingeführt. Dabei fallen unter die erste Subklasse Komponenten wie Festplatten, Magnetbänder oder Wechselfestplatten. Die zweite Subklasse

[4] **F**irst **L**evel **Cache** beziehungsweise **S**econd **L**evel **Cache**.

umfaßt diejenigen Komponenten, die ursprünglich über die `Storage`-Klasse instantiiert wurden. Als Folge davon werden die Subklassen `Ram`, `VirtualMemory`, `FLCache` und `SLCache` eingeführt.

3. Operationen anstelle von „Pushbutton"-Variablen

Das Problem der sogenannten „Pushbutton"-Variablen, welches in der SNMP-MIB bestand, ist auch in der ersten Version des Objektmodells (siehe Abbildung 5.3) enthalten: Eine Operation auf einem Objekt (und damit auf der entsprechenden Systemkomponente) muß dadurch ausgelöst werden, daß einem Objektattribut ein Wert zugewiesen wurde. Hinsichtlich der Semantik einer Operation ist dies generell fragwürdig, da auf den ersten Blick nicht zwischen einem Wertattribut und einem „Pushbutton"-Attribut unterschieden werden kann. Beim objektorientierten Ansatz ist dieses Problem jedoch sehr leicht zu lösen: An die Stelle der „Pushbutton"-Variablen treten Operationen. Für jeden möglichen Wert einer „Pushbutton"-Variablen wird dabei eine eigene Operation eingeführt. Ein Beispiel hierfür ist das Attribut `cpuAction` der `Processor`-Klasse. Für die fünf Werte, die dieser Variable ursprünglich zugewiesen werden konnten, werden nun die entsprechenden Operationen enable(), disable(), lock(), unlock() und shuttingdown() eingeführt. Die bisherige Wertzuweisung cpuAction:=1, ausgeführt durch das Senden einer SNMP `set`-Protokolldateneinheit an den Agenten, entspricht nun einem Aufruf der Operation enable(). Da diese fünf Operationen in vielen Komponenten eines Systems ebenfalls auftreten, wurden auch sie in die Klasse `Generic_Device` aufgenommen (siehe Abbildung 5.4).

4. Möglichst realitätsgetreue Modellierung der Objektbeziehungen

Hinsichtlich der Beziehungen zwischen Klassen weist die erste Version der Objektmodells dieselben Defizite auf, wie die SNMP-MIB: Es existiert keinerlei Vererbungshierarchie; Containment-Hierarchien sind nur ansatzweise vorhanden. Ein Grundgedanke des objektorientierten Paradigmas ist es jedoch, die reale Welt, d.h. die einzelnen Objekte sowie ihre Beziehungen zueinander, möglichst gut abzubilden. Die Einführung einer Vererbungshierarchie ist ein Modell für den Sachverhalt, daß eine Systemkomponente eine Spezialisierung einer anderen ist. So ist z.B. ein Mikroprozessor eine Spezialisierung eines Gerätes (`Device`).

Zur Wurzel des Enthaltenseins- bzw. Containment-Baumes wird die Klasse `System` bestimmt. Dies entspricht auch der Realität: Ein (End-)System ist diejenige Hauptkomponente, die andere Teilkomponenten enthält. Dabei sollten Enthaltenseinsbeziehungen der Objektklasse `System` mit möglichst spezialisierten Klassen bestehen, also Klassen, die sich im Vererbungsbaum weiter „unten" befinden. Damit können die einzelnen Beziehungen individuell gestaltet werden: So benötigt ein System mindestens einen Prozessor (hier also eine 1:n-Beziehung mit $n \geq 1$), jedoch kann es beliebig viele `permanentStorageDevices` besitzen (in diesem Fall eine 1:n-Beziehung mit $n \geq 0$).

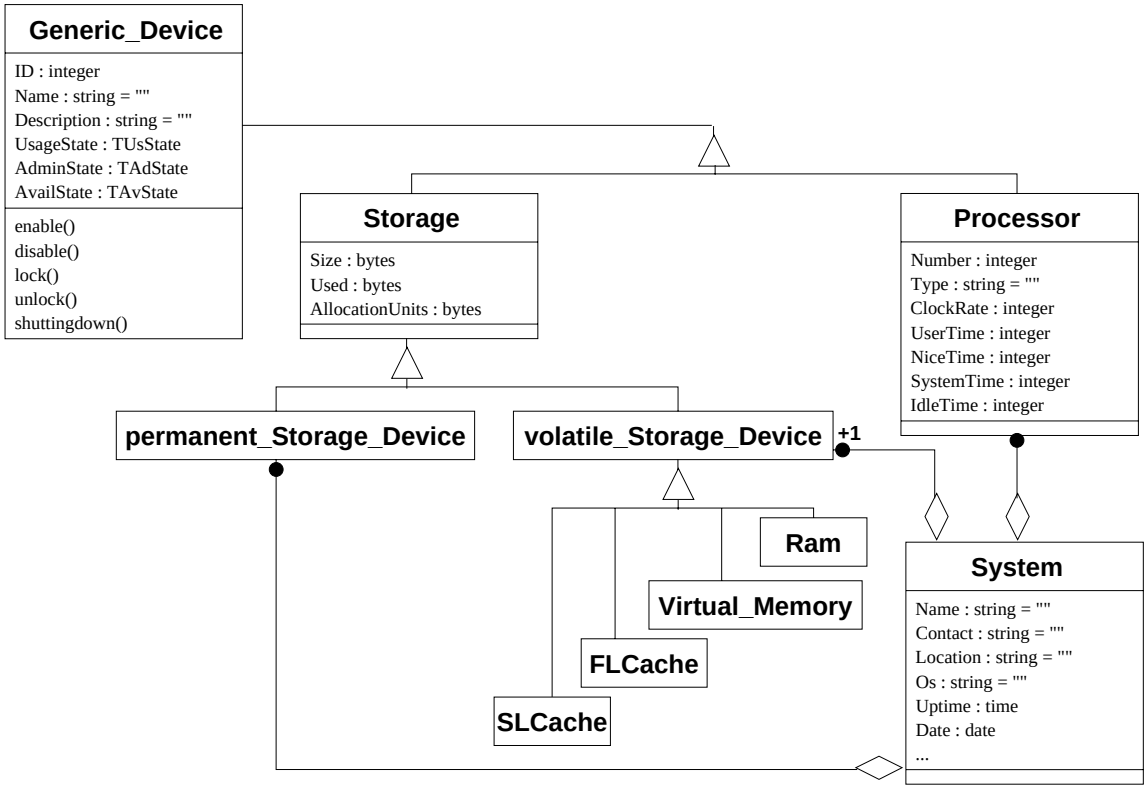


Abbildung 5.4: Optimiertes Objektmodell für das Workstation-Management (Ausschnitt)

Ein Drucker wiederum ist kein unmittelbarer Bestandteil eines Systems (im Sinne einer Workstation), sondern ein Peripheriegerät. Hier besteht also keine Aggregationsbeziehung, sondern eine einfache 1:n-Beziehung mit $n \geq 0$. Wäre die Beziehung zwischen System und „höher“ liegenden Klassen modelliert worden, so wäre eine individuelle Gestaltung nicht möglich gewesen, da jede Subklasse dieselbe Beziehung zu `System` wie die entsprechende Superklasse gehabt hätte. Abbildung 5.4 gibt einen Überblick über die Beziehungsstruktur eines Teils des optimierten Objektmodells.

Eine weitere Besonderheit findet sich in der Beziehung zwischen den Klassen `Filesystem` und `Account` (vormals die `User`-Klasse). Die Klasse `Account` enthält in der SNMP-MIB Attribute, die benutzerspezifische Quoten für Betriebsmittel wie zum Beispiel Plattenplatz darstellen. Dies ist semantisch nicht korrekt, da Quoten keine Eigenschaft eines Benutzers sind, sondern eine Eigenschaft der Beziehung zwischen einem Benutzer und einem Dateisystem. Aus diesem Grund werden die entsprechenden Attribute ausgelagert und unter der Assoziationsklasse `Quota` zusammengefaßt. Abbildung 5.5 veranschaulicht dies.

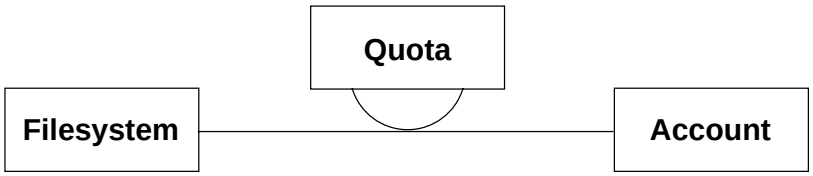


Abbildung 5.5: Anwendung von OMT-Assoziationsklassen

5. Neue Variablentypen für Variablen mit eingeschränktem Wertebereich

Die algorithmische Übersetzung der SNMP-MIB ändert nichts an der Tatsache, daß die meisten Attributtypen im Objektmodell einfache ASN.1-Grundtypen sind. Dies ist in vielen Fällen problematisch: So hat die Variable `OpState` in der Objektklasse `Processor` den Datentyp integer, obwohl sie eigentlich nur die Werte 1 (für enabled) und 2 (für disabled) annehmen darf. Es wäre nun im Rahmen der Optimierung naheliegend, den Datentyp dieser Variablen auf boolean zu setzen. Es erweist sich jedoch als besser, einen (gleichwertigen) Aufzählungstyp zu definieren, der sofort erkennen läßt, in welchem Nutzungszustand sich ein Gerät befindet (enabled oder disabled).

Ein weiteres Beispiel für die Einführung eines neuen Datentyps ist die Variable `AdminState` (vorher `AdState`). Hier wird (aus denselben Gründen wie oben) ein Aufzählungstyp definiert, der die Administrationszustände *unknown*, *unlocked*, *shutting down* und *locked* zuläßt.

5.1.4 Implementierung in CORBA

Da als Implementierungsarchitektur CORBA vorgesehen ist, werden zunächst die dem Objektmodell entsprechenden IDL-Schnittstellenbeschreibungen vom CASE-Tool generiert. Als Beispiel folgt in Tabelle 5.1 ein Auszug aus der Schnittstellenbeschreibung der Objektklasse `System`.

Die Ausgabe verdeutlicht, daß alle mit dem CASE-Tool angegebenen Einstellungen (readonly, Datentypen u.ä.) in die IDL-Ausgabe übernommen werden. Auch ist gut zu erkennen, wie Beziehungen aus dem Objektmodell in IDL-Schnittstellenbeschreibungen abgebildet werden. Dabei wird deutlich, daß diese mit Ausnahme der Vererbungseigenschaften nur mangelhaft in IDL wiedergegeben werden können. Ferner sind sämtliche dynamischen Aspekte, die sich sowohl auf den Daten- als auch auf den Kontrollfluß beziehen, zwangsläufig nicht in IDL abbildbar. Bei der Transformation eines OMT-Objektmodells in IDL gehen somit wichtige Informationen verloren; eine semantische Nachbesserung der erzeugten Schnittstellendefinitionen ist daher unbedingt vonnöten.

Semantische Nachbesserungen bei der Generierung von IDL-Schnittstellen

Die IDL-Klassenbeschreibung gibt die Beziehungen zwischen Objektklassen zwar korrekt wieder; allerdings erscheinen letztere nur indirekt als Attribute der betroffenen Ob-

```
{...}
// stp class definition 108
interface System
{
// stp class members
attribute string Contact;                    // einfache les-
attribute date Date;                         // und schreibbare
attribute string Hardware;                   // Attribute
attribute string Location;
attribute string Name;
readonly attribute string Os;                // nur lesbare
readonly attribute time Uptime;              // Attribute
readonly attribute long maxProcessNumber;
readonly attribute long maxProcessSize;
attribute sequence<Printer> assnPrinter;     // 1:n Assoziation
attribute Process assnProcess;               // einfache Assoz.
attribute sequence<Processor> aggrProcessor; // 1:n Aggregation
...
};
```

Tabelle 5.1: IDL-Schnittstellenbeschreibung der Objektklasse `System` (Auszug)

jektklassen. Dies ist ein charakteristisches Beispiel für den mangelnden Reifegrad gegenwärtiger ORB-Implementierungen, da zur Zeit neue (Hilfs-) Methodenschnittstellen eingefügt werden müssen, die die Gültigkeit der Beziehungen periodisch überwachen. So überprüft beispielsweise die Methode der `System`-Objektklasse update_Processes(), welche Prozesse momentan aktiv sind. Für die Methoden, die ausschließlich der Verwaltung von Beziehungen zwischen Objektklassen dienen, müssen daher neue Methoden entworfen und implementiert werden. Dieses Problem resultiert aus der Tatsache, daß der CORBA *Relationship Service* [OMGSS 97] zwar spezifiziert wurde, aber noch nicht Bestandteil der CORBA-Entwicklungssysteme ist. Ist dies jedoch in zukünftigen Versionen von ORB-Implementierungen der Fall, bedeutet dies eine signifikante Einsparung an Implementierungsaufwand, weil man sich dann vollständig auf bestehende generische Dienste (wie eben den Relationship Service) abstützen kann. Für das Design des Objektmodells bedeutet dies, daß die von der OMG in maschinenlesbarer Form erhältlichen IDL-Schnittstellenbeschreibungen der CORBAservices bereits zu Beginn der Modellierung mit Hilfe der Reengineering-Komponente in OMT überführt werden können und so als generische Objektklassen bereitstehen. Letztere müssen im Falle von Beziehungen lediglich mit konkreten Belegungen spezialisiert werden, um anschließend in IDL ausgegeben zu werden. Die so erzeugten CORBA-Managementobjektklassen machen dann von den bereits bestehenden Diensten Gebrauch.

Die Problematik der Abbildbarkeit von Beziehungen hat aufgezeigt, daß der mit der Umwandlung eines leistungsfähigen OMT-Modells in verhältnismäßig „ausdrucksschwache“ IDL-Schnittstellenbeschreibungen einhergehende Verlust an Semantik durch die geschickte Abstützung auf CORBAservices zu großen Teilen kompensiert werden kann. Langfristig ist es daher keinesfalls nutzlos, hohen Aufwand in die Entwicklung eines reichhaltigen Objektmodells zu stecken, selbst wenn man durch die mittelfristigen Unzulänglichkeiten gegenwärtiger ORB-Implementierungen momentan lediglich einen geringen Anteil der im Objektmodell enthaltenen Semantik erhalten werden kann. Ist ein reichhaltiger Dienstumfang vorhanden, bietet sich die Verwendung des OMT-Nachfolgers **Unified Modeling Language (UML)** an, welche einige Erweiterungen diesbezüglich bereitstellt. In unserem Fall erschien jedoch angesichts der Tatsache, daß bereits der OMT-Funktionsumfang nur zum Teil genutzt werden konnte, der Mehraufwand von UML nicht gerechtfertigt.

Bezüglich des Erzeugens und Löschens von Objekten bzw. der Objektverwaltung allgemein ergibt sich ein weiteres Problem: Es besteht keine zentrale Komponente (eine sogenannte *Object Factory*), über die Objekte einer Klasse erzeugt, gelöscht oder zum Beispiel aufgelistet werden könnten. Eine Lösung für dieses und das obige Problem ist die Einführung von Metaklassen. Zu jeder Klasse existiert damit eine weitere Klasse, von der allerdings nur ein einziges Objekt instantiiert wird und Funktionen bereitstellt, die der Verwaltung von Objekten der Hauptklasse dienen. Da diese Überlegungen rein implementierungsbedingt sind, wird darauf verzichtet, die Metaklassen in das Objektmodell aufzunehmen.

Ein weiteres Beispiel für den Nutzen von Metaklassen wird im Zusammenhang mit der Prozeß-Klasse sichtbar: Generell gilt, daß die CORBA-Managementobjekte beim Systemstart instantiiert werden sollten. Bei statischen Objekten bzw. Objekten, die nur über eine geringe Änderungsdynamik verfügen (wie z.B. Prozessoren, Festplatten) ist dies auch kein Problem. Anders verhält es sich aber mit dynamischen Objekten, wie zum Beispiel Prozessen: Es ist nahezu unmöglich, diese Objekte mit dem realen Systemzustand konsistent zu halten, weil dazu bei jedem Start eines Prozesses ein entsprechendes Prozeßobjekt instantiiert bzw. bei Prozeßterminierung wieder gelöscht werden müßte. Durch den Zugriff auf die Prozeßobjekte über eine sogenannte „before/after“-Metaklasse[5] MetaProcess kann dieses Problem gelöst werden. Sobald nun ein Zugriff auf die Metaklasse durchgeführt wird, wird der Bestand an Prozeßobjekten aktualisiert; das anfragende Objekt (in diesem Fall das Managementsystem) erhält also beim Zugriff immer den aktuellen Systemzustand.

Syntaktische Nachbearbeitung der generierten IDL-Schnittstellen

Als CORBA-Entwicklungsumgebung wird, unter anderem, das IBM *SOMobjects Developer Toolkit* [IBMSOM94] eingesetzt; es umfaßt einen CORBA 1.2-konformen Object Request Broker, einen IDL-Compiler, die CORBA-Laufzeitumgebung und mehrere zu den OMG-Standards konforme Dienste [SOM 96].

[5] Der before/after Metaklassenmechanismus ist jedoch ein proprietäres Merkmal unserer Entwicklungsumgebung und somit nur schwer auf andere ORB-Systeme portierbar.

Die vom CASE-Tool erzeugten IDL-Schnittstellenbeschreibungen können in diesem Fall nicht unmittelbar als Eingabe für den IDL-Compiler eingesetzt werden, da Nachbearbeitungen an folgenden Stellen erforderlich sind:

- **Definition von Attributtypen**
  Datentypen, die nicht zu den IDL-Grundtypen gehören, müssen – sofern sie nicht in der Attributdeklaration festgelegt sind – gesondert definiert werden. Weiter ist zu beachten, daß der SOM IDL-Compiler zwar Sequenzen kennt, oft jedoch nicht den Datentyp, über dem die Sequenz definiert wurde. Dem Auftreten von `sequence⟨Processor⟩` muß die Definition der Objektklasse `Processor` vorangehen.
- **Ableitung aller IDL-Interfaces von SOMObject**
  Die Konventionen des SOM IDL-Compilers verlangen, daß alle auftretenden Schnittstellen von der Objektklasse SOMObject abgeleitet werden.
- **Ergänzung aller IDL-Dateien um eine „Implementation Section“**
  Hierbei muß dem IDL-Compiler mitgeteilt werden, in welche Dynamic Link Library (DLL)[6] die spätere Ausgabe geschrieben werden soll. Außerdem müssen noget- oder noset-Anweisungen für Nur-Lese- bzw. Schreib-Lese-Attribute eingefügt werden, um zu verhindern, daß der Compiler automatisch interne get- oder set-Operationen erzeugt, die den entsprechenden Attributwert liefern bzw. setzen. Dies ist notwendig, da der einzubindende Code bereits existiert.

Es sei an dieser Stelle erwähnt, daß die Notwendigkeit der manuellen Nachbearbeitung von IDL-Schnittstellen *ausschließlich* auf die Unzulänglichkeiten dieses Werkzeuges zurückzuführen ist. Versuche mit neuesten CORBA-Toolkits (OrbixWEB, VisiBroker for Java) haben gezeigt, daß die vom CASE-Tool generierten Schnittstellenbeschreibungen in der Regel problemlos akzeptiert werden. Dieser Teilabschnitt hat folglich anhand eines „worst-case-Szenarios“ aufgezeigt, welcher Anpassungsaufwand durch unausgereifte ORB-Implementierungen entstehen kann.

Übernahme bestehenden Agentencodes

Nachdem die IDL-Beschreibungen der Objektklassen mit ihren Attributen und Methoden erzeugt und an das CORBA-Entwicklungssystem angepaßt sind, muß nun in einem letzten Schritt die Umsetzung der Funktionalität erfolgen, die die Nutzung der Managementinformation erst ermöglicht.

Dies ist gleichbedeutend mit der in zahlreichen Bereichen der Informatik auftretenden Fragestellung, wie bestehende Altsysteme (*legacy systems*) schonend in die objektorientierte Welt migriert werden können. Wir waren diesem Problem bereits in Abschnitt 4.2.4 bei der Nutzung bestehender Plattformdienste von CORBA aus begegnet. Der zentrale

[6] Eine von mehreren Anwendungen nutzbare Programmbibliothek.

Lösungsgedanke ist hierbei, unter Zuhilfenahme sogenannter *IDL-Wrapper* bestehenden Code geeignet in Objektklassen zu kapseln und somit für neue objektorientierte Systeme zugreifbar zu machen. Der betrachtete SNMP-Agent ist in der Programmiersprache C implementiert worden und genügt damit keinesfalls objektorientierten Prinzipien. Dies ist jedoch für die Migration ohne Bedeutung, da die vom Agenten verwalteten MOCs lediglich an ihrer Schnittstelle CORBA-konform sein müssen.

Sehr wichtige Voraussetzungen für eine Kapselung in hinreichend feiner Granularität sind, daß das zu migrierende System genügend modular implementiert worden ist und die Schnittstellen seiner Prozeduren offengelegt sind bzw. die Prozeduren im Quellcode vorliegen. Beides ist in unserem Fall gegeben (siehe Abschnitt 5.1.2).

Die IDL-Schnittstellenbeschreibungen werden nun einem IDL-Compiler übergeben, der zum einen die Schnittstellen der neu erstellten Objektklassen (hier: die *Wrapper* für die bereits bestehenden Prozeduren) innerhalb des ORB-Laufzeitsystems bekanntmacht und zum anderen die Datenstrukturen und Schnittstellen des CORBA-Agenten in einer herkömmlichen Programmiersprache (im betrachteten Fall: C) generiert. Momentan sind Algorithmen zur Übersetzung (sogenannte *Language Mappings*) der Quellsprache IDL in die Zielsprachen C, C++ und Smalltalk durch OMG-Standards spezifiziert; Abbildungen für ADA, COBOL und Java sind ebenfalls in jüngster Zeit verabschiedet worden.

Die so entstandenen C-Programmrümpfe des CORBA-Agenten können nun um die entsprechenden Aufrufe von Prozeduren des bestehenden SNMP-Agenten ergänzt werden, deren Aufgabe lediglich darin besteht, die erhaltenen Parameter auf die Parameter der bestehenden Agentenprozeduren abzubilden und anschließend diese Prozeduren aufzurufen. Hierbei ist es von großem Vorteil, daß der Agent modular aufgebaut ist und die Schnittstellen des Agenten sowohl zu den Ressourcen als auch zum Managementprotokoll hin klar definiert sind.

5.2 Methodik zur Gewinnung generischer MOCs

Wir hatten in Abschnitt 2.3.3 drei unterschiedliche Sichtweisen auf Managementinformation vorgestellt, die für verteilte kooperative Managementsysteme relevant sind und werden, um die starke Abhängigkeit von den Betriebssystem- und Netzdiensten zu betonen, neben den funktionalen, strukturellen und operationellen Sichtweisen auch eine systembezogene Sichtweise einführen. Diese insgesamt vier verschiedenartigen Sichtweisen sind in Abbildung 5.6 dargestellt und lassen sich wie folgt voneinander abgrenzen:

- Die **funktionale Sichtweise** betrachtet ein Managementsystem als installierbares Softwarepaket (z.B. *NetView for AIX*), das sich in einzelne Komponenten aufteilt. Das IBM AIX-Systemadministrationswerkzeug SMIT listet die Bestandteile des o.a. Systems wie folgt auf: (*Base System*, *Database*, *Developer Supplement*, *Online Documentation* usw.) Diese Bestandteile können nun weiter unterteilt werden in die

von einem Managementsystem angebotenen Dienste wie z.B. Konfigurationsmanager, Leistungsmonitor, Topologiemodul, Ereignisverarbeitung, Zustandsmonitor, Oberflächenbaustein, Kommunikationskomponente, MIB-Browser, Informationsverwaltung.

- Sieht man ein verteiltes kooperatives Managementsystem als ausführbares Programmsystem an, d.h. legt man eine **strukturelle** Betrachtungsweise zugrunde, so erscheint dieses als eine Menge von Dateien, die entweder das Managementsystem selbst oder Konfigurationsdateien, Hilfefunktionen sowie Managementdaten enthalten.

- Sobald das Gesamtsystem instantiiert ist, liegt es in Form miteinander kommunizierender UNIX-Prozesse bzw. in Bearbeitung befindlicher Dateien vor. Wir hatten dies in Abschnitt 2.3.3 als die **operationelle** Sichtweise bezeichnet.

- Die **systembezogene** Sichtweise schließlich trägt der Tatsache Rechnung, daß ein Managementsystem nicht isoliert betrachtet werden kann, sondern dessen Funktion stark von Betriebssystem- (Plattenplatz, Zugriffsrechte für Benutzer) und Netzdiensten abhängig ist: Die Prüfung der Netzkonnektivität mittels ICMP, die Abbildung von Namen auf IP-Adressen mittels DNS, die NIS-basierte Benutzerverwaltung sowie der Zugriff auf entfernte Dateisysteme durch NFS erfordert die Verfügbarkeit einer Vielzahl von Netz- und Systemdiensten.

Offensichtlich gelten diese Sichtweisen nicht nur für verteilte kooperative Managementsysteme, sondern auch für eine Vielzahl verteilter Anwendungen und Netzdienste. Daher ist es im Sinne einer Wiederverwendbarkeit von Managementinformation und -diensten angebracht, ein generisches Managementmodell zu entwerfen, das für ein breites Spektrum von Anwendungsklassen gemeinsam verwendbare Managementinformation definiert und durch Vererbungsmöglichkeiten entsprechend der Anwendungsarten verfeinert werden kann. Neben der Berücksichtigung aller vier obengenannten Sichtweisen sollte das Managementmodell ferner von den Spezifika einzelner Managementarchitekturen unabhängig sein und auch keine Annahmen über die zugrundeliegende Kommunikationsinfrastruktur (ORB, Protokoll, Inband- oder Outband-Management, Push- oder Pull-Kommunikation) machen.

Das Ziel unseres Top-down-Vorgehens liegt darin, zuerst generische, d.h. nicht systemspezifische Basisklassen zu gewinnen. Hierzu werden Konzepte des RM-ODP herangezogen, weil dieses Modell ein architekturelles Rahmenwerk für die Entwicklung und Spezifkation von Software in einer offenen, verteilten Systemumgebung darstellt. Da das RM-ODP völlig systemunabhängig ist, aber dennoch die erforderlichen Objekte einer verteilten Systemlandschaft genau beschreibt, werden einige Konzepte und Objekte des RM-ODP unter dem Managementgesichtspunkt benutzt, um generische Basisklassen für das Anwendungsmanagement zu finden. Die gegenwärtige Verflechtung von ODP mit der von uns als Enterprise Management Architektur ausgewählten Common Object Request Broker

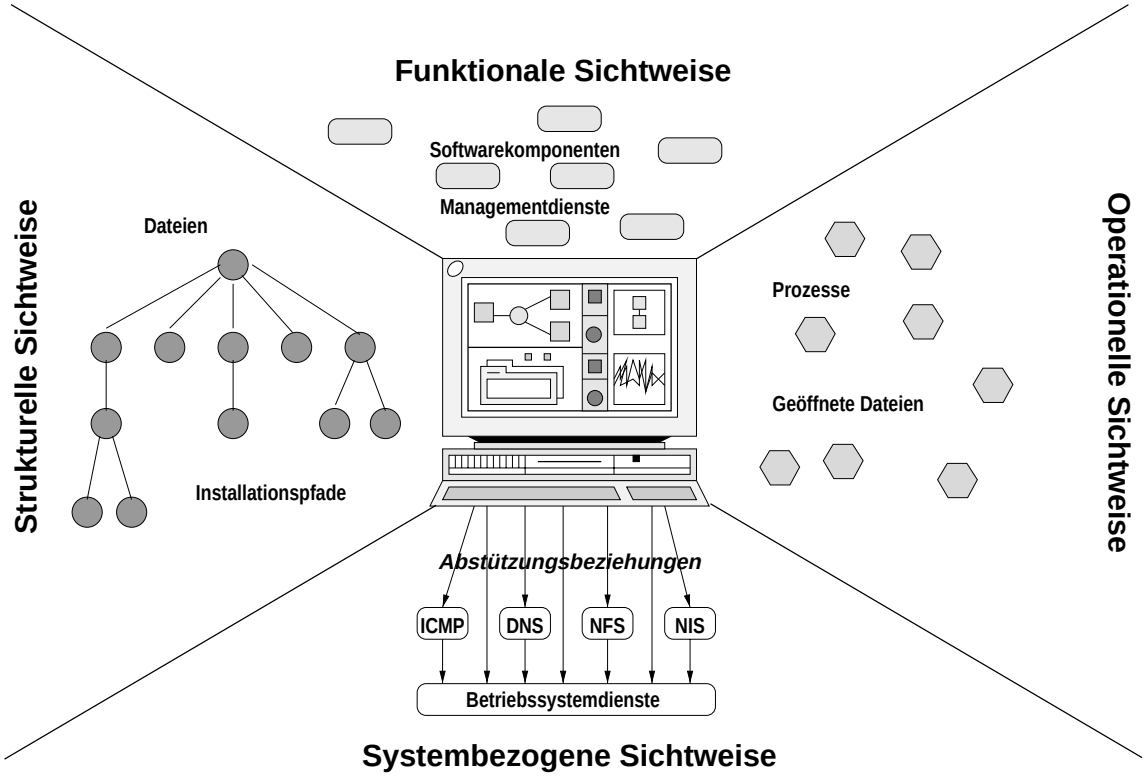


Abbildung 5.6: Sichtweisen auf verteilte kooperative Managementsysteme

Architecture kann als ein weiteres Anzeichen für die Machbarkeit eines durchgehenden Analyse-, Design- und Implementierungsprozesses angesehen werden. Obwohl der Entwurf eines vollständigen Objektmodells für beliebige verteilte Anwendungen und Dienste den Rahmen dieser Arbeit bei weitem sprengen würde, werden wir basierend auf dem RM-ODP, den in Abschnitt 2.3.3 identifizierten Managementanforderungen an verteilte Anwendungen sowie dem Ansatz von Neumair [Neum 99] generische anwendungsbezogene Managementobjektklassen (**Generic Application Managed Object Classes (GAMOCs)**) [KeNe 97c] definieren, die wir in Abschnitt 5.3 als Basisklassen für unsere speziellen MOCs verteilter kooperativer Managementsysteme nutzen werden. Eine wesentliche Anforderung an die Modellierung ist, heterogene Systeme durch ein einheitliches Informationsmodell beschreiben zu können. Hierzu eignen sich objektorientierte Analyse- und Designmethoden sehr gut. Es sollen also Objektklassen gefunden werden, die von herstellerspezifischen, proprietären Details der Ressourcen abstrahieren, aber dennoch ein effizientes Management der Ressource erlauben. Hierzu müssen wir zunächst die GAMOCs in einer Notation beschreiben, die sowohl die Bildung einer Vererbungshierarchie auf einfache Art gestattet als auch eine *Rapid Prototyping*-basierte Implementierung erlaubt. Ausschlaggebend ist hierbei eine gute Werkzeugunterstützung, die eine möglichst automatische Transformation der Modelle auf konkrete Definitions- bzw. Programmiersprachen vornimmt. Unsere im vorangehenden Abschnitt 5.1 beschriebenen positiven Erfahrungen

mit der *Object Modeling Technique* in Verbindung mit dem CASE-Tool *Software through Pictures* haben uns dazu bewogen, unsere Modellierung darauf abzustützen. Das OMT-Modell der GAMOCs kann nun dahingehend verfeinert werden, um die Spezifika verteilter kooperativer Managementsysteme geeignet zu reflektieren. Hierzu ist eine geeignete Spezialisierungshierarchie zu finden, bei der strikte Vererbung anzuwenden ist. Auf diese Weise können, wie wir in Abschnitt 5.1 nachgewiesen haben, auch bereits vorhandene Modelle in das Objektmodell integriert werden. Hierin enthaltene allgemeingültige Attribute und Methoden können gegebenenfalls in die Basisklassen des generischen Modells aufgenommen werden. Die in IDL überführten Schnittstellen der von uns konzipierten MOCs dienen anschließend als Eingabeparameter für eine CORBA-Entwicklungsumgebung, die uns die CORBA-basierte Implementierung der Managementobjekte gestattet. Abbildung 5.7 stellt die Methodik graphisch dar.

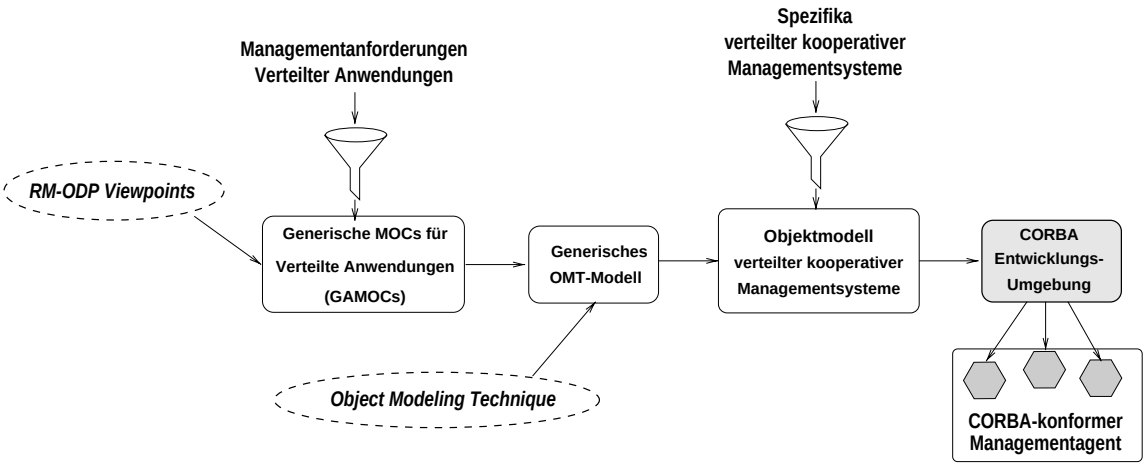


Abbildung 5.7: Überblick über die Methodik

5.2.1 Ableitung der GAMOCs aus RM-ODP

Wie in Abschnitt 3.1.4 ausgeführt wurde, beschreibt das ODP-Referenzmodell für offene verteilte Verarbeitung beschreibt unter Zuhilfenahme verschiedener Viewpoints die Architektur verteilter Systeme. Für den von uns betrachteten Problembereich relevante Konzepte finden sich vor allem im Computational Viewpoint sowie im Engineering Viewpoint. Aus den Konzepten dieser beiden Viewpoints sowie der damit vebundenen Viewpoint Languages können für die meisten der zu managenden Ressourcen generische Managementobjektklassen gewonnen werden. Mit der Identifikation und Beschreibung solcher MOCs befassen sich die folgenden Abschnitte.

Sind Basisklassen gefunden, gilt es, für diese Attribute und Methoden zu definieren, da in ihnen letztlich die Managementinformation bzw. -funktionalität der MOCs enthalten ist. Attribute und Methoden ergeben sich nicht zuletzt aus der in Kapitel 2 durchgeführten

Anforderungsanalyse hinsichtlich der Funktionsbereiche Konfiguration, Fehler, Leistung, Sicherheit und Abrechnung. Die Definition von Attributen und Methoden beinhaltet jedoch grundsätzlich folgenden Zielkonflikt: Wird *zu wenig* Information oder Funktionalität definiert, ist ein effizientes Management der Ressource mit dieser Objektklasse nicht möglich. Eine wichtige Fragestellung ist daher, welche Managementaspekte generell für diese Klasse von Ressourcen gelten. Bei einer *zu detaillierten* Modellierung besteht andererseits die Gefahr, daß nicht alle Ressourcen die durch diese Basis-MOC verlangten Informationen erbringen können. Nicht zuletzt muß darauf geachtet werden, daß die spätere Implementierung des Modells auf einfache Art möglich ist. Die Bereitstellung der für die modellierte Ressource spezifizierten Informationsmenge durch die Instrumentierung muß daher gesichert sein.

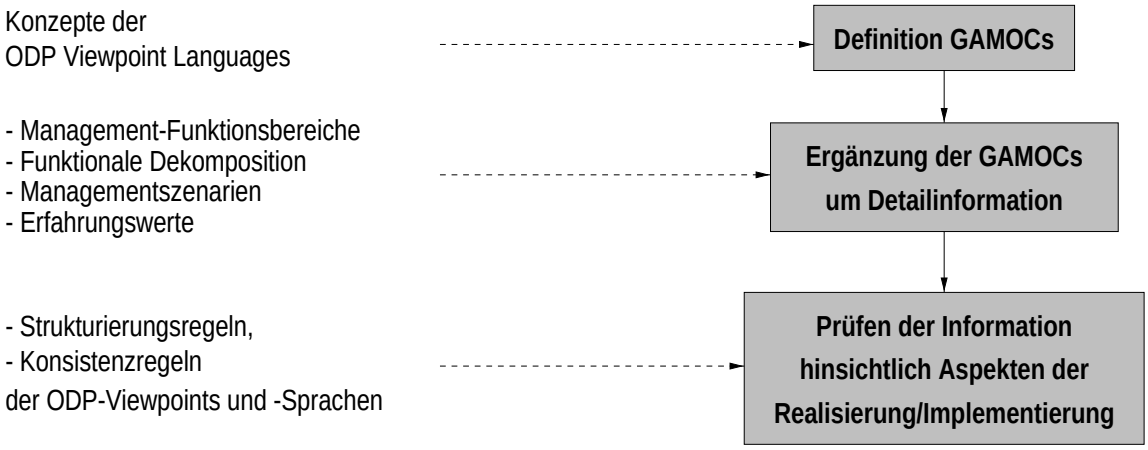


Abbildung 5.8: Der Ansatz von Neumair

Die Definition von Vorgehensweisen zur Vermeidung dieses Zielkonfliktes ist Gegenstand des in Abbildung 5.8 skizzierten Ansatzes von Neumair (siehe dazu auch [Neum 98]), der aus insgesamt drei Teilschritten besteht:

1. **Definition generischer MOCs für Anwendungen (GAMOCs)**. Wie bereits oben angesprochen, müssen zunächst die in RM-ODP definierten Konzepte der ODP Viewpoint Languages ausgewertet werden, d.h. es müssen diejenigen Aspekte verteilter Anwendungen identifiziert werden, die für das Management notwendig sind.

2. **Erweiterung der GAMOCs um managementspezifische Aspekte**. Dies impliziert eine Top-down-Vorgehensweise, die anhand der Management-Funktionsbereiche die Anforderungen an die von den generischen MOCs zur Verfügung gestellte Managementinformation und -funktionalität bestimmt und klassifiziert. Das Ziel dieses zweiten Schrittes besteht somit in der Beantwortung der Frage: „Welche Information bezüglich der Ressourcen ist erforderlich, um ein effektives Management zu gewährleisten?“

3. **Prüfen des Informationsgehaltes der GAMOCs hinsichtlich der Realisierungsaspekte**. Dies impliziert eine Bottom-up-Analyse, deren Ziel in der Ermittlung derjenigen Informationsmenge besteht, die üblicherweise von der Instrumentierung der Ressource bereitgestellt wird. Es geht also um die Beantwortung der Frage: „Wie kann die Durchführbarkeit der im vorigen Schritt identifizierten Managementinformation garantiert werden und welcher Umfang hinsichtlich der Informations- und Funktionsmenge von Ressourcen kann als gegeben vorausgesetzt werden?“

Der Nutzen dieses Ansatzes besteht darin, einen elementaren Informations- und Dienstumfang festzulegen, der für das Management eines breiten Spektrums verteilter Anwendungen gültig ist. Unterstützt wird dies durch die Ausrichtung von RM-ODP als *allgemeingültiges* Rahmenwerk für beliebige verteilte Anwendungen.

Wir werden im folgenden diejenige Managementinformation und -funktionalität vorstellen, die sich aus der Anwendung dieses Ansatzes ergeben. Dabei werden wir zunächst diejenigen GAMOCs bestimmen, die sich aus den Computational und Engineering Viewpoints ergeben.

5.2.2 Computational Viewpoint

Im weiteren werden die auf dem Computational Viewpoint basierenden Objektklassen herausgearbeitet und ihre Bedeutung für das Management erläutert.

Computational Object

Ein **Computational Object** ist eine Komponente einer verteilten Anwendung, die an ihren fest definierten und typisierten Schnittstellen Dienste anbieten oder nutzen kann. Damit eine Interaktion stattfinden kann, müssen die beteiligten Computational Objects eine **Bindung** (s.u.) eingehen. Für das Management betrachten wir ein Computational Object als eine Softwarekomponente, welche zur Laufzeit ein Operation Interface zur Verfügung stellt, über das Managementoperationen aufgerufen bzw. zur Ausführung gebracht werden können. Unter diese Definition fallen auch Agentensysteme eines verteilten Managementsystems. Wir führen daher die MOC **compObject** ein, deren Instanzen die managementrelevanten Abstraktionen benannter Softwarekomponenten darstellen. Ein Beispiel für eine Instanz der Klasse **compObject** ist ein laufendes Managementsystem. Vorerst erhält die Klasse nur das Attribut `Uptime`, welches der Startzeitpunkt der SW-Komponente ist.

Als Superklasse von **compObject** definieren wir die Klasse **object**. In CORBA besitzt jedes Objekt einen eindeutigen Identifikator (`ID`) und kann zu beliebigen Zeitpunkten kreiert oder gelöscht (`create()` und `delete()`) werden. Auf welche Weise dies in einer konkreten Laufzeitumgebung geschieht (z.B. Konstruktoren, *Object Factories*), ist für das Modell unerheblich.

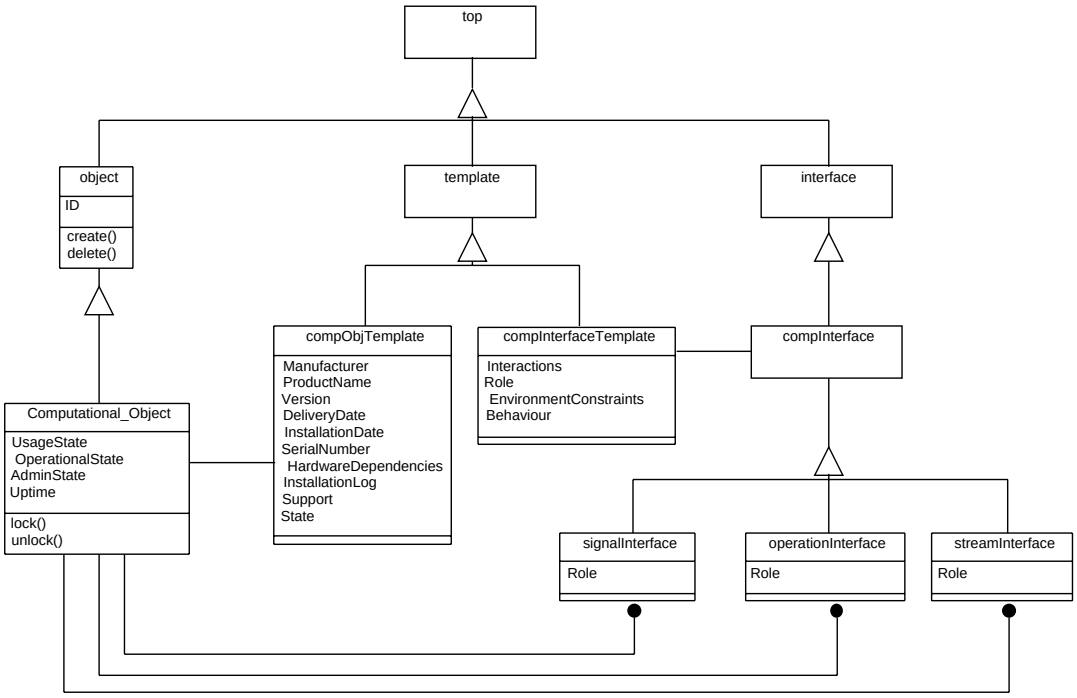


Abbildung 5.9: Funktionale Sichtweise für den Computational Viewpoint

Computational Interface

Wie oben bereits erläutert wurde, sind die Schnittstellen eines Computational Objects diejenigen Zugangspunkte, an denen ein Dienst bereitgestellt bzw. genutzt wird. RM-ODP definiert unterschiedliche Arten von **Computational Interfaces** in Abhängigkeit davon, ob es sich bei den Interaktionen um Signale, Operationen oder Datenströme handelt. Schnittstellen werden daher durch die allgemeine Klasse **interface** repräsentiert. Hiervon abgeleitet wird die Klasse **compInterface** für ein Computational Interface, die wiederum in die drei Klassen **signalInterface**, **operationInterface** und **streamInterface** spezialisiert wird. Letztere werden durch Aggregationsbeziehungen der Klasse **compObject** zugeordnet, um zu verdeutlichen, daß in RM-ODP ein Objekt mehrere (unterschiedliche) Schnittstellen (-Arten) besitzen kann. Die Rolle, die das Objekt hinsichtlich der Interaktionen an einer Schnittstelle einnimmt, wird durch das Attribut `Role` modelliert. Für eine Instanz von **operationInterface** können dies die Werte „Client“ oder „Server“ sein.

Anhand dieser Klassen kann überwacht werden, welche Dienste eine Softwarekomponente anbietet. Somit realisieren sie die **funktionale Sichtweise** auf eine laufende Anwendung. Den entsprechenden Ausschnitt aus dem Objektmodell zeigt Abbildung 5.9.

Computational Interface Template

Laut RM-ODP gehört zu jedem Computational Interface ein *Computational Interface Template*, welches die Eigenschaften der Schnittstelle genau spezifiziert. Das Template beinhaltet die Signatur der möglichen Interaktionen, die Rolle, das Verhalten und die Vor- bzw. Nachbedingungen, die die Schnittstelle an die Umgebung stellt. Im RM-ODP umfaßt das Template alle Informationen, die für das Instantiieren der Schnittstelle zur Laufzeit erforderlich sind.

Für das Modell wird die Klasse **compInterfaceTemplate** mit entsprechenden Attributen eingeführt. Diese geht eine 1:1-Assoziation zur Klasse **compInterface** ein. Für das Management ist wichtig, welche Interaktionen an einer Schnittstelle überhaupt möglich sind. Für das Fehler- und Leistungsmanagement sind zusätzlich die Anforderungen an die Dienstgüte relevant.

Die Attribute, die die Managementinformation bereitstellen, werden anhand eines Operation Interface erklärt. Das Attribut `Interactions` gibt Auskunft über die Operationen, die ein Server an dieser Schnittstelle ausführen kann. Für die GUI-Komponente eines Managementsystems sind dies z.B. die akzeptierten Befehle. Das Attribut `Role` legt fest, ob die Schnittstelle für die Menge der Operationen die Rolle „Client“ oder „Server“ einnimmt. Schnittstellen mit der Rolle „Client“ bezeichnen Dienste, die das Computational Object in Anspruch nimmt, und die deshalb für den ordnungsgemäßen Betrieb der Komponente von anderen Computational Objects im System bereitgestellt werden müssen.

`Behaviour` beschreibt das beobachtbare Verhalten des Objektes in Bezug auf die Operationen der Schnittstelle. Die Beschreibung kann auch Angaben zu internen Aktionen des Objekts und zur erlaubten Aufrufreihenfolge von Operationen enthalten. Das Attribut `EnvironmentConstraints` enthält Dienstgüteanforderungen des betreffenden Objekts und der unterstützenden Umgebung, damit der an der Schnittstelle angebotene Dienst seinerseits einen bestimmten Grad an Dienstgüte bereitstellen kann.

Interaction Information

Netzweite Dienste stellen ihre Funktionalität an einer Schnittstelle zur Verfügung. Ein in unserem Kontext relevantes Beispiel für einen solchen Dienst ist ein SNMP-basierter Managementagent. Dieser besitzt eine Protokollschnittstelle, an der sich Managementsysteme anmelden und authentifizieren können, um danach Managementinformation aus dem hierarchischen Registrierungsbaum auslesen bzw. setzen zu können. Der SNMP-Agent, der hier in der Rolle eines „Managementinformations-Servers“ agiert, instantiiert also ein Operation Interface, welches Interaktionen zwischen dem Client (dem Manager) und dem Server über Operationen wie z.B. `get`, `set`, `get-next`, `get-bulk` und `inform` zuläßt. Durch das Kommando `get` wird zusätzlich ein Stream Interface instantiiert, über welches Managementinformation als Datenstrom (*flow*) übertragen wird.

Für das Management einer SNMP-basierten Ressource sind nun beispielsweise folgende Fragestellungen interessant:

- „Welche Manager haben den Dienst seit einem bestimmten Zeitpunkt in Anspruch genommen?“
- „Wieviele unberechtigte Zugriffsversuche wurden zurückgewiesen?“
- „Welche Datenmenge hat der Agent übertragen?“
- „Welche durchschnittliche Datenrate wurde bei diesen Übertragungen erzielt?“
- „Wie oft kam es zu Abbrüchen von Übertragungen?“

Diese Fragestellungen berühren offensichtlich mehrere funktionale Dimensionen des Managements. Insbesondere ist zu erkennen, daß speziell die Management-Funktionsbereiche Leistung, Abrechnung, Fehler und Sicherheit Informationen über Interaktionen benötigen. Dieses Bedürfnis wird im Objektmodell durch die generische Managementobjektklasse **interactionInfo** berücksichtigt. Diese Klasse definiert Managementinformation, die auf alle Typen von Interaktionen anwendbar ist. Das Attribut `Name` identifiziert die betreffende Interaktion. Im Zähler `Count` wird die Häufigkeit einer bestimmten Interaktion festgehalten, während `Last` den Zeitpunkt des letzten Auftretens enthält. `Bytes` ist ein Zähler für die Anzahl der übertragenen Bytes einer Interaktion. Das Zurücksetzen der Attribute wird durch die Methode `reset()` realisiert.

Über den Typ der Interaktion wird die Superklasse **interactionInfo** weiter spezialisiert. Es entstehen die Subklassen **signalInfo**, **announcementInfo**, **interrogationInfo**, **terminationInfo** und **flowInfo**. Diese spezialisierten MOCs werden durch Assoziationen mit den entsprechenden Typen von Computational Interfaces in Relation gesetzt. Zur Laufzeit können einer Instanz eines Interface – je nach Anzahl der erlaubten Interaktionen – eine oder mehrere Instanzen eines speziellen Interaction-Infos zugeordnet sein. Die OMT-Modellierung der Klassen ist der Abbildung 5.10 zu entnehmen.

Bei einem kontinuierlichen Datenstrom, wie z.B. einer Audioübertragung, sind Echtzeitbedingungen einzuhalten, um einen gewissen Grad an Dienstgüte zu garantieren. Dementsprechend werden in der MOC **flowInfo** u.a. Attribute wie Übertragungsverzögerung (`Latency`), Varianz der Übertragungsverzögerung (`Jitter`), Durchsatz (`Throughput`) und das Verhältnis zwischen Spitzen- und durchschnittlicher Datenübertragungsrate (`Burstiness`) vorgesehen. Auch der Zeitpunkt des Beginns einer Übertragung (`StartTime`) ist von Interesse.

Die Klasse **interrogationInfo** enthält zusätzliche Managementinformation für RPC-ähnliche Interaktionen. Hier sind für das Fehler- und Leistungsmanagement Attribute wie Antwortzeit (`LastDelay`) und Absendezeitpunkt einer Anfrage (`StartTime`) wichtig. Zu einer Anfrage kann es mehrere Typen von Antworten (*terminations*) geben. Natürlich sind auch sämtliche Fehler, die bei einer Anfrage auftreten können, mögliche Antworten. Die Klasse **terminationInfo** ist daher durch die Anforderungen des Fehler- und Sicherheitsmanagements gerechtfertigt. Für die Klasse **signalInfo** gilt dies analog.

Alle oben skizzierten Fragen beim Management einer SNMP-Ressource lassen sich nun durch Auswertung der Managementinformation der eingeführten Klassen beantworten.

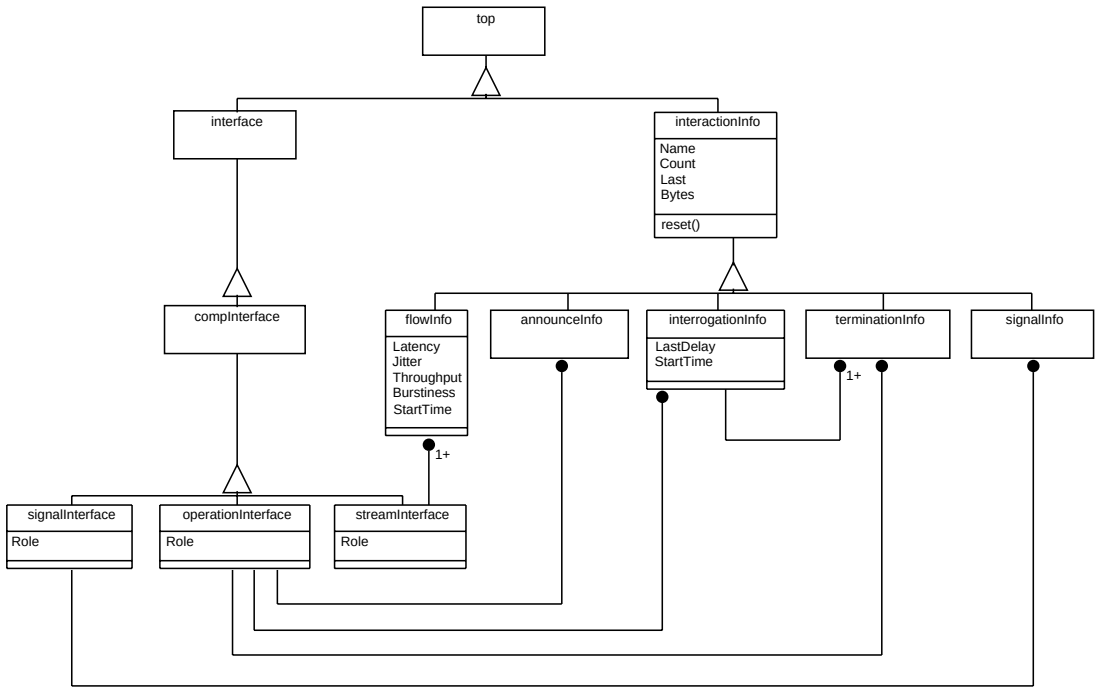


Abbildung 5.10: Computational Viewpoint: GAMOCs zu Client/Server-Interaktionen

Das Attribut `Count` in einer Instanz von **interrogationInfo** für die Operation `open` liefert die Anzahl der Dienstnutzer. Instanzen von **terminationInfo** für die Antworten ≪Access denied≫ und ≪Connection broken≫ geben Auskunft über die Anzahl abgewiesener Anmeldeversuche bzw. unterbrochener Datenübertragungen. Der Mittelwert über das Attribut `Throughput` der Instanzen von **flowInfo** ist die durchnittliche Übertragungsrate.

Gegenwärtig fehlen Attribute, um auch spezielle Interaktionen wie z.B. Transaktionen in einer verteilten Datenbank managen zu können. Dies kann aber durch weitere Spezialisierung der eingeführten generischen Managementobjektklassen leicht erreicht werden. Durch diese Beschränkung wird vielmehr erreicht, daß die MOCs für viele verschiedene Systemdienste verwendbar sind. Außerdem wird deutlich, daß trotz des hohen Abstraktionsgrades viele Aspekte aus den genannten Funktionsbereichen von Managementanwendungen durch die bereitgestellten Informationen abgedeckt werden können.

Computational Object Template

Ein **Computational Object Template** umfaßt laut RM-ODP alle für die Instantiierung des Computational Objects erforderlichen Informationen. Unter Managementgesichtspunkten entspricht ein Computational Object Template einer Softwarekomponente in einem verteilten Rechensystem. Die Instantiierung des Templates entspricht dem Starten der Softwarekomponente und liefert beispielsweise die Bereitstellung eines Systemdienstes.

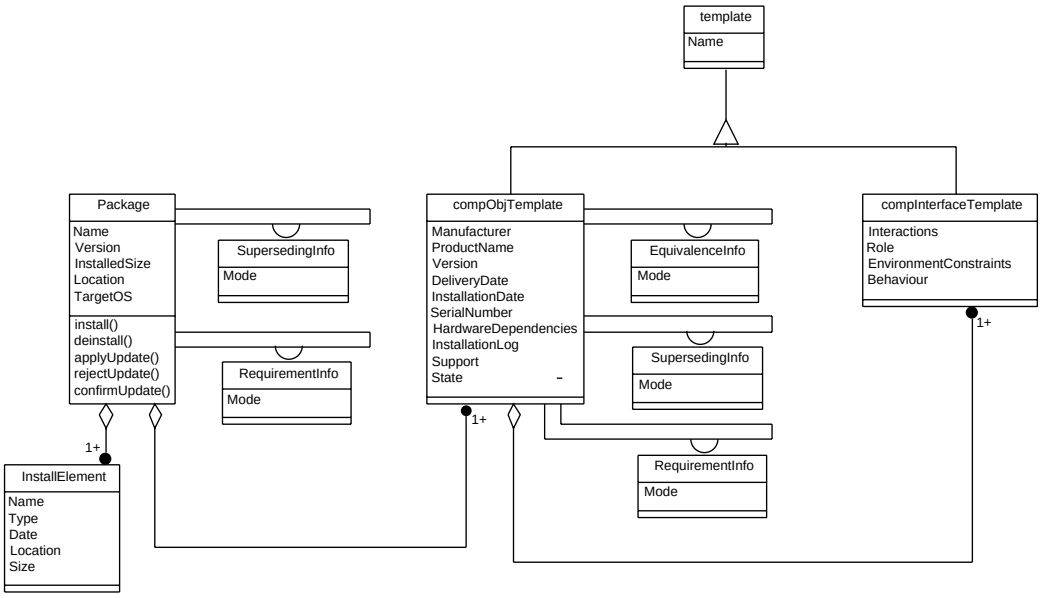


Abbildung 5.11: Strukturelle Sichtweise des Computational Viewpoint

In das Objektmodell wird die Klasse **compObjTemplate** eingefügt, die wichtige Informationen für das Software-Management zur Verfügung stellt. Das Hauptanwendungsgebiet liegt dabei in der Installation von Software-Komponenten, worauf bei der Modellierung daher besonders großer Wert gelegt wird. Die Attribute der MOC **compObjTemplate** (siehe Abbildung 5.11) basieren auf der *Software Standard Groups Definition* der DMTF. Es ist sinnvoll, detaillierte Informationen für das SW-Management bereits bei der generischen Klasse **compObjTemplate** anzusiedeln, da diese Managementinformation unabhängig von der konkreten Ausgestaltung der Software-Komponente ist und von der spezifischen Komponente möglichst abstrahiert werden sollte. Die dabei anfallende Managementinformation wurde aus Gründen, die wir in Kürze erläutern, zusätzlich auf die MOCs **Package** und **InstallElement** aufgeteilt; Assoziationsklassen zur Darstellung der Abhängigkeiten wurden ebenfalls eingefügt.

Im Computational Viewpoint sind verteilte Anwendungen und Systemdienste Objekte, die über wohldefinierte Schnittstellen interagieren, wobei Orts- und Verteilungstransparenz gegeben ist: Es ist nicht Bestandteil des Computational Viewpoint, welche Prozesse und Dateien ein Computational Object in verteilter Umgebung letztendlich realisieren; diese Aspekte sind stattdessen Gegenstand des Engineering Viewpoints. Für das Management durchaus relevante Attribute einer Software-Komponente – wie z.B. Installationsort

oder die zu der Komponente gehörenden Dateien – dürfen daher nicht in die MOC **compObjTemplate** aufgenommen werden. Da jedoch auch der Engineering Viewpoint kein vergleichbares Konzept für ein Software-Paket beinhaltet, wird die Klasse **Package** mit diesen Attributen ins Modell des Computational Viewpoints aufgenommen. Diese Modellierung entspricht auch der Realität, da ein Software-Paket – also eine Instanz der Klasse **Package** – aus mehreren Software-Komponenten, d.h. Instanzen der Klasse **compObjTemplate**, bestehen kann. So enthält beispielsweise ein Paket für den Systemdienst NFS sowohl die Komponenten NFS-Server als auch NFS-Client. Im folgenden gehen wir auf die Attribute und Methoden der betrachteten Klassen ein.

Die Attribute `Manufacturer`, `ProductName`, `Version`, `DeliveryDate`, `InstallationDate`, `SerialNumber` der Klasse **compObjTemplate** sind selbsterklärend. `HardwareDependencies` beschreibt spezifische Anforderungen der Software-Komponente an die Hardware wie Haupt- und Plattenspeicherbedarf oder benötigte Bildschirmauflösung. `InstallationLog` ist der Name des Logs, in dem Meldungen über Ereignisse abgelegt werden, die während der Installation auftreten. `Support` gibt an, auf welche Weise Unterstützung für die Software erlangt werden kann. In `State` könnten Informationen darüber enthalten sein, ob die Installation erfolgreich war und ob die Komponente gestartet werden kann. Software-Komponenten können andere Komponenten erfordern, ersetzen oder zu diesen funktional äquivalent sein, was durch die Beziehungen von **compObjTemplate** mit den Assoziationsklassen **RequirementInfo**, **SupersedingInfo** und **EquivalenceInfo** modelliert wird. Dabei beschreibt das Attribut `Mode` Einschränkungen, welche für die Beziehung gelten: Beispielsweise kann eine Komponente *ab* einer oder *bis* zu einer bestimmten Version erforderlich sein.

Die Bedeutung der Attribute der Klasse **Package** ist ebenfalls klar ersichtlich: `Location` beschreibt die Orte, an denen Teile der Software-Komponenten installiert sind; Software-Pakete können installiert (install()) und deinstalliert (deinstall()) werden. Das Einspielen einer neuen Version eines Pakets kann unter Sicherung des alten Stands durchgeführt werden (applyUpdate()). Anschließend können die Änderungen entweder zurückgenommen (rejectUpdate()) oder endgültig übernommen werden (confirmUpdate()). Auch zwischen Software-Paketen können Abhängigkeiten bestehen, was analog zur Klasse **compObjTemplate** modelliert wird. Ein Software-Paket besteht in der Praxis aus einer Anzahl von Dateien. Im Objektmodell wird dies mit der Aggregationsbeziehung zur Klasse **InstallElement** berücksichtigt. Diese Überlegungen stützen sich auf die bestehenden Möglichkeiten zur Administration von Software-Paketen bei unterschiedlichen UNIX-Varianten: So wird beispielsweise unter IBM AIX die hier beschriebene Managementinformation und -funktionalität durch das Administrationskommando *installp* vollständig abgedeckt und vom AIX-eigenen Systemmanagementwerkzeug *SMIT* bereitgestellt.

Trotzdem ist es denkbar, daß die Attribute und Methoden der beiden Managementobjektklassen **Package** und **compObjTemplate** nicht ausreichend, um alle Szenarien des Software-Managements von einer Managementanwendung aus abzudecken. Andererseits kann jedoch auch nicht erwartet werden, daß eine Anwendung, die auf generischen Ba-

sisklassen arbeitet, die gleiche Funktionalität wie ein proprietäres Spezialwerkzeug bietet. Unbestreitbar lassen sich aber wichtige Basisklassen für das Software-Management aus dem RM-ODP ableiten. Im Computational Viewpoint des RM-ODP werden verteilte Anwendungen und Systemdienste als Computational Objects angesehen, die unabhängig von ihrer Verteilung durch Interaktionen an ihren Schnittstellen kooperieren. Das Modell abstrahiert bewußt von allen Details einer darunterliegenden verteilten Plattform, auf der die Objekte zu Prozessen werden und die Interaktionen über Kommunikationskanäle abgewickelt werden müssen. Konzepte für diese Abstraktionsebene bietet der Engineering Viewpoint, auf den wir nun eingehen werden.

5.2.3 Engineering Viewpoint

Die Prinzipien des Engineering Viewpoints werden ebenfalls in [ISO 10164-2] festgelegt. In ihm beschreibt RM-ODP Infrastrukturobjekte einer offenen, verteilten Systemplattform, welche beispielsweise Kommunikationsmechanismen und Verteilungstransparenzen realisieren. Zu diesen Infrastrukturobjekten, die die eigentlichen Ressourcen eines verteilten Systems darstellen, werden wir nun generische Managementobjektklassen einführen, die uns bei der managementtechnischen Instrumentierung helfen werden. Da sich der Engineering Viewpoint insbesondere mit den Laufzeiteigenschaften einer verteilten Systemplattform beschäftigt, liegt es nahe, hieraus MOCs für das Konfigurations- und Fehlermanagement zu gewinnen.

Im folgenden werden wir generische Managementobjektklassen zum Engineering Viewpoint definieren, die Abstraktionen von Systemressourcen zur Laufzeit darstellen. Aus diesen lassen sich MOCs ableiten, die Managementinformation aus dem traditionellen Bereich des Systemmanagements liefern. Damit können Managementanwendungen auf Systemressourcen wie Prozesse, Geräte und Kommunikationsverbindungen einwirken. Die Notwendigkeit hierfür ergibt sich aus dem erforderlichen Monitoring von Ressourcen für das Leistungs- und Fehlermanagement.

Capsule, Cluster und Basic Engineering Object

RM-ODP bezeichnet einen Prozeß im virtuellen Speicher eines Rechners als **Capsule**; die entsprechende MOC ist die für das Management geeignete Darstellung eines sich in Ausführung befindenden Prozesses unter einem beliebigen Betriebssystem. Die in der Klasse enthaltene Managementinformation ist generisch, damit **Capsule** als Superklasse für speziellere MOCs wie **UNIX_Process**, **OS/2_Process** oder **WinNT_Task** dienen kann: Diese enthält einen (systemspezifischer) Identifikator (`ID`), Angaben über Laufzeit (`ElapsedTime`) und Prozessorzeit (`CPUTime`) des Prozesses, seinen momentanen Zustand (`State`), die Menge des belegten Hauptspeichers (`Memory`) und der Eigentümer (`Owner`). Allgemeine Eingriffsmöglichkeiten auf Prozesse sollten das Stoppen und Starten, das Senden von Signalen und das Beenden zulassen. Diese Eingriffe sind durch die Methoden stop(), resume(), signal() und terminate() vorgesehen.

Laut RM-ODP besteht ein Capsule aus einem bzw. mehreren **Clusters** und diese wiederum aus einem oder mehreren **Basic Engineering Objects**. Diese Enthaltenseinshierarchie spiegelt auch das Objektmodell der MOCs durch Aggregationsbeziehungen wider. Abbildung 5.12 zeigt den entsprechenden Ausschnitt aus dem Objektmodell.

Nucleus

Die Managementobjektklasse **Nucleus** beinhaltet Informationen über das Betriebssystem eines Rechners. Sie dient als Superklasse für MOCs bestimmter Betriebssysteme wie beispielsweise **OS/2_System**, **UNIX_System** oder **WinNT_System**. Ein Rechner wird innerhalb eines Netzes anhand seines logischen (Host-)Namens identifiziert (`Name`). Der Status eines Systems kann über die Attribute `AdminState`, `OperationalState` und `UsageState` ausgelesen werden (vgl. [ISO 10164-2]). Da das Betriebsssystem alle Rechen-, Speicher- und Kommunikationsressourcen eines Rechners verwaltet und somit grundsätzlich verfügbar sein muß, ist der Operational State in der Regel «enabled», der Usage State «active» und der Admin State «unlocked». Während eines Neustarts (restart()) oder eines Systemabschlußes (shutdown()) wechselt der Operational State zu «disabled» und der Admin State zu «shuttingDown» . Außerdem gibt diese Klasse Auskunft über die Systemzeit (`Date`), Name und Version des Betriebssystems (`OS`), Ländereinstellungen (`CountryInfo`) und Zeitzone (`Timezone`) sowie über die Zeitspanne seit dem letzten Neustart (`Uptime`).

Templates für Capsule und Cluster

Grundsätzlich besteht jede Software-Komponente aus einer oder mehreren ausführbaren Dateien. Dies wird im Objektmodell durch eine entsprechende Assoziation von der Klasse **compObjTemplate** zu der neuen Klasse **capsuleTemplate** gekennzeichnet. Laut RM-ODP muß das Capsule Template die zur Instantiierung eines Capsules benötigte Information – die jedoch nicht näher spezifiziert wird – enthalten. Die Instantiierung eines Capsules ist gleichbedeutend mit dem Starten eines Prozesses und gehört zu den Aufgaben des Konfigurationsmanagements. Für die MOC **capsuleTemplate** werden Attribute eingeführt, die speziell Konfigurationsinformationen enthalten: Das Attribut `InstantProcedure` gibt Auskunft über Ort (z.B. Pfadangabe) und Name einer ausführbaren Datei. Eine Programmdatei besitzt einen Typ (`Type`) – binär oder Skript – der ebenfalls Informationen zum Betriebssystem bzw. Prozessor enthält. Über mögliche und sinnvolle Belegungen von Aufrufparametern informiert das Attribut `RuntimeParameter`; `Size` gibt die Größe des ausführbaren Programms an. Ferner wird ein Attribut `RequiredObjects` modelliert, das angibt, welche Objekte zur Ausführung der Datei vorhanden sein müssen. Hierbei kann es sich um Initialisierungs- oder Konfigurationsdateien, um benötigte Bibliotheken oder Systemschnittstellen handeln. Diese Informationen sind nicht zuletzt auch für das Fehlermanagement relevant. Arbeitet ein Systemdienst nicht ordnungsgemäß, kann der

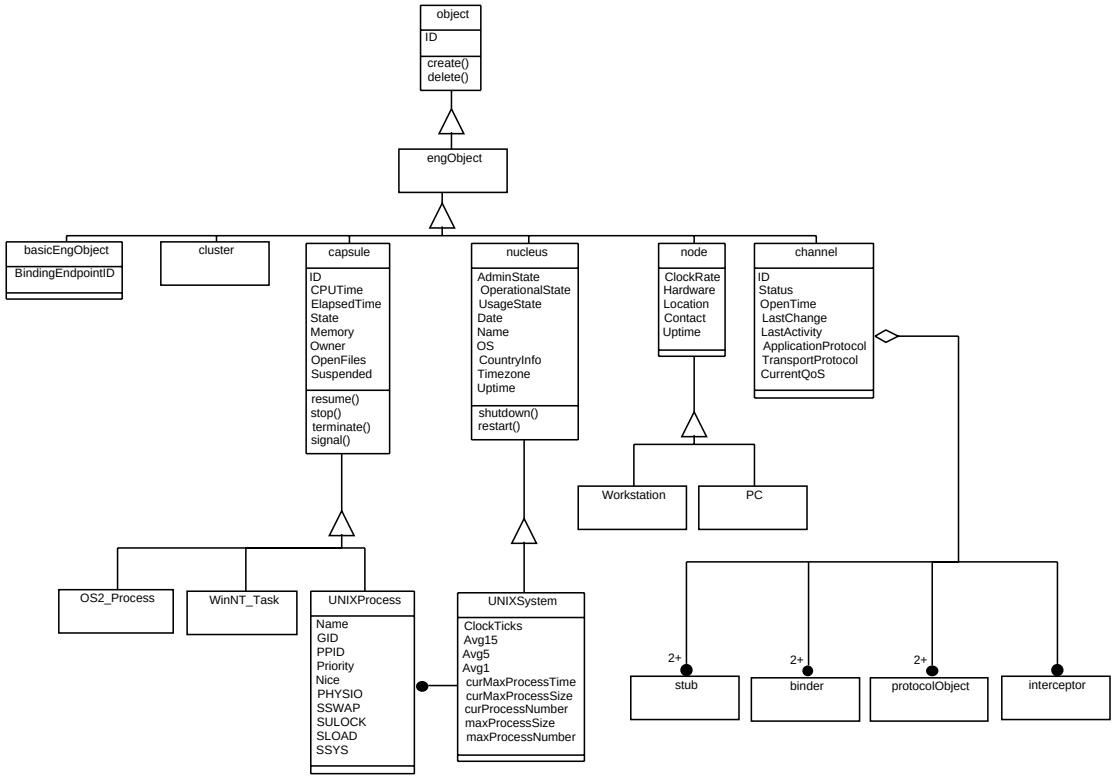


Abbildung 5.12: Funktionale/Systembezogene Sichtweise des Engineering Viewpoint

Administrator überprüfen, ob die zugehörigen Capsules korrekt gestartet wurden und ob alle erforderlichen Objekte in der Umgebung vorhanden sind.

Bei der Modellierung dieser Klasse besteht die Gefahr, die Information auf mehrere speziellere Attribute aufzuteilen und somit die Allgemeingültigkeit der generischen Klasse aufzugeben, weil man Spezifika eines bestimmten Systems oder einer Maschinenarchitektur zu sehr in den Vordergrund stellt: Beispielsweise kann ein Capsule Template für ein ausführbares Programm auf einem Großrechner unter MVS nicht gleich aussehen wie ein Template für eine Batch-Datei unter MS-DOS. Für unterschiedliche Plattformen muß das Capsule Template daher noch weiter spezialisiert werden. Außerdem ist es in einem erweiterten Modell denkbar, das Attribut `RequiredObjects` durch Assoziationen (≪*requires*≫) zu anderen MOCs des Engineering Viewpoint zu ersetzen. Nachfolgend ist ein Beispiel für eine Instanz der generischen Klasse **capsuleTemplate** für den Prozeß ovtrapd angegeben, der für den Empfang und die Verarbeitung von SNMP-Traps in der Managementplattform *HP OpenView* auf einem HP-UNIX System zuständig ist:

Name: ovtrapd

InstantProcedure: /opt/OV/bin/ovtrapd

Type: PA-RISC1.0 shared executable dynamically linked

RequiredObjects: /opt/OV/newconfig/OVEVENT-MIN/lrf/ovtrapd.lrf

RuntimeParameters: none

Size: 36966

Außerdem enhält die Klasse **capsuleTemplate** eine Aggregationsbeziehung zur Klasse **clusterTemplate**. Dies ist die Folgerung aus der Tatsache, daß in RM-ODP ein Capsule aus einem oder mehreren Clusters besteht. Managementinformation für die Klasse **clusterTemplate** ist analog zu **capsuleTemplate** definiert, wobei die Attribute `Size` und `RuntimeParameters` entfallen. Analog zu obigem Beispiel ist dem Capsule Template ovtrapd eine Instanz eines Cluster Template für eine Bibliothek zugeordnet:

Name: libc.sl

InstantProcedure: /usr/lib/libc.1

Type: /usr/lib/libc.1: PA-RISC1.1 shared library - not stripped

RequiredObjects: dynamic loader: dld.sl

An dieser Stelle muß ein wenig von den Festlegungen des RM-ODP abgewichen werden, da die Strukturierungsregeln für Cluster (*cluster rules*) in [ISO 10746-3] festlegen, daß ein Cluster lediglich in exakt einem Capsule enthalten sein darf. Das Konzept der dynamischen, gemeinsam genutzten Bibliotheken (*dynamic shared libraries*), welches in vielen Systemen aus Geschwindigkeitsgründen realisiert ist, verstößt gegen diese Regel. Da wir jedoch das Management heute existierender Systeme betrachten (welche nicht nach den Prinzipien des RM-ODP konstruiert wurden), weichen wir in diesem Fall vom Standard ab.

Node

Während **Nucleus** sich auf das Betriebssystem beschränkt, repräsentiert die Klasse **Node** eine hardwarenahe Sichtweise auf einen Rechner: Ein Node besteht physikalisch aus mehreren Hardware-Komponenten (*devices*) wie Prozessoren, Speicher, Festplatten, etc. Die hierzu eingeführten Attribute sind selbsterklärend und lassen sich grob in die Gruppen ≪MaschineInfo≫ für Maschinenspezifika wie `Manufacturer`, `Model` und `Architecture` und ≪LocationInfo≫ aufteilen. `Location` enthält beispielsweise den Aufstellungsort, der sich in Land, Stadt, Gebäude, Stockwerk, Raum gliedert. Den Namen des zuständigen Administrators enthält das Attribut `Contact`.

Channel

Die Interaktionen zwischen Computational Objects einer Anwendung werden auf einer verteilten Plattform über Kanäle (*Channels*) zwischen den zugehörigen Basic Engineering Objects abgewickelt. Die Managementobjektklasse **channel** ist folglich die Abstraktion

einer Kommunikationsverbindung innerhalb einer verteilten Anwendung bzw. eines verteilten Systemdienstes. Die interne Struktur eines Channels wird auf die Aggregationsbeziehungen zu den MOCs **stub**, **binder**, **protocolObject** und **interceptor** abgebildet. Im einfachsten Fall einer Punkt-zu-Punkt-Verbindung besteht ein Channel aus genau zwei Stubs, zwei Binders und zwei Protocol Objects; bei Multicast-Verbindungen sind dementsprechend mehrere dieser MOCs involviert. Befinden sich die kommunizierenden Objekte in unterschiedlichen administrativen, organisatorischen oder technischen Domänen, kommen noch ein oder mehrere Interceptors hinzu. Stubs, Binders um Protocol Objects werden im RM-ODP eingeführt, um eine Kommunikationsarchitektur zu definieren, die die in Abschnitt 2.3.1 beschriebenen Transparenzen ermöglicht. CORBA kann beispielsweise als die Realisierung einer solchen Architektur angesehen werden. Da RM-ODP jedoch unabhängig von konkreten Systemen ist, wird nicht spezifiziert, *wie* diese Transparenzen realisiert werden bzw. *welche* Funktionen die Objekte implementieren. Darüber hinaus handelt es sich bei einem Channel um ein sehr allgemeines Konzept, mit dem sowohl Interaktionen wie RPCs als auch multimediale Datenströme beschrieben werden können. Daher ist es schwierig, den beschriebenen Objektklassen konkrete Managementinformation zuzuordnen, ohne die Allgemeinheit einzuschränken.

Das Attribut `CommDomain` in der Klasse **protocolObject** gibt die Kommunikationsdomäne (z.B. „OSI“ oder „Internet“) an, in welcher das Protokoll Objekt existiert. Innerhalb einer Kommunikationsdomäne können Protocol Objects ohne dazwischenliegendes Gateway (Interceptor) miteinander kommunizieren.

Die Attribute, die für die Klasse **channel** eingeführt werden, beschreiben Managementinformation für beliebige Arten von Kommunikationsverbindungen. Jeder Channel besitzt einen eindeutigen Identifikator (`ID`), den Zeitpunkt, an dem er kreiert wurde (`OpenTime`) und einen Zustand (`Status`). Mögliche Zustände für einen Kanal sind ≪up≫, ≪down≫, ≪congested≫ oder ≪unknown≫. Bei der Einrichtung eines Kanals wird eine bestimmte Dienstgüte gefordert, die durch die Environment Constraints des zugehörigen Computational Interface Templates festgelegt ist. In der Praxis wird die Dienstgüte jedoch beim Verbindungsaufbau ausgehandelt. Die tatsächliche Dienstgüte eines Kanals kann über das Attribut `CurrentQoS` abgefragt werden. Den Zeitpunkt der letzten Konfigurationsänderung eines Kanals liefert das Attribut `LastChange`, den der letzten Aktivität `LastActivity`. Die Attribute `ApplicationProtocol` und `TransportProtocol` geben Auskunft über die eingesetzten Protokolle der Anwendungs- bzw. Transportschicht.

An einen Channel können mehrere Basic Engineering Objects gebunden sein. Zur Identifizierung der Kommunikationspartner dienen in diesem Fall *Binding Endpoint Identifiers*, die innerhalb eines Prozesses (Capsule) Gültigkeit haben und nichts mit den Engineering Object References zu tun haben. Es ist damit auf Anwendungsebene ein Multiplexen von Verbindungen (*associations*) über einen Kanal möglich, was sich im Modell in Form der Aggregationsbeziehung zur Klasse **Association** niederschlägt. Mögliche Attribute für diese Klasse sind `ID`, `Duration`, `State` und `TransferredBytes`.

Das Ziel der generischen MOCs **channel** und **Association** ist die Unterstützung eines Monitoring von Kommunikationsverbindungen innerhalb verteilter Anwendungen zum Zwecke des Fehler-, Leistungs- und Abrechnungsmanagements. Die Modellierung dieser beiden Klassen greift auf Vorschläge der *Network Services Monitoring MIB* [rfc1565] zurück (siehe auch Abschnitt 3.1.3).

Beziehungen zwischen den Klassen des Computational und Engineering Viewpoint

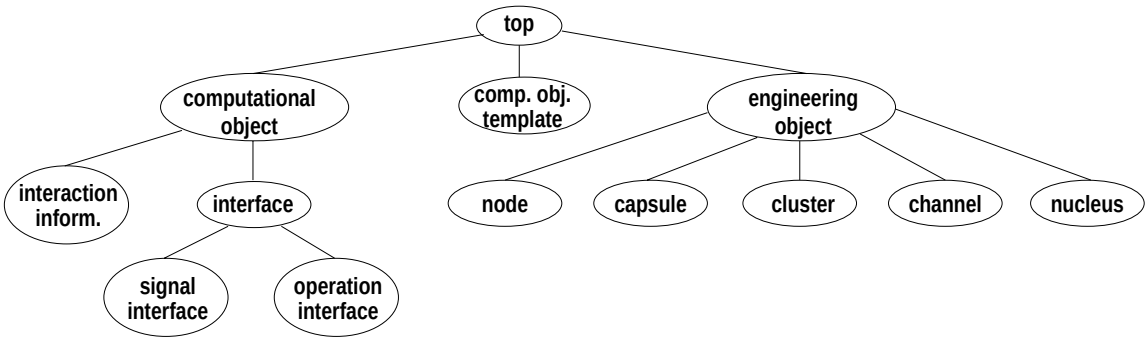


Abbildung 5.13: Generische MOCs als Basis der Vererbungshierarchie

RM-ODP definiert die Abhängigkeiten zwischen den Konzepten verschiedener Sichten durch Strukturregeln der **Viewpoint Correspondences**. Wir gehen an dieser Stelle kurz auf die Beziehungen zwischen Computational und Engineering Viewpoint ein, die ebenfalls durch das Objektmodell wiedergegeben werden sollen.

Grundsätzlich wird ein Computational Object durch mehrere Basic Engineering Objects realisiert. Hierfür gibt es im Modell eine 1:n-Assoziation zwischen der Klasse **compObject** und **basicEngObject**. Da Basic Engineering Objects wiederum in Capsules enthalten sind, kann das Management über die Relationen zwischen den MOCs die zu einer Software-Komponente gehörenden Prozesse ermitteln. Weiterhin legt das Referenzmodell fest, daß ein Computational Interface durch genau ein Engineering Interface realisiert wird. Zur Adressierung eines Engineering Interface bei der Einrichtung eines Kanals dient die eindeutige *Engineering Interface Reference*. Innerhalb der IP-Welt kann beispielsweise ein Tupel, bestehend aus IP-Adresse und Port (*Socket*) bezüglich eines Transportprotokolls (TCP oder UDP) als Engineering Interface Reference angesehen werden. In einer CORBA-Umgebung kommt den IORs der CORBA-Objekte diese Aufgabe zu. Das Objektmodell sieht folglich eine von **interface** abgeleitete Klasse **engInterface** vor, die das Attribut `Reference` besitzt und eine 1:1-Beziehung zu **compInterface** eingeht.

RM-ODP gestattet die Bindung mehrerer Basic Engineering Objects an einen Kanal und damit an ein Engineering Interface. Zur Unterscheidung von Kommunikationsverbindungen einzelner Basic Engineering Objects, die über den gleichen Kanal abgewickelt werden,

dient der sog. *Binding Endpoint Identifier*, der nur innerhalb des Basic Engineering Objects gültig ist. Das Attribut `BindingEndpointID` wird folglich der Klasse **basicEngObject** zugeordnet. In der IP-Welt ist eine Referenz auf einen Socket (File-Deskriptor) ein Beispiel für einen Binding Endpoint Identifier.

Insgesamt haben wir nun zahlreiche GAMOCs aus dem RM-ODP gewinnen können, die allgemeine verteilte Anwendungen entsprechend der zu Beginn dieses Abschnitts besprochenen vier unterschiedlichen Sichtweisen beschreiben. Hinsichtlich einer Top-down-Betrachtungsweise sind sie insofern als vollständig anzusehen, als *sämtliche* Anforderungen des RM-ODP an die Viewpoint-Sprachen ausgewertet wurden. Die Frage nach der erforderlichen Menge an Managementinformation bedingt demgegenüber eine Bottom-up-Analyse möglichst unterschiedlicher verteilter Anwendungen, um sicherzustellen, daß keine Informationen übersehen wurden. Konkret haben wir dabei verteilte Anwendungen wie Managementsysteme sowie Netzdienste wie WWW, NFS und NIS modelliert, um die Informationsmenge der GAMOCs anhand realer Anwendungen zu messen[7]. Dies ist auch notwendig, um feststellen zu können, ob die Instrumentierungen dieser verteilten Anwendungen in der Lage sind, die geforderten Informationen auch tatsächlich zu liefern. Wir haben dazu einerseits bestehende Objektkataloge für Anwendungen aus anderen Managementarchitekturen untersucht (z.B. die Software-Objektkataloge der DMTF und des Internet-Managements sowie die Tivoli Application Management Specification [AMS2.0]) und andererseits die Plausibilität der GAMOCs anhand der verschiedenen von uns entworfenen Objektmodelle geprüft. Hierbei hat sich herausgestellt, daß die in den anwendungsspezifischen MOCs vorhandenen Informationsmengen disjunkt waren. Folglich ist davon auszugehen, daß die in den GAMOCs enthaltene Managementinformation vollständig ist.

Die GAMOCs bilden das Grundgerüst der Basisklassen unserer Vererbungshierarchie und stellen somit bereits eine beträchtliche Anzahl an Managementinformation und -diensten bereit, die grundsätzlich von jeder verteilten Anwendung (und damit auch von verteilten kooperativen Managementsystemen) erbracht werden muß. Der in der Motivation dieser Arbeit vorgetragenen Forderung nach aussagekräftigen Basisklassen wurde hiermit Rechnung getragen. Wir werden im folgenden Abschnitt auf diesen Basis-MOCs unser Objektmodell verteilter kooperativer Managementsysteme aufbauen und auf die Möglichkeiten der Instrumentierung dieser Systeme eingehen.

Ferner werden wir in Kapitel 6 anhand unserer prototypischen Implementierungen den Nachweis führen, daß die von uns gewonnenen GAMOCs auch tatsächlich hinreichend generisch sind, d.h. für eine Vielzahl von Diensten (wie z.B. NFS, NIS und WWW) und Anwendungen (verteilte kooperative Managementsysteme) gelten.

[7] Auf die entsprechenden Prototypen gehen wir in Abschnitt 6.1 ein.

5.3 Objektmodelle von Managementsystemen

Wir werden nun anhand der in Abschnitt 2.3 identifizierten Anforderungen an das Management verteilter kooperativer Managementsysteme und auf der Grundlage der im vorigen Abschnitt 5.2 vorgestellten generischen Managementobjektklassen Management-Objektmodelle für Managementsysteme und Gateways entwickeln. Wir machen uns dabei die Tatsache zunutze, daß die GAMOCs in einer Notation (OMT) vorliegen, die einerseits eine einfache Erweiterbarkeit gestattet und andererseits die Transformation von OMT in die Zielsprache (die OMG Interface Definition Language) durch die von uns genutzte Werkzeugumgebung gesichert ist.

5.3.1 Die spezifischen Objektklassen

Aus den in Kapitel 2 beschriebenen Managementszenarien ergeben sich in einer Top-down-Analyse folgende Objektklassen:

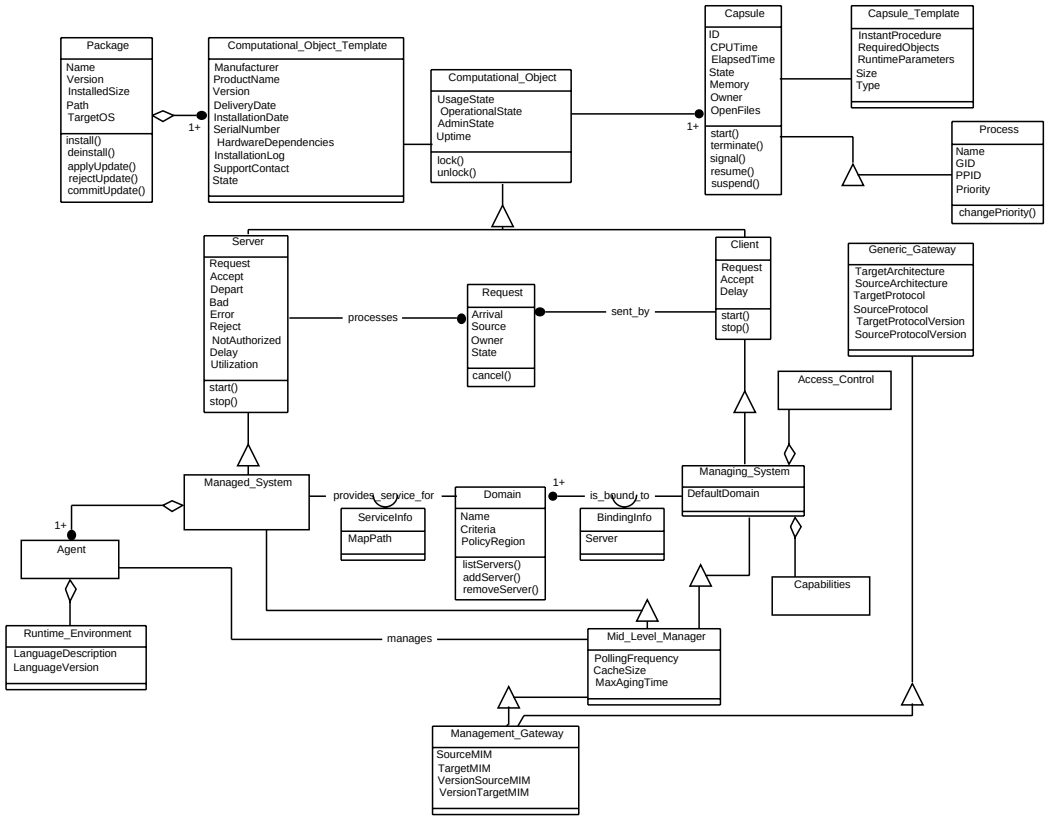


Abbildung 5.14: Objektmodell verteilter kooperativer Managementsysteme

- Um unsere speziellen Objektklassen verteilter kooperativer Managementsysteme in die Begriffswelt des RM-ODP einordnen zu können, müssen wir zunächst eine Ab-

bildung der in der Managementwelt gebräuchlichen Rollen „Manager“ und „Agent“ auf die in verteilten Umgebungen anzutreffenden Rollen „Client“ und „Server“ vornehmen. Hierzu läßt sich feststellen, daß ein Managementsystem grundsätzlich in der Rolle eines Clients agiert, der von einem Server Managementinformationen bezieht bzw. Aktionen darauf ausführt. In der Management-Terminologie agiert ein Managementagent in der Rolle eines Servers. Folglich können wir unsere speziellen Objektklassen **Managed_System** von der generischen Objektklasse **Server** ableiten und die Klasse **Client** als Vaterklasse für **Managing_System** benutzen.

- Die Tatsache, daß wir den Begriff **Managed_System** anstelle von „Agent“ verwenden, liegt darin begründet, daß wir bei modular aufgebauten Agentensystemen eine Unterteilung in Protokoll- bzw. Ressourcenmodule vornehmen müssen. **Managed_System** entspricht somit einem Protokollmodul, das mit einem oder mehreren Ressourcenmodulen (hier als **Agent** bezeichnet) interagiert, was durch eine entsprechende 1:n-Aggregationsbeziehung zum Ausdruck kommt.

- Die Zuständigkeit eines Managers für eine bestimmte Menge von Agentensystemen ergibt sich aus der Aufteilung des Kommunikationssystems in Domänen. Demzufolge werden **Managed_System** und **Managing_System** über eine Objektklasse **Domain** gekoppelt, zu der wiederum jeweils eine Objektklasse **ServiceInfo** bzw. **BindingInfo** Informationen zur Domänenzugehörigkeit von Manager bzw. Agent bereitstellen.

- Aufgrund der Tatsache, daß die Eigenschaft verteilter kooperativer Managementsysteme auch die gegenseitige Überwachung und Steuerung einschließt, muß jedes Managementsystem Informationen mitführen, auf welchen anderen Systemen es Aktionen ausführen darf und umgekehrt. Dies ist der Zweck der beiden Objektklassen **Capabilities** und **Access_Control**.

- Um die Manager/Agent-Doppelrolle von Mid-Level-Managern und Management-Gateways geeignet darzustellen, haben wir die Objektklasse **Mid_Level_Manager** sowohl von **Managing_System** als auch von **Managed_System** abgeleitet.

- Management-Gateways sind andererseits auch ein Spezialfall allgemeiner Gateways, die wir durch die Vererbungsbeziehung zwischen **Generic_Gateway** und **Management_Gateway** zum Ausdruck bringen. Die Manager/Agent-Doppelrolle eines Management-Gateways wird durch die zweite Vererbungsbeziehung zu **Mid_Level_Manager** repräsentiert.

In den folgenden Abschnitten betrachten wir nun anhand der von uns in Abschnitt 5.2 ermittelten vier Sichtweisen auf verteilte kooperative Managementsysteme, inwiefern die notwendige Managementinformation sowie die Dienste auch aus realen Managementsystemen ermittelbar sind. Diese Bottom-up-Analyse ist erforderlich, um eine Aussage hinsichtlich der Realisierbarkeit und praktischen Anwendbarkeit unserer Objektmodelle tref-

fen zu können. Wir werden diese Analyse am Beispiel des Managementsystems IBM NetView in der uns vorliegenden Version 4.1 durchführen.

5.3.2 Funktionale Sichtweise auf Managementsysteme

Wir werden im folgenden die Eingriffsmöglichkeiten in IBM NetView unter funktionalen Gesichtspunkten analysieren, indem wir untersuchen, inwiefern sich das Managementsystem als Menge von Softwarekomponenten und Managementdiensten auffassen läßt. Aus der Instrumentierung von NetView for AIX können wir hinsichtlich der funktionalen Aspekte folgende Managementinformationen gewinnen:

Das Softwarepaket NetView teilt sich in in sogenannte **Filesets** auf, die jeweils inhaltlich zusammengehörige Komponenten gruppieren. Wie aus nachfolgender Aufstellung hervorgeht, können zu jedem Fileset folgende Informationen ausgelesen werden: Der Name des Fileset, sein gegenwärtiger Versionsstand (`Level`) inklusive erfolgter Updates, der Zustand (`State`), der über den Erfolg der Installation Auskunft gibt, sowie eine Kurzbeschreibung der Funktion eines Filesets (`Description`).

```
  Fileset                      Level  State  Description
  ----------------------------------------------------------------------------
  nv6000.base.obj            4.1.1.0    C    NetView for AIX Server - Base
                     4.1.1.0.U441259    C    NetView for AIX Server - Base
                     4.1.1.0.U443133    C    NetView for AIX Server - Base
                     4.1.2.0.U446444    C    NetView for AIX Server - Base
  nv6000.book.En_US.admingd  4.1.2.0    C    Administrator's Guide (lbzl0)
                                              DynaText-U.S. English
                     4.1.2.0.U444131    C    Administrator's Guide (lbzl0)
                                              DynaText-U.S. English IBM-850
  nv6000.database.obj        4.1.1.0    C    NetView for AIX Server -
                                              Database
                     4.1.1.0.U443133    C    NetView for AIX Server -
                                              Database
                     4.1.2.0.U446444    C    NetView for AIX Server -
                                              Database
  nv6000.features.obj        4.1.1.0    C    NetView for AIX Server -
                                              Features
                     4.1.1.0.U443133    C    NetView for AIX Server -
                                              Features
                     4.1.2.0.U446444    C    NetView for AIX Server -
                                              Features
  nv6000.nvbooks             4.1.1.0    C    NetView for AIX Server - Books


State Codes:
 A -- Applied.
 B -- Broken.
 C -- Committed.
 O -- Obsolete.  (partially migrated to newer version)
 ? -- Inconsistent State...Run lppchk -v.
```

Es ist gegenwärtig nicht möglich, hieraus die funktionalen Komponenten eines Filesets zu bestimmen, d.h., die konkreten Managementdienste, die in einem Fileset enthalten sind. Wir müssen diese daher anhand der operationellen Aspekte (vgl. dazu Abschnitt 5.3.4) bestimmen.

Nichtsdestoweniger sind die Abhängigkeitsbeziehungen zwischen einzelnen Filesets ermittelbar. Für das Fehlermanagement lassen sich so wichtige Aussagen über das Vorhandensein notwendiger Softwarekomponenten treffen. Aus der folgenden (gekürzten) Auflistung ist beispielsweise ersichtlich, daß das NetView-Entwicklungssystem (`nv6000.features.obj`), die Datenbank (`database`) und die NetView TMN-Erweiterung (`nvtmnsf`) auf das Basissystem angewiesen sind. Ebenfalls setzen Updates (`nv6000.base.obj 4.1.1.0.U441259`) natürlich das Vorhandensein des Basissystems voraus.

Man erkennt weiterhin, daß auch die Online-Dokumentation (`nv6000.book.En_US.admingd`) ebenfalls in die Versionsverwaltung des ausführbaren Programmsystems aufgenommen ist.

```
  Fileset               Dependents
  ----------------------------------------------------------------------------
<Name> is a requisite of <Dependents>
Path: /usr/lib/objrepos
  nv6000.base.obj 4.1.1.0
                        nv6000.features.obj 4.1.1.0
                        nv6000.database.obj 4.1.1.0
                        nv6000.base.obj 4.1.1.0.U441259
                        nvtmnsf_ci.obj 1.2.0.0
                        nvtmnsf_base.obj 1.2.0.0
                        nv6000.base.obj 4.1.1.0.U443133
                        nv6000.base.obj 4.1.2.0.U446444
  nv6000.book.En_US.admingd 4.1.2.0
                        nv6000.book.En_US.admingd 4.1.2.0.U444131

  nv6000.nvbooks 4.1.1.0
                        NONE

{...}
```

Leider verhindert auch hier die mangelnde Granularität eine umfassende Beschreibung der Abhängigkeiten zwischen einzelnen vom Managementsystem angebotenen Diensten.

Von den Plattformdiensten können sämtliche Funktionen, die dem Operator an der graphischen Benutzerschnittstelle zur Verfügung stehen, auch über eine C-Programmierschnittstelle von Managementagenten genutzt werden (vgl. auch [IBMNVAR 95]). Diese umfassen unter anderem:

- Das Auslösen von SNMP-Operationen auf entfernten Systemen sowie die Konfiguration des SNMP-Subsystems eines Managers (z.B. Domänen, Community Strings, Polling-Intervalle).
- Die Konfiguration von Schwellwerten und die Definition von Regeln zur Ereignisweiterleitung.
- Die (De-)Registrierung von Managementapplikationen (`ovaddobj`, `ovdelobj`).
- Die Definition (`nvsec_admin`) und Überprüfung (`vfy_access`) der Zugriffsrechte auf das Gesamtsystem.

- Die Administration der Zugriffsrechte auf die Topologiedarstellung des Managementsystems: So ist es möglich, einem Benutzer bzw. einer Benutzergruppe einzelne *Maps* zu übereignen (mit den Kommandos ovwchown(), ovwchgrp()), Zugriffsrechte auf Maps anzuzeigen (ovwls()) und zuzuweisen (ovwchmod(), ovwperms()).

Weitere Dienste, die sich mit der Verwaltung des Managementsystems selbst befassen, beinhalten:

- Das Sichern der Datenbank in Dateien sowie das Wiederherstellen der Datenbank.
- Die Prüfung der Datenbank auf Konsistenz.
- Das Aufsetzen von Logs und Traces zur Diagnose von Fehlersituationen.
- Das Starten und Stoppen des Gesamtsystems bzw. einzelner Dienste.
- Das Setzen von Schwellwerten für verfügbaren Platz in Dateisystemen und für den Paging Space. Dies ist ein gutes Beispiel für Dienste, die a priori nichts mit dem Managementsystem selbst zu tun haben, sondern direkt an das Betriebssystem „durchgereicht“ werden.

5.3.3 Strukturelle Sichtweise auf Managementsysteme

Die strukturelle Sichtweise befaßt sich mit den Charakteristiken eines bereits installierten und ausführbaren Managementsystems. Dies beinhaltet insbesondere die Ermittlung, aus welchen Dateien sich ein Managementsystem zusammensetzt und welchen Filesets diese zugeordnet werden können. Ferner sind die Installationspfade der Dateien relevant. Der nachfolgende Auszug aus den insgesamt mehr als 4500 einzelnen Dateien des Gesamtsystems zeigt die Zuordnung einzelner Dateien zu spezifischen Filesets auf.

```
 Fileset               File
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  nv6000.base.obj 4.1.1.0
                        /usr/OV/prg_samples/ovsnmp_app/snmpdemo.c
                        /usr/OV/bitmaps/C/file.50.m
                        /usr/OV/man/man3/OVsnmpXTrapOpen
                        /usr/OV/prg_samples/ovsnmp_app/snmpdemo.h
                        /usr/OV/install/system/OVSNMP-DC/CDFinfo
                        /usr/OV/install/system/OVEMS-PRG-MAN/customize
                        /usr/OV/bitmaps/C/site2.20.p
                        /usr/OV/bitmaps/C/s_lic.26.p
                        /usr/OV/bitmaps/C/fr.32.m
                        /usr/OV/install/filesets
                        /usr/OV/install/system/OVWIN/ovindex
                        /usr/OV/install/system/OVMIN-PRG/ovindex
                        /usr/OV/symbols/C/Connector
                        /usr/OV/install/products/DMRUN/pindex
                        /usr/OV/bitmaps/C/thick_c.44.p
                        /usr/OV/bitmaps/C/inet.26.p
                        /usr/OV/registration/C/ovhelp
                        /usr/OV/bitmaps/C/opsys.44.m
```

```
                        /usr/OV/man/man8/ovdelobj
                        /usr/OV/bin/sortEvents
                        /usr/OV/bin/ovlicense
                        /usr/OV/databases/snmpCollect
                        /usr/OV/databases/openview/topo
{...}
```

Umgekehrt ist es auch möglich, zu einer gegebenen Datei das Fileset zu ermitteln, dem sie angehört.

Ein weiterer wichtiger Aspekt für die Administration des Gesamtsystems besteht darin, die Installationshistorie seiner Softwarekomponenten zu bestimmen und insbesondere Abfragen hinsichtlich ihres Installationsstatus (definitiv installiert; installiert, aber jederzeit rückgängig machbar; fehlerhaft installiert) zu machen. Dies ist mit den nachfolgend aufgeführten Angaben möglich, die neben dem Namen der Komponenten und ihrem Versionsstand insbesondere Informationen hinsichtlich des Installationsstatus umfassen und die einzelnen Schritte mit einem Zeitstempel versehen.

Fehlerhafte installierte Komponenten (Status: `incomplete`) können so jederzeit wieder deinstalliert werden.

```
[TOP]
  Fileset         Level     Action       Status       Date         Time
  -------------------------------------------------------------------------------
Path: /usr/lib/objrepos
  nv6000.base.obj
                  4.1.1.0   COMMIT       COMPLETE     09/04/96     10:04:21
          4.1.1.0.U441259   COMMIT       COMPLETE     08/26/97     11:14:06
          4.1.1.0.U443133   COMMIT       COMPLETE     01/31/97     20:39:27
          4.1.2.0.U446444   COMMIT       COMPLETE     01/31/97     20:49:15

  nv6000.book.En_US.admingd
                  4.1.2.0   COMMIT       COMPLETE     01/31/97     19:51:18
          4.1.2.0.U444131    APPLY     INCOMPLETE     01/31/97     20:39:32
```

5.3.4 Operationelle Sichtweise

Die operationelle Sichtweise betrachtet ein Managementsystem zum Zeitpunkt der Instantiierung, d.h. in Form miteinander interagierender Prozesse und in Bearbeitung befindlicher Dateien. Um Fehlersituationen feststellen zu können, ist es notwendig, den momentanen Zustand der Prozesse zu bestimmen und diese Stoppen bzw. Starten zu können. Der folgende Ausschnitt liefert die hierfür notwendigen Informationen, indem für jeden Prozeß sein Zustand, sein Identifikator und die zuletzt erfolgte Rückmeldung protokolliert werden. Beim Abbruch eines Prozesses wird ferner noch sein Beendigungszustand in Form eines Fehlercodes ausgegeben, der Rückschlüsse auf die Fehlerursache zuläßt.

```
 object manager name: ovwdb
 behavior:            OVs_WELL_BEHAVED
 state:               RUNNING
 PID:                 21938
 last message:        Initialization complete.
 exit status:         -
```

```
 object manager name: trapd
 behavior:            OVs_WELL_BEHAVED
 state:               RUNNING
 PID:                 19906
 last message:        Initialization complete.
 exit status:         -

 object manager name: pmd
 behavior:            OVs_WELL_BEHAVED
 state:               RUNNING
 PID:                 22196
 last message:        pmd completed initialization
 exit status:         -

{...}
```

Insgesamt ist es mit diesen Angaben möglich, sich ein umfassendes Bild vom gegenwärtigen Systemverhalten zu machen, das durch zusätzliche dynamische Informationen wie die Größe des Prozesses und seines aktuellen Rechenzeitverbrauchs wertvolle Hinweise für das Fehlermanagement liefert. Es ist zweckmäßig, auf diesen Basisinformationen Dienste einzusetzen, die die zustandsbehaftete Aufbereitung über frei definierbare Zeiträume gewährleisten: So deutet beispielsweise ein kontinuierliches Anwachsen des Speicherplatzbedarfs eines Prozesses auf vorhandene Programmierfehler hin („Memory Leaks“), was durch eine Neuinitialisierung des Prozesses zumindest für kurze Zeit bis zur endgültigen Problembehebung neutralisiert werden kann.

5.3.5 Systembezogene Sichtweise

Ein besonders wichtiges Mittel zur Fehlerdiagnose und -behebung ist die Beschreibung der Abhängigkeiten zu Systemkomponenten und Diensten, was wir als systembezogene Sichtweise bezeichnen. Wie wir zu Beginn von Abschnitt 5.2 ausgeführt hatten, ist die Funktionsfähigkeit eines Managementsystems zu weiten Teilen von der Verfügbarkeit der zugrundeliegenden Netzdienste abhängig. Ein wichtiger Schritt zur Bestimmung von Fehlerursachen besteht darin, bei Ausfall eines Dienstes gezielt diejenigen Systemdienste (unter Umständen automatisch) zu überprüfen, auf die der ausgefallene Dienst angewiesen ist. Voraussetzung hierfür ist, daß für jeden Managementdienst diese Abhängigkeiten bekannt sind.

Eine vollständige Beschreibung der Dienstabhängigkeiten ist jedoch nicht gegeben, da zwischen Komponenten (*Filesets*), für die solche Beschreibungen existieren und Diensten keine 1:1-Abbildung besteht; vielmehr sind mehrere Managementdienste in einem Fileset enthalten. Der nachfolgende Auszug aus den in der Systemdatenbank enthaltenen Abhängigkeitsbeschreibungen kommt dem Ziel der Dienstabhängigkeiten jedoch schon relativ nahe.

```
  Fileset              Requisites
  ---------------------------------------------------------------------------
Path: /usr/lib/objrepos
  nv6000.base.obj 4.1.1.0
                       At least 1 of the following must be installed {
                         *prereq bos.obj v=3 r=2 o>2 m<9999
                         *prereq bos.compat.links v=4 r=1 m=0 o>0
                       }
                       At least 1 of the following must be installed {
                         *prereq bos.obj v=3 r=2 o>2 m<9999
                         *prereq bos.sysmgt.serv_aid v=4 r=1 m=1 o>1
                       }
                       At least 1 of the following must be installed {
                         *prereq bosnet.snmpd.obj v=3 r=2 p=U428290
                         *prereq bos.net.tcp.client v=4 r=1 o>1
                       }
                       At least 1 of the following must be installed {
                         *prereq bosnet.tcpip.obj v=3 r=2 o>2
                         *prereq bos.net.tcp.client v=4 r=1 m=1 o>1
                       }
                       At least 1 of the following must be installed {
                         *prereq bosnet.nfs.obj v=3 r=2 o>2
                         *prereq bos.net.nfs.client v=4 r=1 m=1 o>1
                       }
                       At least 4 of the following must be installed {
                           *prereq X11rte.obj v=1 r=2 m=3 p=U431144
                           *prereq X11rte.obj v=1 r=2 m=3 p=U432909
                           *prereq X11rte.obj v=1 r=2 m=3 p=U428199
                           *prereq X11rte.obj v=1 r=2 m=3 p=U428198
                         }
{...}
```

So ist zum Beispiel nicht nur ersichtlich, auf welche Betriebssystem- und Anwendungskomponenten (und damit auch Dienste) das Basissystem von NetView `nv6000.base.obj` angewiesen ist, sondern auch deren jeweilige Versionsstände. Im konkreten Fall bestehen Abhängigkeiten der NetView-Dienste zu:

- Dem Betriebssystem selbst: `bos.obj`
- Dem TCP/IP-Protokollstack: `bos.net.tcp.client`
- Dem SNMP-Subsystem: `bosnet.snmpd.obj`
- Dem *Network File System*: `bos.net.nfs.client`
- Dem X-Window System Version 11 als Basis der graphischen Benutzerschnittstelle: `X11rte.obj`

Erste (evtl. automatisch durchgeführte) Fehlerbehebungsmaßnahmen beim Ausfall des Managementsystems könnten daher folgendermaßen ablaufen:

- Prüfen der IP-Konnektivität mittels ICMP.
- Testen des SNMP-Subsystems durch Absetzen eines SNMP-GET-Requests auf die MIB-II des Systems.

- Absetzen von Betriebssystem-Kommandos auf dem entfernten System mittels des UNIX-Kommandos rexec; dabei Zugriff auf ein per NFS gemountetes Verzeichnis.
- Auslesen der Prozeßtabelle des Systems und prüfen, ob alle relevanten Prozesse aktiv sind.

Somit lassen sich die ursprünglich als Hilfsmittel zur Softwareinstallation gedachten Abhängigkeitsbeschreibungen ebenfalls für das Fehlermanagement nutzen. Konkrete Beziehungen zwischen einzelnen Diensten des Managementsystems müssen jedoch vom Benutzer eingetragen werden, da diese a priori nicht ermittelbar sind.

5.4 Zusammenfassung

Wir haben uns in diesem Kapitel eingehend mit dem Design von Management-Objektmodellen für verteilte kooperative Managementsysteme befaßt, die nahtlos in ein CORBA-basiertes Enterprise Management integrierbar sind. Wir haben dabei die folgenden Ziele verfolgt:

- Liegen bereits Managementmodelle und die dazugehörigen Instrumentierungen in anderen Managementarchitekturen (wie dem OSI- oder dem Internet-Management) vor, sollten diese in einem weitgehend automatisierbaren Verfahren in CORBA überführbar sein. Wir haben in Abschnitt 5.1 einen solchen Ansatz konzipiert und seine Machbarkeit anhand eines konkreten Beispiels vorgestellt. Hilfreich war dabei die Abstützung auf den offengelegten JIDM-Algorithmus zur Transformation von Internet-SMI in OMG IDL. Der Einsatz von modernen Software-Werkzeugen wie das von uns verwendete CASE-Tool *Software through Pictures* hat sich dabei als sehr positiv herausgestellt, da es signifikante Vereinfachungen bei der Umwandlung objekt*basierter* Managementinformationsbeschreibungen in ein vollständig objekt*orientiertes* Managementmodell bietet.
- Verfügt eine neue Managementarchitektur noch über keine geeigneten Basisklassen einer Vererbungshierarchie (wie im Falle von CORBA), so ist es möglich, aus den Viewpoint-Definitionen des ODP-Referenzmodells generische Objektklassen abzuleiten, die dies leisten. Unter Verwendung des Ansatzes von Neumair war es uns möglich, diese Basis-MOCs mit der notwendigen Managementinformation auszustatten, damit bereits eine hinreichende Grundmenge an Managementinformation bereits von den Basisklassen zur Verfügung gestellt wird, was einen hohen Grad an Wiederverwendbarkeit auch für andere Anwendungsklassen als die in dieser Arbeit betrachteten Managementsysteme sicherstellt.
- Schließlich haben wir auf der Grundlage dieser generischen Basisklassen für den von uns betrachteten Problembereich geeignete Objektmodelle entworfen, die den in Kapitel 2 herausgearbeiteten Anforderungen gerecht werden.

Das zentrale Designziel bestand darin, offene, standardisierte Verfahren zu verwenden um damit eine breite Anwendbarkeit der Konzepte zu gewährleisten. Hierbei hat sich insbesondere die Verwendung der *Object Modeling Technique* als Notation zur Beschreibung der von uns entworfenen Objektmodelle bewährt, da sie einerseits eine vollständige Unterstützung grundlegender Konzepte der Objektorientierung (wie z.B. Vererbungs- und Enthaltenseinsbeziehungen zwischen Objektklassen) bietet; andererseits ist diese OOA/OOD-Methodik in der Praxis weit verbreitet, was sich im Vorhandensein guter OMT-basierter Software-Entwicklungswerkzeuge wiederspiegelt. Letztere sind in der Lage, aus OMT-konformen Objektmodellen entsprechende Klassendefinitionen für eine Vielzahl gebräuchlicher (Programmier-) Sprachen (C++, Smalltalk, Java, IDL) automatisch zu generieren und umgekehrt mit Hilfe sogenannter *Reengineering-Komponenten* aus bestehendem Programmcode OMT-Diagramme zu erstellen. An dieser Stelle zeigt sich ebenfalls der Vorteil unserer Wahl von CORBA als Architektur für das Management, da das Anwendungsspektrum dieser Architektur über das Management hinausgeht[8]: Die dazugehörige Schnittstellenbeschreibungssprache IDL wird von einem Großteil der am Markt angebotenen CASE-Tools unterstützt, was für die Sprachen spezieller Managementarchitekturen (GDMO/ASN.1 bei OSI/TMN oder eine ASN.1-Teilmenge beim Internet-Management) nicht gegeben ist.

Nicht zuletzt rechtfertigt dies die in Abschnitt 2.3.1 vorgeschlagene Betrachtung des Managements als *einen speziellen Fall einer verteilten Anwendung*.

[8] Schließlich wurde CORBA als Basis für *sämtliche* Arten verteilter Anwendungen konzipiert.

KAPITEL 6

Prototypische Implementierungen

In diesem Kapitel stellen wir einige prototypische Implementierungen vor, die als Tragfähigkeitsnachweis für die im Rahmen dieser Arbeit entwickelten Konzepte dienen. Hierzu werden wir zunächst in Abschnitt 6.1 die Testumgebung vorstellen, in der unsere Prototypen ablaufen, da unsere Implementierungen überwiegend CORBA-basiert sind, bisher jedoch keine CORBA-konformen Managementsysteme existieren. Zur Überbrückung der Heterogenität der involvierten Managementarchitekturen (CORBA, OSI/TMN, SNMP) werden wir dabei auf die in Kapitel 4 vorgestellten Mechanismen und prototypischen Implementierungen für das Umbrella Management zurückgreifen.

Bisher waren generische, wiederverwendbare Managementdienste ausschließlich auf Seiten des Managementsystems implementiert. In Abschnitt 3.2.1 hatten wir das Konzept des *Management by Delegation* vorgestellt, das darauf abzielt, Managementdienste zur Laufzeit von Managern an Agentensysteme zu delegieren, um damit die Verarbeitungslast auf Seiten des Managementsystems zu reduzieren und bereits bei den Agentensystemen die Gewinnung von Managementinformation aus Rohdaten durchzuführen. Die Spezifikation generischer, abgesetzt implementierter Managementdienste in CORBA sowie die Verfügbarkeit portabler Programmiersprachen wie Java legt die Einführung eines *CORBA-basierten Management by Delegation* nahe. Unsere Erfahrungen mit diesem Konzept werden wir kurz in Abschnitt 6.2 darlegen.

Zum Abschluß dieses Kapitels werden wir auf der Grundlage unserer Implementierungserfahrungen ein Konzept aufzeigen, wie gegenwärtige zentralistisch orientierte Managementsysteme in mehreren Schritten schonend zu CORBA-basierten verteilten kooperativen Managementsystemen migriert werden können, um so ein vollständig auf CORBA aufbauendes Enterprise Management zu realisieren.

6.1 Beschreibung der Prototypen

Die im Rahmen dieser Arbeit aufgeworfene Frage der Einsetzbarkeit von CORBA für integriertes Management hat zu Untersuchungen an einer Vielzahl unterschiedlicher Anwendungsfälle geführt, in denen einige Forschungsprototypen entwickelt wurden, von denen wir im folgenden diejenigen vorstellen werden, die in dieser Arbeit entworfene Konzepte und Mechanismen unmittelbar anwenden. Diese umfassen im einzelnen:

- Eine auf der kommerziellen Managementplattform *IBM NetView for AIX* basierende Managementapplikation für das Management verteilter kooperativer Managementsysteme. Es handelt sich hierbei um die Implementierung der in Abschnitt 5.3 konzipierten Objektmodelle.
- Ein Web-basiertes Front-End zur Administration von UNIX-Systemen über CORBA und Java, was der Notwendigkeit der zu Beginn des Abschnitts 5.2 identifizierten *systembezogenen Sichtweise* Rechnung trägt.
- Hierzu zählen ebenfalls die in Java/CORBA implementierten Agenten für das integrierte Management der verteilten Systemdienste *Network File System (NFS)* und *Network Information System (NIS)*, da die Funktion eines Managementsystems unmittelbar von der Verfügbarkeit dieser Dienste abhängt. Dies wurde, unter anderem, in Abschnitt 5.3.5 deutlich. Die Betrachtung dieser Dienste ist auch insofern interessant, als zur Modellierung ihrer Management-Objektmodelle die in Abschnitt 5.2 entwickelte Methodik erfolgreich angewandt wurde. Eine Web-basierte Benutzerschnittstelle gestattet die Administration dieser wichtigen Dienste über gängige WWW-Browser.
- Um die breite Anwendbarkeit des in Abschnitt 5.2 beschriebenen GAMOC-Ansatzes nachzuweisen, wurde darauf aufbauend ein Objektmodell für WWW-Server entwickelt und mit Java/CORBA eine geeignete Applikation sowie entsprechende Managementagenten implementiert.
- Die vielfältigen Einsatzmöglichkeiten des von uns in Abschnitt 5.1 konzipierten Ansatzes zur Transformation bestehender Managementagenten in CORBA-konforme Objekte haben wir anhand eines Agenten für das Management eines ATM-Switches demonstrieren können, dessen proprietäre Managementschnittstelle in OMG IDL überführt wurde. Der Agent besitzt nun eine CORBA-konforme Managementschnittstelle und wird über ein Web-basiertes CORBA/Java Front-End verwaltet.

6.1.1 Vorstellung der Testumgebung

Zunächst stellen wir unsere in Abbildung 6.1 abgebildete Umgebung vor, in der sämtliche in den folgenden Abschnitten beschriebenen Prototypen entwickelt und getestet wurden. Wie aus der obigen Kurzbeschreibung der Prototypen hervorgeht, basieren alle von uns

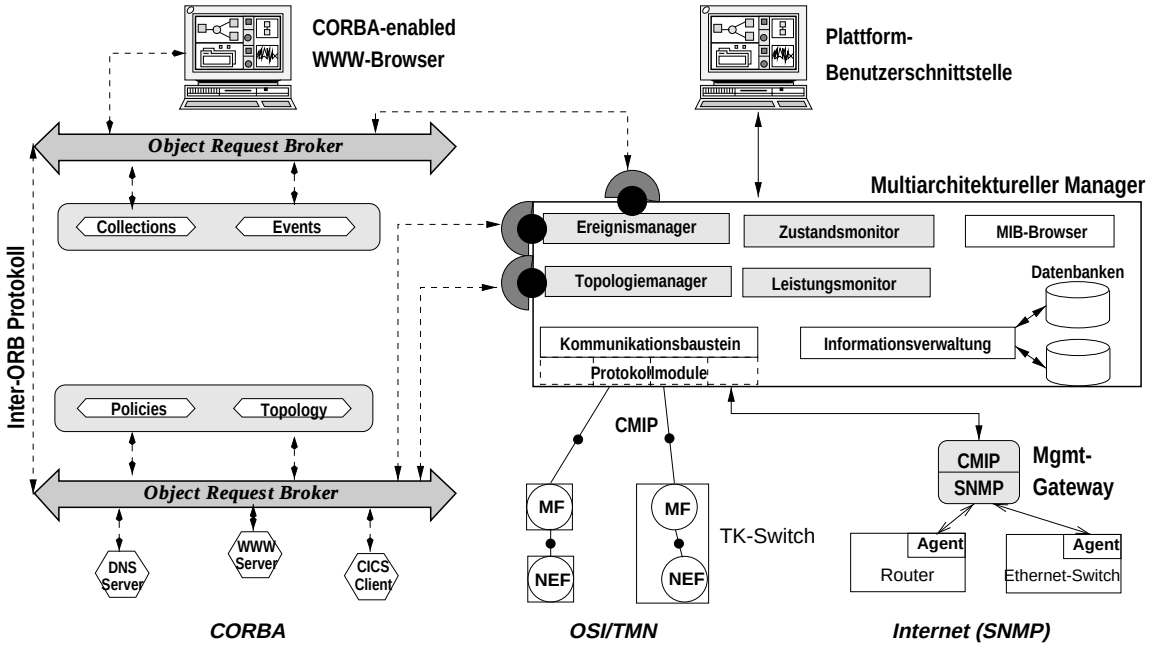


Abbildung 6.1: Überblick über die Testumgebung

implementierten Agenten auf CORBA. Um die von diesen Agenten angebotene Managementfunktionalität optimal nutzen zu können, bedarf es CORBA-basierter Managementapplikationen. Hierfür gibt es zwei Möglichkeiten, die in Abbildung 6.1 dargestellt sind:

1. Sowohl der Manager, als auch der Agent verfügen über einen CORBA 2.0-konformen Object Request Broker, die unmittelbar über ein Inter-ORB-Protokoll miteinander kommunizieren. Dies ist beispielsweise bei sogenannten „CORBA-enabled WWW-Browsern“ (wie z.B. der Netscape Communicator ab Version 4.0) der Fall, die dann in der Rolle eines Clients agieren und von den als CORBA-Objekten vorliegenden Agenten Informationen abrufen bzw. Aktionen darauf ausführen. Die Agenten spielen dann die Rolle von CORBA-Servern. Generische CORBAservices (z.B. Namens- und Ereignisdienste), die im Lieferumfang der ORBs enthalten sind, können hierfür genutzt werden. Dieses Szenario beruht auf der Prämisse, daß eine hinreichende Zahl von CORBAservices für das Management vorliegt, wie sie im linken Teil der Abbildung eingezeichnet sind.

2. Sollen die von kommerziellen Managementplattformen bereitgestellten Dienste wie z.B. Ereignisfilterung und Logging oder das Gruppieren von Managementobjekten (vgl. die in Abschnitt 4.2.4 gemachten Ausführungen) genutzt werden (in der Mitte der Abbildung 6.1), so müssen Übergange zwischen den CORBA-Agenten und der Managementplattform bzw. der Plattform und einer CORBA-basierten Managementapplikation vorhanden sein. Dies ist exakt das Szenario des **multiarchitekturellen Managers**, dessen Architekturkonzept wir in Abschnitt 4.2 vorgestellt haben.

Wie bereits in Abschnitt 4.1 angesprochen wurde, impliziert der Entwicklungsstand heutiger kommerzieller ORBs das Vorhandensein eines nur geringen Umfangs an CORBAservices, so daß sich im ersten Fall die Interaktion zwischen Manager und Agent im wesentlichen auf die Abfrage bzw. das Setzen einzelner MIB-Variablen beschränkt. Ein Benutzer erhält so lediglich rohe Managementdaten statt der gewünschten aussagekräftigen Managementinformationen. Während zwar einerseits Möglichkeiten zur Zustellung asynchroner Ereignismeldungen vorliegen, existieren beispielsweise bisher keine Implementierungen von Diensten zur Filterung und persistenten Speicherung von Ereignissen. Somit wird eine ausschließlich auf CORBA basierende Managementlösung derzeit den Anforderungen eines skalierbaren, integrierten Managements nicht gerecht.

Wir haben uns daher für die zweite Alternative entschieden und nutzen die in Kapitel 4 beschriebenen Prototypen für das Umbrella Management als Basis einer alle drei Managementarchitekturen (CORBA, OSI, Internet) umfassenden, integrierten Managementumgebung. Der Kern dieser Umgebung besteht aus dem SNMP-basierten Managementsystem *IBM NetView for AIX* mit der OSI/TMN-Zusatzkomponente *IBM NetView TMN Support Facility*, das durch die in Abschnitt 4.2 beschriebene Erweiterung um eine CORBA-Komponente alle gängigen Managementarchitekturen abdeckt. Um die aus dem OSI-Management stammenden leistungsfähigen Scoping- und Filteringmechanismen auch auf SNMP-basierte Ressourcen anwenden zu können, nutzen wir das in Abschnitt 4.4.4 entworfene CMIP/SNMP-Gateway zum Zugriff auf SNMP-Agenten, obwohl IBM NetView auch den unmittelbaren Zugriff auf diese Ressourcen gestattet. Dies ist im rechten Teil von Abbildung 6.1 dargestellt. OSI/TMN-Agenten werden unmittelbar über den CMIP-Protokollstack des multiarchitekturellen Managers angesprochen. Insgesamt ist man somit in der Lage, sämtliche von der Managementplattform zur Verfügung gestellten Dienste für das Management von CORBA-, OSI/TMN- und SNMP-Managementagenten zu nutzen.

Als Benutzerschnittstelle steht wahlweise die graphische Benutzerschnittstelle des Managementsystems oder alternativ ein CORBA-fähiger WWW-Browser zur Verfügung.

6.1.2 Management verteilter Managementsysteme

Der von uns entwickelte Prototyp realisiert das in Abschnitt 5.3.1 vorgestellte Objektmodell für das Management verteilter kooperativer Managementsysteme, von dem wir nun einen charakteristischen Teilbereich besprechen. Wir haben dazu das Managementsystem *NetView* geeignet instrumentiert, das sich in der Mitte der Abbildung 6.2 befindet und auf dem UNIX-System ibmhegering1 abläuft.

Unter anderem sind für die operationellen und funktionalen Sichtweisen die Beziehungen zu folgenden Systemen bzw. Komponenten relevant:

- Die Liste der Administratoren des Systems mit ihren jeweiligen Zugriffsrechten, die in einer Tabelle gespeichert ist und über das Icon **NVAdminGroup** ausgelesen und modifiziert werden kann.

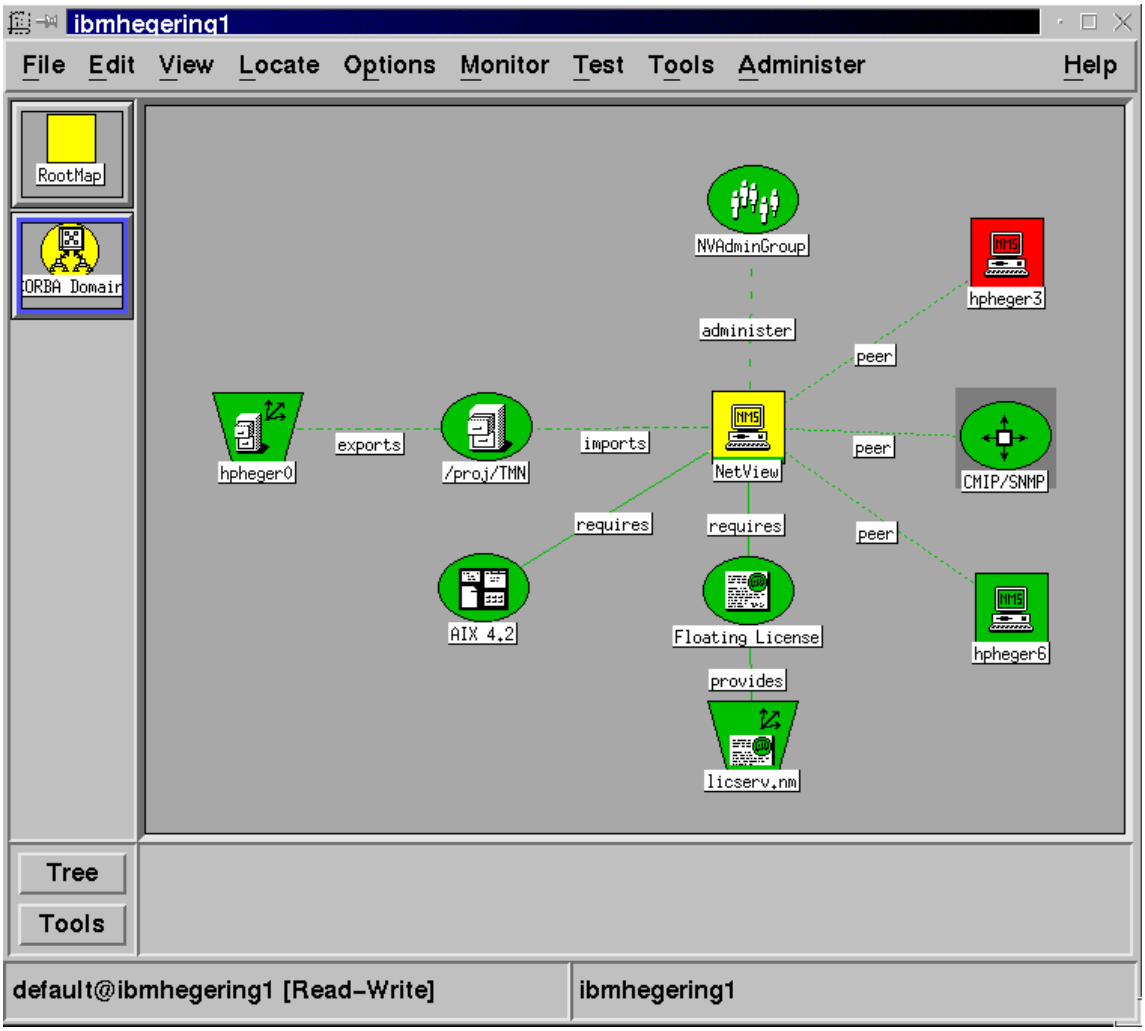


Abbildung 6.2: Beziehungen zu anderen Systemen und Komponenten

- Ferner müssen die Beziehungen des Managementsystems zu Servern dargestellt werden, von denen Dateisysteme importiert werden (im konkreten Beispiel: **/proj/TMN**), da der Ausfall eines Fileservers (hier: das System **hpheger0**), auf dem Daten des Managementsystems abgelegt sind, in der Regel unmittelbar zu einer Beeinträchtigung der Funktionsfähigkeit des Managers führt.

- Außerdem werden wichtige Betriebssystemkenngrößen überwacht und unter einem Icon (**AIX 4.2**) zusammengefaßt. Der Ausfall eines wichtigen Betriebssystemprozesses würde dann eine Farbänderung des Icons nach sich ziehen, um dem Administrator einen Fehlerzustand zu melden. Dieser wird daraufhin eine Analyse der Fehlerursache vornehmen müssen, wofür ihm jedoch ein probates Hilfsmittel zur Verfügung steht: Der zu Beginn des fünften Kapitels entworfene und im Agent für das integrierte Management von UNIX-Systemen gestattet eine weitgehende Überwachung und

Steuerung des Betriebssystems AIX. Wir werden den entsprechenden Prototypen im folgenden Abschnitt 6.1.3 behandeln.

- Wir hatten in der Anforderungsanalyse festgestellt, daß größere Softwarepakete (und damit auch Managementsysteme) überwiegend in Verbindung mit Servern zur Lizenzverwaltung und -überwachung ausgeliefert werden. Demzufolge ist es naheliegenderweise für den Betrieb des Managementsystems umumgänglich, festzustellen, ob beim Start des Systems eine gültige Lizenz vorhanden ist. Hierzu müssen sowohl die Lizenz selbst (hier durch das Icon **Floating License** dargestellt) sowie die dazu gespeicherten Informationen überwacht werden (Art und Anzahl Lizenzen, Ablaufdatum, Produktumfang usw.) als auch die Funktionsfähigkeit des entsprechenden Lizenzservers.
- Schließlich ist es für ein kooperatives Managementsystem wichtig, zu wissen, welche Partner-Managementsysteme existieren und welche Arten von Interaktionen jeweils gestattet sind. Im vorliegenden Fall handelt es sich um die HP OpenView-Managementplattformen auf den Maschinen **hpheger3** und **hpheger6** sowie das CMIP/SNMP Management-Gateway, das die SNMP-Domäne administriert.

Die Beziehungen des Managementsystems **NetView** zu den anderen Systemen bzw. Komponenten sind zur besseren Übersichtlichkeit jeweils mit Namen versehen, die über die Art der Beziehung (peer, requires, provides usw.) Auskunft geben. Die Beziehungen selbst besitzen jedoch keine Managementinstrumentierung, da die bei der Implementierung verwendete CORBA-Umgebung *IBM SOM/DSOM* keine Implementierung des CORBA Relationship Service beinhaltet.

In Abbildung 6.2 ist unter anderem aus der hellen (gelben) Einfärbung des Systems **NetView** ersichtlich, daß ein (nicht besonders schwerwiegender) Fehlerzustand aufgetreten ist, der bis an die Spitze der Topologiehierarchie (am linken Rand ersichtlich) propagiert wird. Ein Grund hierfür könnte in der Tatsache bestehen, daß zugleich das Partner-Managementsystem **hpheger3** dunkel (rot) eingefärbt ist, was auf einen vollständigen Ausfall dieses Systems hindeutet. Zur Eingrenzung des Fehlers sind einerseits die eingehenden Ereignismeldungen wichtig und andererseits die Zusatzinformationen, die sich in der über das Icon zugänglichen, darunterliegenden Ebene verbergen. Abbildung 6.3 verdeutlicht, daß dort Informationen über den aktuellen Zustand aller Dienste ablesbar sind, aus denen das Managementsystem besteht: Hieraus geht hervor, daß die Prozesse **snmpCollect** und **netmon** ausgefallen sind (Farbe: rot), die für das Polling der Ressourcen sowie die Entgegennahme der Resultate zuständig sind. Ferner bedeutet die blaue Farbe der Prozesse **nvcorrd** und **nvsecd**, daß diese Dienste bisher noch nicht instantiiert wurden. Aus der grünen Einfärbung der anderen Prozeßsymbole kann gefolgert werden, daß die restlichen Dienste des Managementsystems aktiv sind. Anzumerken ist hierbei, daß ein Dienst jeweils drei Zustände annehmen kann (aktiv, inaktiv, bisher nicht instantiiert). Eine feingranularere Zustandsbeschreibung der einzelnen Dienste läßt sich mit den vom Betriebssystem angebotenen Systemaufrufen derzeit nicht ermitteln.

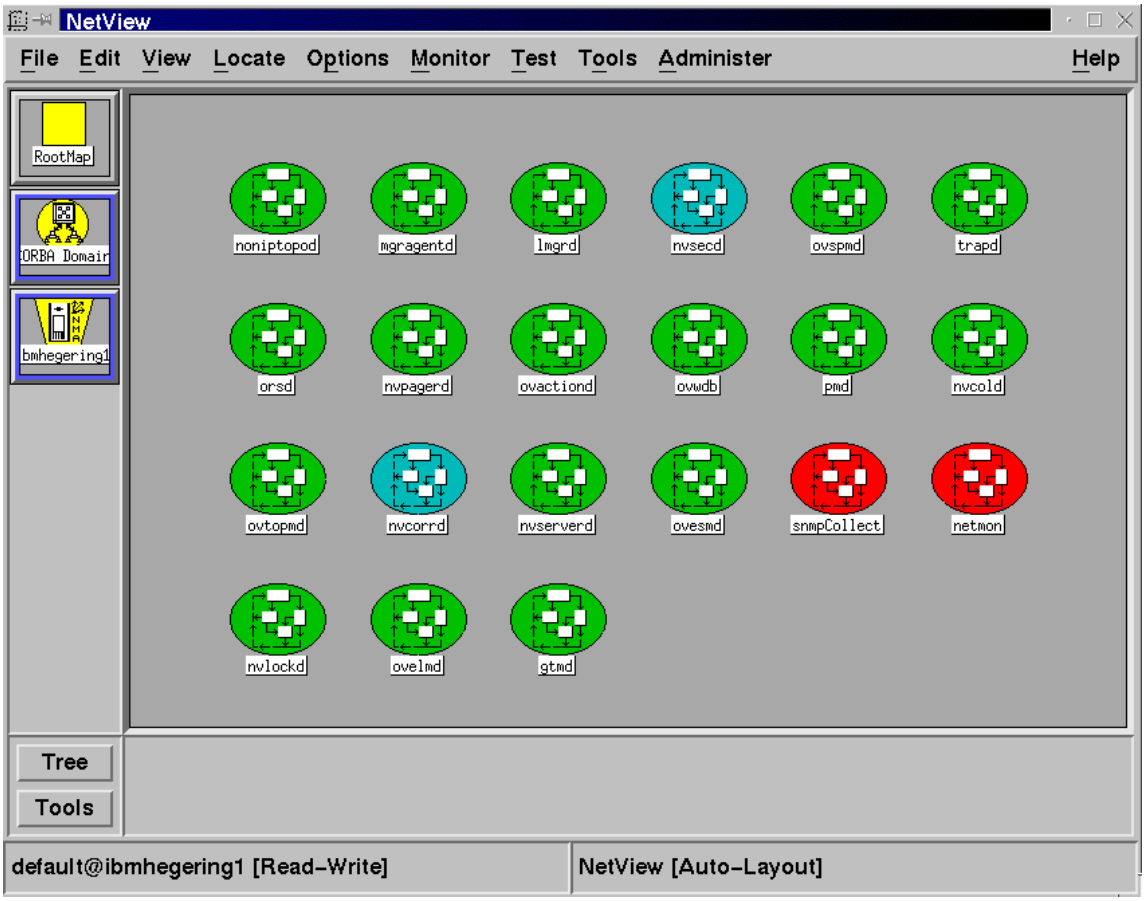


Abbildung 6.3: Zustand der NetView-Prozesse

Wie bereits oben angesprochen, besteht dieser Managementagent aus CORBA-Objekten, deren Methoden jeweils auf die in der Sprache C vorliegenden Programmierschnittstellen der Managementplattform bzw. des Betriebssystems abgebildet wurden und diese somit kapseln. Die Integration in die Testumgebung erfolgte über die IDL-Kapseln des Managementsystems, die wir im Rahmen des multiarchitekturellen Managers (siehe Abschnitt 4.2) vorgestellt haben.

6.1.3 Integriertes Management von UNIX-Systemen

In Abschnitt 5.1 haben wir anhand eines Agenten für das integrierte Management von UNIX-Systemen den von uns konzipierten Transformationsansatz zur Migration von SNMP-Agenten in eine CORBA-Umgebung vorgestellt. Der daraus hervorgegangene CORBA-konforme Systemmanagementagent wurde auf der Basis des *VisiBroker for Java*, einem Java-ORB der Firma Inprise, implementiert. Als graphische Benutzerschnittstelle kommen Java-Applets zum Einsatz, die in einem gewöhnlichen WWW-Browser ablaufen und umfassende Eingriffsmöglichkeiten in den Betrieb von UNIX-Workstations bieten.

Letztere wurden in Abschnitt 5.1.2 vorgestellt.

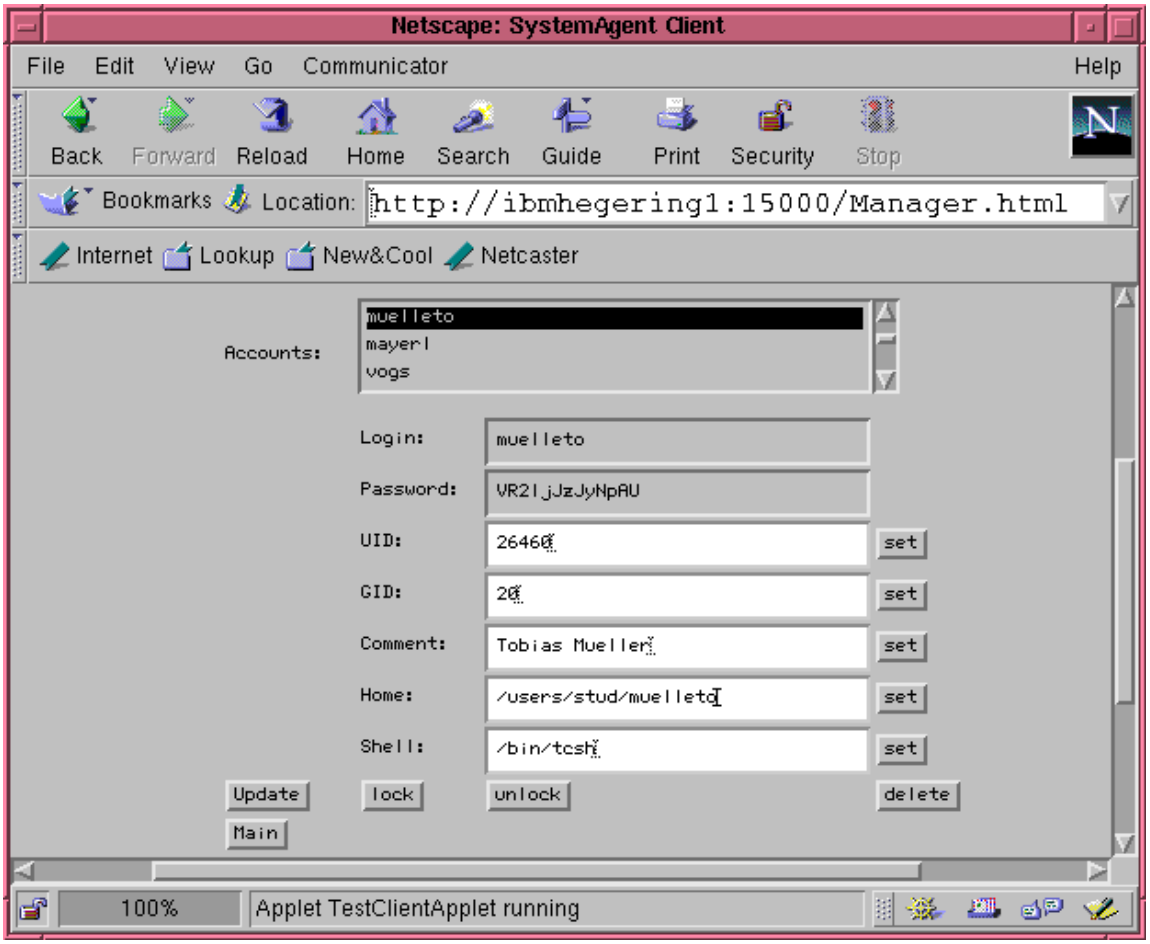


Abbildung 6.4: Web-basierte Benutzerverwaltung eines UNIX-Systems

In Abbildung 6.4 ist beispielhaft das Java-Applet zu sehen, das Informationen zur Benutzerverwaltung beinhaltet. Während grau unterlegte Felder ausschließlich lesbare Angaben beinhalten, können in den weißen Eingabefeldern Konfigurationsparameter gesetzt werden.

Der Zugriff auf die Managementinstrumentierung des Agenten erfolgt über das *Java Native Interface (JNI)*, da diese lediglich über ein C-API verfügt. Das Entwicklungsziel von JNI besteht darin, innerhalb der virtuellen Maschine ablaufenden Java-Programmen die Möglichkeit zu geben, mit Programmen zusammenzuarbeiten bzw. Bibliotheksfunktionen zu nutzen, die in einer anderen Programmiersprache (z.B. C oder Assembler) geschrieben sind. JNI besorgt die Umwandlung der Datenformate bei der Übergabe von Parametern zwischen dem Java-Programm und der Fremdfunktion (*native method*). Der Fremdprozeß hat zudem die Möglichkeit, auf Klassen und Objekte des Java-Programms zuzugreifen und kann Ausnahmen erzeugen. Der Einsatz des JNI ist erforderlich, wenn eine Java-Anwendung plattformabhängige Funktionen nutzen möchte, die nicht von den

Standardklassen des *Java Development Kit* unterstützt werden. Ein weiteres Szenario für die Nutzung des JNI ist der Fall, daß bestimmte Funktionen bereits von einer bestehenden Bibliothek bereitgestellt werden und diese von Java-Programmen genutzt werden sollen.

Für den CORBA-Systemmanagementagenten treffen beide Fälle zu: So ist es manchmal erforderlich, Betriebssystemaufrufe zur Ermittlung von Managementinformation zu nutzen. Außerdem besteht zum Objektmodell für das Management von UNIX-Workstations, welches in das generische Objektmodell integriert wurde, bereits ein SNMP-Agent (vgl. Abschnitt 5.1.2). Dieser Agent ist in C geschrieben und modular aufgebaut, d.h. Funktionen zum Lesen und Modifizieren von Attributen bzw. zum Ausführen von Aktionen werden von einzelnen Code-Modulen implementiert. Hierdurch können Funktionen, die von einem Agentenobjekt aufgerufen werden sollen, zu einer Bibliothek zusammengefaßt werden, die ihrerseits über JNI angesprochen werden kann.

Insgesamt konnte gezeigt werden, daß der in Abschnitt 5.1 vorgestellte Ansatz zur Transformation bestehender Agentenmodelle durch die Abstützung auf standardisierte Algorithmen sowie die Verwendung von CASE-Werkzeugen in wesentlichen Punkten automatisch abläuft. Die modulare Struktur des Agenten hat darüberhinaus eine schnelle und unkomplizierte Anbindung der C-basierten Agenteninstrumentierung an die vom CASE-Tool erzeugten IDL-Schnittstellen begünstigt.

6.1.4 Management verteilter Systemdienste

Die Analyse der systembezogenen Sichtweise auf Managementsysteme hat ergeben, daß diese nicht nur auf die volle Funktion des Betriebssystems angewiesen sind, sondern auch verteilte Systemdienste in Anspruch nehmen (vgl. Abbildung 5.6 in Abschnitt 5.2).

Es ist daher notwendig, neben dem im vorangehenden Abschnitt beschriebenen UNIX-Agenten ebenfalls Agenten für das Management von Diensten wie *Network Information System (NIS)* und *Network File System (NFS)* zu konzipieren. Hierbei wurde der Standpunkt verfolgt, diese beiden Systemdienste als Ausprägungsform verteilter Anwendungen zu betrachten, was zur Erweiterung der in 5.2 identifizierten GAMOCs führte. Die dort entwickelten Objektmodelle wurden hinsichtlich des Managements der Client/Server-basierten Systemdienste NFS und NIS erweitert.

Hierzu wurden die in Abschnitt 5.2 entworfenen generischen Klassen **Server** und **Client** verfeinert und zusätzliche dienstespezifische MOCs eingeführt. Obwohl die in den generischen Klassen definierte Managementinformation sinnvoll auf die betrachteten Dienste anwendbar ist, hat die Bottom-up-Analyse ergeben, daß die verbreiteten Implementierungen von NFS und NIS leider nur einen Teil davon instrumentieren, was darauf schließen läßt, da der Managementaspekt bei der Entwicklung dieser Systeme vernachlässigt wurde. Trotzdem gestattet das in [Muel 98] entworfene Modell die Überwachung des Status aller Software-Komponenten dieser verteilten Systemdienste.

Für den Dienst NFS wurden darüber hinaus durch die Verwendung von Zustandsdiagrammen des OMT-Dynamikmodells Vorgaben für asynchrone Ereignismeldungen erar-

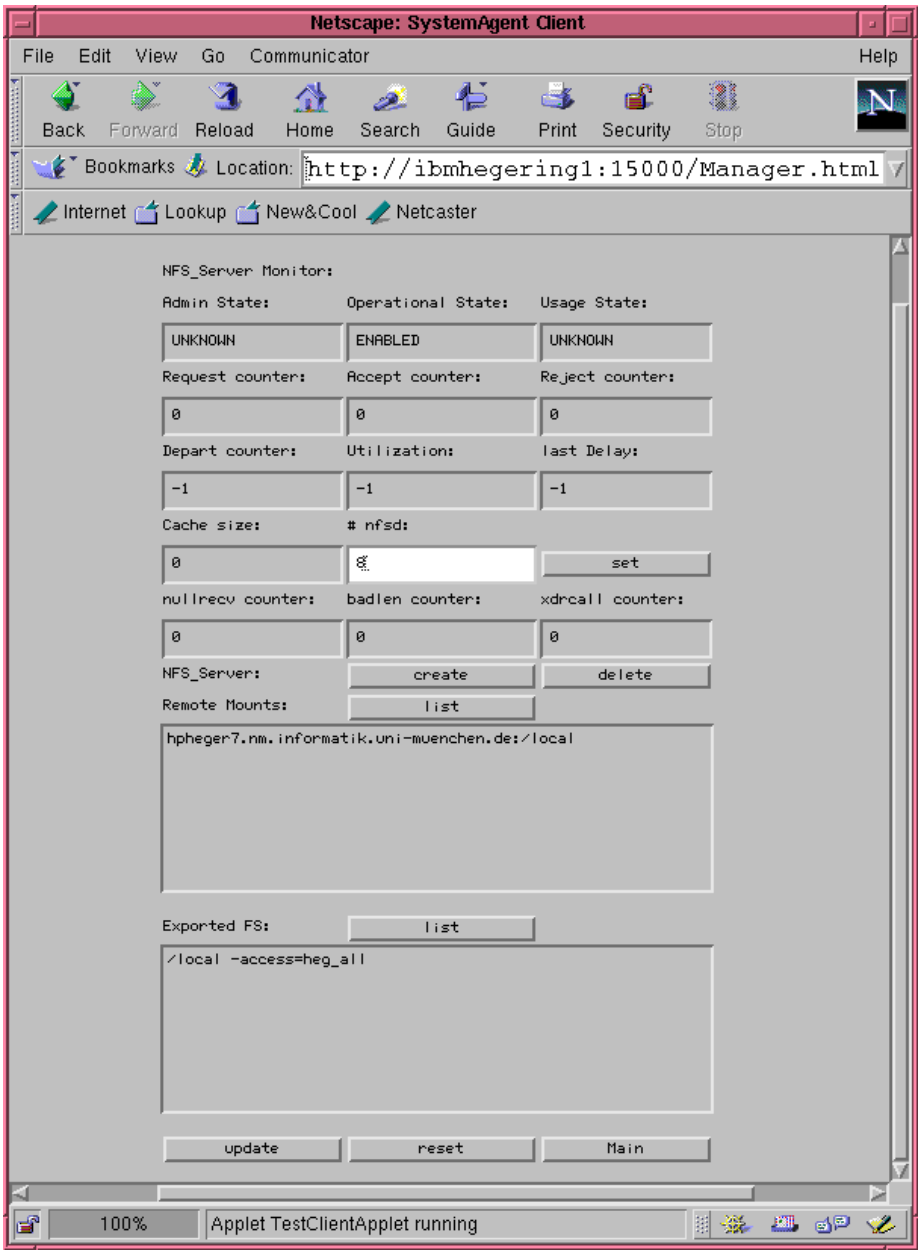


Abbildung 6.5: Applikation für das Management des NFS-Dienstes

beitet, die das Management erheblich erleichtern könnten. Damit Anwendungen und Systemdienste die geforderte Managementinformation bereitstellen, müßten sie jedoch entsprechend instrumentiert sein. Wäre dies der Fall, so könnte ein Agent auch detailliertere Informationen durch Instantiierung der GAMOCs **compInterface** und **interactionInfo** (siehe Abbildung 5.10 in Abschnitt 5.2) anbieten. Abbildung 6.5 zeigt einen Blick auf die mit CORBA und Java implementierte WWW-basierte Managementapplikation für einen NFS-Server.

Durch die Konzeption und Implementierung der Managementmodelle für NFS und NIS konnte nachgewiesen werden, daß die den GAMOCs zugrundeliegende Annahme einer für alle Arten von verteilten Anwendungen identischen Informationsmenge trotz verschiedener Ausprägungsformen haltbar ist. Das Vorhandensein eines Basisumfangs an Managementinformation nahe der Wurzel der Vererbungshierarchie trägt ferner zur Übersichtlichkeit der spezialisierten Objektmodelle bei.

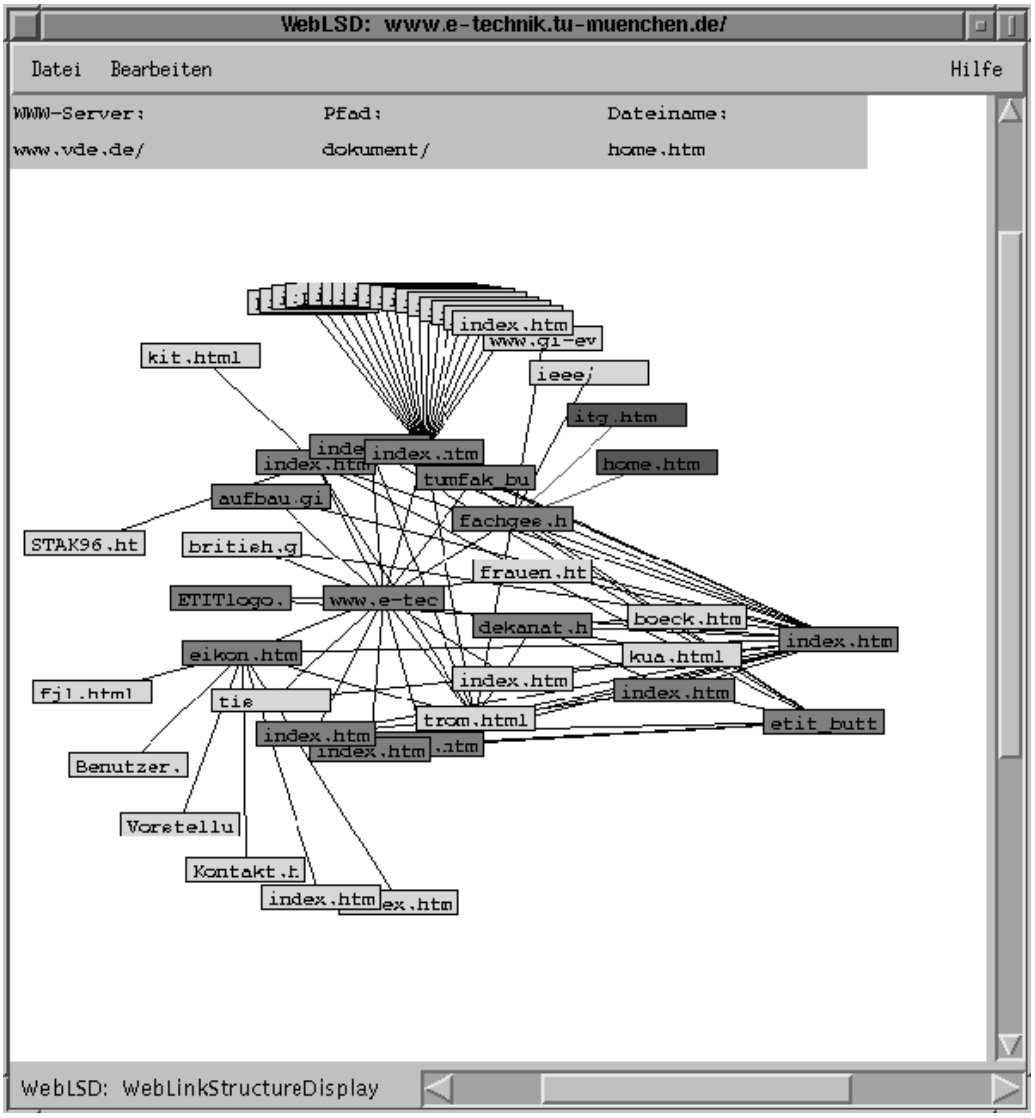


Abbildung 6.6: Applikation für das Management des WWW-Dienstes

Ein weiteres Beispiel für die praktische Anwendbarkeit dieser Methodik ist die managementtechnische Instrumentierung von WWW-Servern durch geeignete CORBA-Agenten. Auch hier wurde die Modellierung des Dienstes WWW auf Basis der GAMOCs und unter Zuhilfenahme des OMT-basierten CASE-Tools *Software through Pictures* durchgeführt.

Die Details des Objektmodells sind in [Stri 98] beschrieben. Die Implementierung beruht auf einem zuvor entwickelten [Eust 96] Java-basierten Werkzeug zur Administration von WWW-Linkstrukturen. Dieses ist in Abbildung 6.6 dargestellt.

6.1.5 Transformation eines bestehenden ATM-Agenten

Ein abschließendes Beispiel für die Anwendbarkeit des Transformationsansatzes aus Abschnitt 5.1 liefert die Umwandlung eines proprietären Managementagenten für einen ATM-basierten Telekommunikationsswitch der Firma Siemens. Das Ergebnis dieser im Rahmen einer Forschungskooperation durchgeführten Arbeit [Hoel 97] besteht in einem CORBA-konformen Managementagenten, der auf der Basis des Object Request Brokers *Orbix* von Iona innerhalb von sechs Mannmonaten implementiert wurde. Ferner wurde eine WWW-basierte Managementapplikation für das System konzipiert und mit Hilfe des Java-ORBs *OrbixWeb* implementiert.

Bei der Modellierung des Agenten haben sich die Vorteile der werkzeuggestützten Modellierung gezeigt, da die erste Fassung der Objektmodelle automatisch aus dem in C++ vorliegenden Programmcode generiert werden konnten. Außerdem war es möglich, die Modelle schnell auf andere Gerätearten anzupassen, da besonderer Wert darauf gelegt wurde, auch hier eine möglichst gut ausgestaltete Vererbungshierarchie zu entwickeln.

Die im Laufe dieses Projekts gemachten guten Erfahrungen haben bewirkt, daß der Transformationsansatz seitdem bei Siemens für die Migration sowohl proprietärer als auch SNMP-basierter Agenten in eine CORBA-Umgebung eingesetzt wird.

6.2 Delegierung von Managementfunktionalität

Bisher wurde von der Prämisse ausgegangen, daß generische Managementfunktionalität entweder innerhalb von Managementplattformen realisiert ist (vgl. die Ausführungen zum multiarchitekturellen Manager in Abschnitt 4.2.4) oder als abgesetzt implementierte Managementdienste vorliegen, wie es beispielsweise im OSI/TMN-Management mit den *Systems Management Functions* der Fall ist. Beide Ausprägungsformen besitzen die Gemeinsamkeit, daß sich die Managementdienste jeweils an *einem spezifischen* Ort befinden und somit überwiegend einem zentralistischen Organisationsschema unterliegen. Bei der Interaktion zwischen Managern und Agenten können im wesentlichen vier fundamentale Komponenten identifiziert werden, die bei der Erfüllung von Managementaufgaben zusammenwirken:

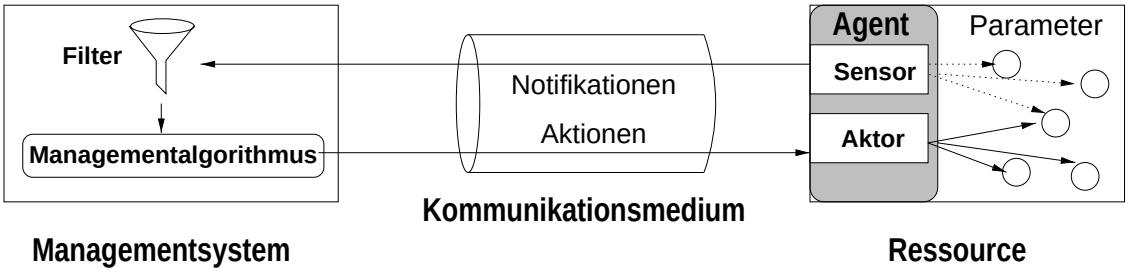


Abbildung 6.7: Management-Regelkreis in zentralistischen Systemen

- **Sensoren**:
  Sie überwachen die für das Management relevanten Parameter einer Ressource und stellen diese in einem geeigneten Format dar. Die Gesamtheit dieser Parameter ist die MIB; sie kann objektorientiert aufgebaut sein und beschreibt alle Managementschnittstellen einer Ressource. Im Falle von Änderungen kritischer Größen senden die Sensoren vordefinierte asynchrone Ereignismeldungen (Notifikationen) durch einen Kommunikationskanal an das Managementsystem.[1]

- **Filter**:
  Ihre Aufgabe besteht darin, die eingehenden Ereignismeldungen vorzuverarbeiten. Dies ist notwendig, da asynchrone Ereignismeldungen nur rohe Daten enthalten (z.B. ein Zähler hat den vorgegebenen Schwellwert überschritten), die zu Managementinformation verdichtet werden müssen. Die Korrelation von Ereignissen ist ein Beispiel für eine Aufgabe von Filtern.

- **Managementalgorithmen**:
  Die von den Filtern gewonnenen Informationen dienen als Eingabe für Managementalgorithmen, die diese auswerten, Entscheidungen treffen, und – soweit erforderlich – das Auslösen von Aktionen auf den Ressourcen vornehmen. Diese Rolle wird zum gegenwärtigen Zeitpunkt vom Bediener des Managementsystems wahrgenommen; in jüngster Zeit wird versucht, diese Aufgabe unter Zuhilfenahme von Expertensystemen zumindest teilweise zu automatisieren.

- **Aktoren**:
  Die Durchsetzung der vom Managementalgorithmus angeordneten Aktionen auf der Ressource ist die Aufgabe der Aktoren. Dies geschieht durch das Setzen von MIB-Variablen, die intern auf Ressourcen-Operationen abgebildet werden.

[1] Dies ist insbesondere beim OSI-Management der Fall, das mit der *Event Report Management Function* sehr gute Möglichkeiten bietet, ereignisgesteuertes Management zu realisieren. Für das Internet-Management gilt jedoch, daß jeweils das Managementsystem dafür verantwortlich ist, über den aktuellen Status aller Ressourcen informiert zu sein. Dies geschieht durch zyklisches Abfragen von Ressourcenparametern (*Polling*).

Diese vier Komponenten bilden den in Abbildung 6.7 skizzierten Management-Regelkreis. In zentralistischen Systemen sind sie folgendermaßen verteilt: Sensoren und Aktoren befinden sich auf der administrierten Ressource und bilden zusammen den Agenten: Sie bilden durch die Abbildung ressourcenspezifischer Parameter in eine für das Management geeignete Form die Grundlage, um die Ressource auf der Grundlage standardisierter Managementarchitekturen zu administrieren.

Das Managementsystem selbst beinhaltet (wie in Abschnitt 4.2.2 beschrieben) neben Komponenten wie graphischen Benutzeroberflächen oder Entwicklungswerkzeugen eine Vielzahl von Filtern und Managementalgorithmen. Um Ereignissen und Aktionen spezifischen Ressourcen zuordnen zu können, muß das Managementsystem die Objektbeschreibungen sämtlicher Ressourcen in Datenbanken speichern, die sich in seinem Zuständigkeitsbereich befinden.

Während das Sammeln managementrelevanter Parameter und das Ausführen von Aktionen verteilt sind, bleibt die Behandlung dieser Daten dem (häufig zentralen) Managementsystem vorbehalten. Die Folge davon ist, daß gegenwärtige Managementsysteme wie *TME 10 NetView* oder *HP OpenView* sehr komplex sind und hohe Ressourcenanforderungen haben. Obwohl diese Systeme zwar prinzipiell die Bildung von Manager-Hierarchien zulassen, handelt es sich dabei letztlich um jeweils eigenständige, vollwertige Managementsysteme.

Wie in Abschnitt 3.2.1 ausgeführt wurde, liegt der wesentliche Nachteil eines zentralistischen Ansatzes in seiner mangelhaften Skalierbarkeit, da die Funktionsweise eines derart aufgebauten Managementsystems unmittelbar von folgenden Faktoren abhängig ist (siehe dazu auch [Gold 96]):

- Die Anzahl der administrierten Ressourcen,
- die Menge an Managementparametern der jeweiligen Ressourcen sowie
- die Bandbreite des Kommunikationsmediums zwischen Managementsystem und Ressourcen.

Wenn die Ressourcenanzahl sowie deren Menge an Managementparametern stark ansteigen oder die Bandbreite des Kommunikationsmediums gering ist, besteht die Gefahr – wie Erfahrungen großer Netzbetreiber zeigen – daß sowohl Managementsysteme als auch das Kommunikationsmedium sehr leicht zu Flaschenhälsen werden.

Angesichts der vier Komponenten des Management-Regelkreises kann einer solchen Gefahr am besten begegnet werden, indem Teile der Filterfunktionalität und der Managementalgorithmen auf die Agentensysteme *zur Laufzeit der Systeme* verlagert werden bzw. diese Funktionalität explizit an die Ressourcen delegiert wird. Unter Rückgriff auf das in Abschnitt 3.2.1 beschriebene *Management by Delegation*-Paradigma wird nachfolgend ein Ansatz beschrieben, der dies in einer CORBA-Umgebung leistet.

6.2.1 CORBA-basiertes Management by Delegation

Aus den bisherigen Kapiteln konnte die Erkenntnis gewonnen werden, daß CORBA nahezu alle Transparenzkriterien aus Abschnitt 2.3.1 erfüllt. Was fehlt, ist die Portabilität verteilter Anwendungen, d.h. deren Ausführbarkeit auf verschiedenartigen Systemen ohne Vornahme von Modifikationen. Dies ist jedoch ein programmiertechnisches Problem und liegt somit nicht im Einflußbereich von CORBA. Zur Erreichung des Kriteriums „Portabilität“ ist also eine Programmiersprache gefordert, für die Interpreter auf unterschiedlichen Systemarchitekturen existieren. Die Programmiersprache Java ist hierfür besonders gut geeignet, da einerseits Interpreter zur Ausführung des aus Java erzeugten Bytecodes für eine Vielzahl unterschiedlicher Hardware- und Betriebssystemplattformen erhältlich sind. Andererseits existiert für die Kopplung von Java und CORBA das von der OMG standardisierte Java/IDL Language Mapping. Ferner existieren WWW-Browser, die IIOP-Nachrichten versenden und empfangen können und als **CORBA-enabled WWW-Browser** bezeichnet werden. Diese wurden bei einigen der in Abschnitt 6.1 beschriebenen Prototypen eingesetzt, um die von CORBA-Agenten angebotene Managementinformation graphisch darzustellen.

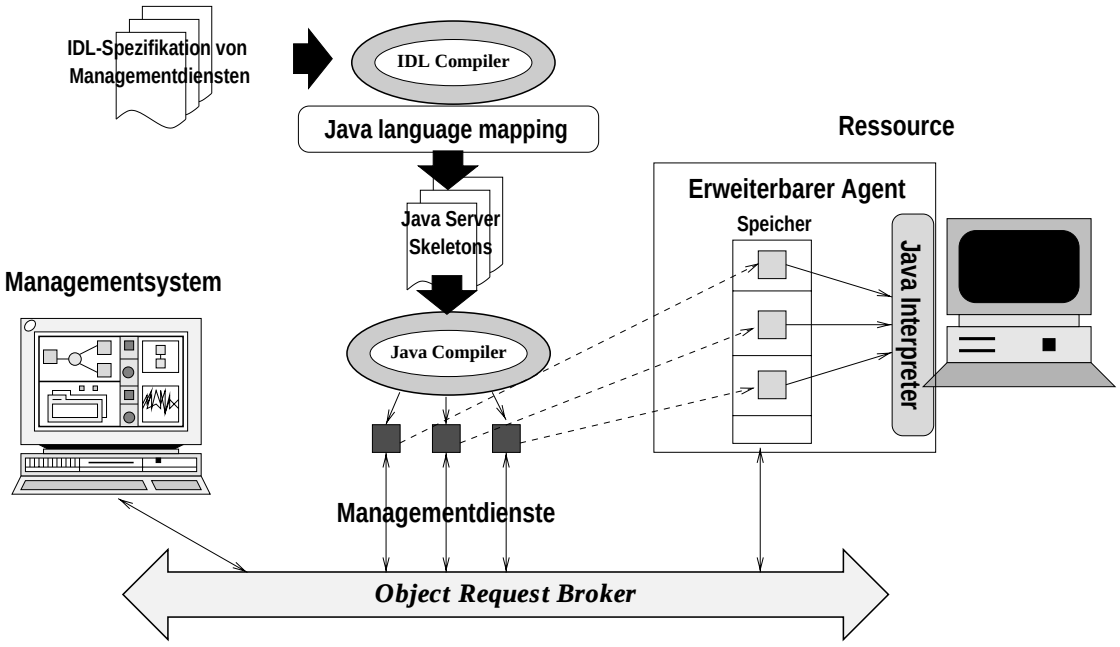


Abbildung 6.8: CORBA-basiertes *Management by Delegation*

Besitzen in Java implementierte Managementdienste IDL-Schnittstellen, so können diese nach dem Prinzip des *Management by Delegation* via CORBA von Managementsystemen an Agenten delegiert werden (vgl. Abschnitt 3.2.1). Die technischen Aspekte der Implementierung delegierbarer CORBA-Agenten sind in Abbildung 6.8 dargestellt.

Aus den IDL-Beschreibungen der Managementfunktionalität entstehen bei der Implementierung CORBA-Serverobjekte, die den Kern der Dienstimplementierung bilden. Der

IDL Compiler generiert aus den IDL-Objektbeschreibungen anhand der von der OMG standardisierten Abbildungsvorschriften *Client Stubs* und *Server Skeletons* in Form von Java-Dateien. Die Skeletons ermöglichen die Kommunikation der Serverobjekte mit dem ORB. Hierzu werden sie zusammen mit dem Code, der die Managementinformation und Funktionalität eines Objekts realisiert, mit dem Java Compiler übersetzt. Es entsteht ein Objekt, das den Dienst implementiert und beim *Implementation Repository* des Object Request Brokers registriert werden kann. Die Stubs ermöglichen es einem Client, über den ORB auf ein Dienstobjekt zuzugreifen. Der Dienst kann somit von beliebigen CORBA-Objekten in Anspruch genommen werden. Der Transfer des Dienstes kann dabei auf zwei unterschiedliche Arten geschehen:

- Mit der *Mobile Agent System Interoperability Facilities Specification (MASIF)* [OMG MAF] der OMG steht ein Rahmenwerk zur Verfügung, das unter anderem Festlegungen zum Transport von mobilen CORBA-basierten Objekten sowie zu deren Lokalisierung macht. Ein am Lehrstuhl entwickelter Prototyp [Kemp 98] hat die praktische Einsetzbarkeit dieses Konzepts für Managementzwecke gezeigt.
- Das *Java Dynamic Management Kit (JDMK)* [Sun 98] ist ein von der Firma Sun Microsystems entwickeltes System zur Implementierung mobiler Java-Komponenten, das speziell auf Managementzwecke zugeschnitten ist und neben anderen Kommunikationsinfrastrukturen auch CORBA zur Delegierung unterstützt. Ein Prototyp zur Evaluierung von JDMK für Java/CORBA-basiertes Management wird gegenwärtig implementiert [Knoe 99], wobei jedoch das frühe Stadium des Projekts noch keine abschließenden Aussagen zuläßt.

Insgesamt ist festzustellen, daß insbesondere in der jüngsten Vergangenheit sowohl von der OMG als auch von Seiten der Hersteller einige Initiativen unternommen wurden, um CORBA-basierte Managementdienste zur Laufzeit an beliebige Systeme zu delegieren. Das damit verfolgte Ziel besteht in der Verbesserung der Skalierbarkeit CORBA-basierten Managements, die gegenwärtig verbesserungsbedürftig ist. Diese Erkenntnis konnten wir nicht zuletzt durch die im vorigen Abschnitt vorgestellten prototypischen Implementierungen gewinnen.

Insbesondere die in jüngster Zeit festzustellenden Fortschritte auf dem Gebiet der *mobilen Agenten* [PhKa 98] geben Anlaß zur Hoffnung, daß CORBA-basiertes Management in absehbarer Zeit auch unter Leistungsgesichtspunkten Vorteile gegenüber anderen Managementarchitekturen für sich verbuchen kann. Erste Anwendungen im Bereich der Telekommunikation [GeDi 98] sind vielversprechend, auch wenn die Sicherheitsaspekte gegenwärtig noch weitgehend ungelöst sind [GrBy 98].

6.3 Zusammenfassung: Der Weg zu verteiltem CORBA-basierten Management

Unsere in Abschnitt 6.1 beschriebenen prototypischen Implementierungen CORBA-basierter Managementagenten für diverse Anwendungsszenarien haben insbesondere zu folgenden Erkenntnissen geführt:

- Das Konzept der auf dem ODP-Referenzmodell sowie dem Ansatz von Neumair aufbauenden GAMOCs, deren Ziel darin besteht, für eine Vielzahl verteilter Anwendungen gültige Managementinformation bereits an der Wurzel der Vererbungshierarchie zu definieren, hat sich als praktikabel erwiesen: Anhand unserer Prototypen konnten wir aufzeigen, daß die in den GAMOCs enthaltene Managementinformation nicht nur für verteilte kooperative Managementsysteme gültig ist, sondern einen Grundumfang an Managementinstrumentierung auch für verteilte Systemdienste (z.B. NFS und NIS; vgl. Abschnitt 6.1.4) und Anwendungen (WWW) bereitstellen.
- Die Eignung des in Abschnitt 5.1 vorgestellten werkzeugunterstützten Transformationsansatzes zur Gewinnung von CORBA-Agenten aus bestehenden Implementierungen konnte nicht nur anhand von SNMP-Agenten nachgewiesen werden, sondern gestattet ebenfalls das Re-Engineering bisher proprietärer Technologie (vgl. Abschnitt 6.1.5).
- Die Kombination WWW-basierter Technologien mit CORBA gestattet eine einfache Entwicklung von Managementapplikationen nach dem *Rapid-Prototyping*-Ansatz. Die heutigen Entwicklungswerkzeuge bieten durch die Bereitstellung eines IDL/Java Language Mappings hierfür eine brauchbare Unterstützung, bei der nur in Ausnahmefällen manuelle Anpassungen des generierten Codes erforderlich sind.
- Die Tatsache, daß es zum heutigen Zeitpunkt aus Sicherheitsgründen nicht möglich ist, unmittelbar aus Java-Programmen auf Systemressourcen zuzugreifen, bedeutet eine signifikante Einschränkung für das Management: Grundsätzlich läuft der interpretierte Java-Bytecode in einer von den Systemressourcen abgeschotteten Umgebung ab (sog. „Sandbox“), was offensichtlich dem Ziel des Managements entgegensteht. Abhilfe soll das in der zukünftigen Java-Version 1.2 realisierte Konzept der „signed Applets“ schaffen, mit dem ein Mechanismus zur Autorisierung von Applets bereitgestellt wird.
- Die (dynamische) Delegierung allgemein verwendbarer Managementdienste ist ein wichtiges Mittel zur Verbesserung der Skalierbarkeit des Managements. Die Kombination von CORBA mit Java erfüllt die Kriterien der Verteilungstransparenz.
- Die gegenwärtig verfügbare Menge an generischen CORBA-Managementdiensten reicht trotz der von der OMG sowie der OpenGroup unternommenen Standardisierungsbemühungen noch nicht aus, um ein umfassendes, ausschließlich CORBA-

basiertes Management zu etablieren. Es ist daher notwendig, in anderen Managementarchitekturen bzw. in Plattformimplementierungen vorhandene Managementdienste für CORBA nutzbar zu machen. Diese können als temporärer Ersatz für momentan in der Spezifikationsphase befindliche CORBA-Managementdienste genutzt werden, wie das in Abschnitt 6.1.2 vorgestellte Implementierungsbeispiel für das Management verteilter kooperativer Managementsysteme verdeutlicht hat.

Hierzu werden wir zunächst aufzeigen, welche Dienste sich besonders gut zur Delegierung an Agentensysteme eignen und anschließend ein Konzept vorstellen, wie die Vielzahl der in heutigen Managementsystemen enthaltenen Dienste für CORBA-basierte Systeme verfügbar gemacht werden können.

Wie in den Abschnitten 2.3.4 und 4.4.5 ausgeführt wurde, hat die Definition von gemeinsam verwendbarer und delegierbarer Managementfunktionalität, zwei Hauptgründe:

- Funktionalität, die von mehreren Managementanwendungen benötigt wird, sollte nur einmal definiert und implementiert und von den Anwendungen dann gemeinsam benutzt werden.
- Skalierbarkeit für eine große Zahl von Endsystemen setzt voraus, daß die Ausführung bestimmter Funktionen an die administrierten Systeme delegiert werden kann. In heterogenen Umgebungen müssen diese Funktionen dann natürlich exakt definiert und offengelegt sein.

Gemeinsam nutzbare Funktionalität, die für CORBA-basiertes Management derzeit definiert wird (vgl. dazu die Ausführungen in Abschnitt 3.1.2), ist zu einem großen Anteil bereits in heutigen Managementplattformen enthalten. Diese in Abbildung 6.9 schematisch dargestellten (grau unterlegten) Basisdienste erstrecken sich vor allem auf folgende Bereiche:

- Kreieren und Löschen von Managementobjekten, die die zu administrierenden Ressourcen für das Management repräsentieren (*Lifecycle Management*),
- Gruppieren von Objekten zu Managementdomänen (*Domains, Collections*) nach wählbaren Kriterien und Administration dieser Domänen,
- Definition und Anwendung bestimmter Zielvorgaben (*Policies*) für das Management der Objekte innerhalb einer Domäne, Erkennung eventueller konfliktärer Vorgaben,
- Handhabung verschiedener Versionen von Objektdefinitionen und Implementierungen, um Interoperabilität von Managementanwendungen und administrierten Ressourcen sicherzustellen,
- Definition umfassender Filtermöglichkeiten für asynchrone Ereignismeldungen,
- Bereitstellung einer Möglichkeit zur persistenten Speicherung von Ereignismeldungen in Logs.

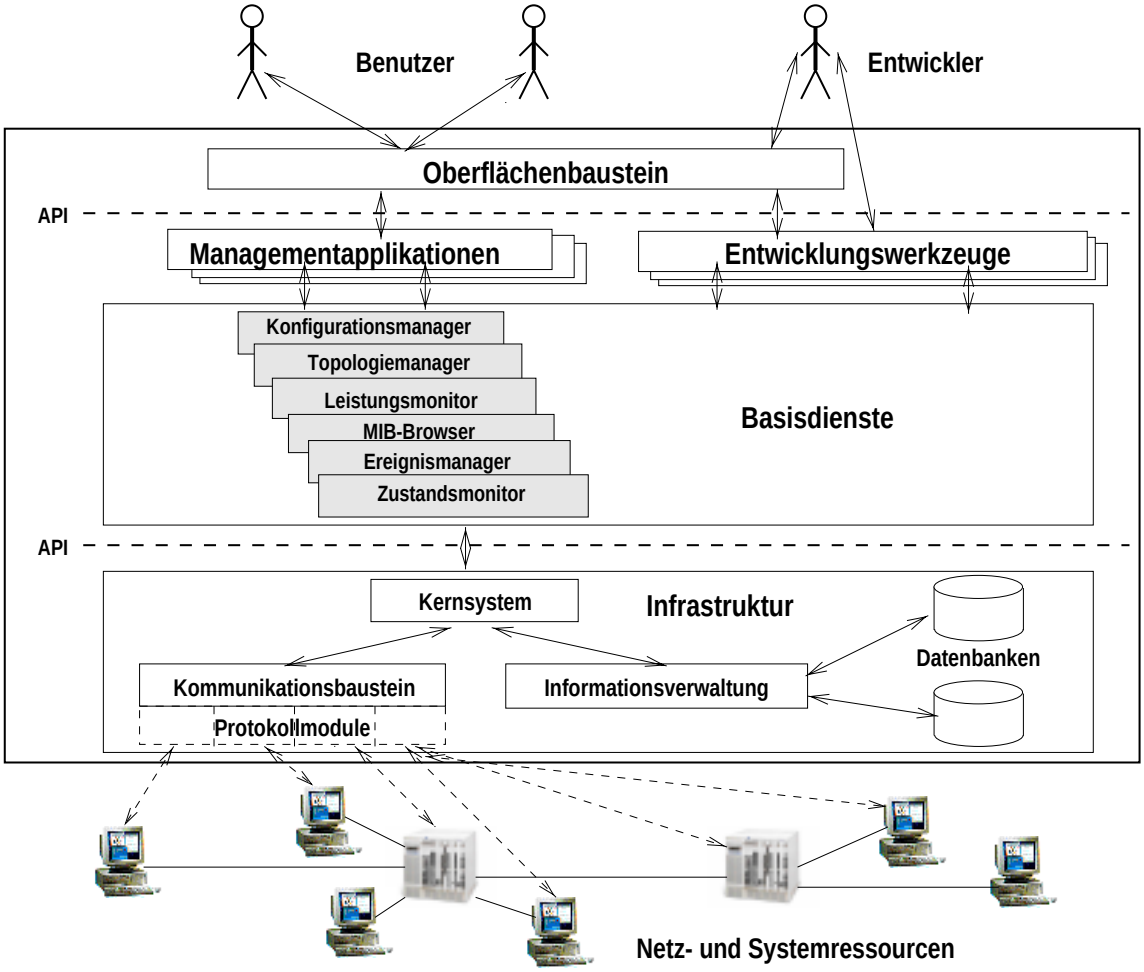


Abbildung 6.9: Aufbau einer Managementplattform

Delegierbare Funktionalität, die die Skalierbarkeit für umfangreiche DV-Infrastrukturen sicherstellt, wird in einem ersten Schritt vor allem für folgende Bereiche benötigt:

- Handhabung asynchroner Ereignismeldungen und Alarme (*Event Management:* Abonnieren wählbarer Ereignistypen durch Anwendungen, Filterung von Ereignismeldungen nach wählbaren Kriterien),
- Handhabung periodisch durchzuführender Managementaktionen (*Scheduling*),
- Überwachung von Schwellwerten (*Threshold Monitoring*),
- Aufzeichnen bestimmter, vor allem auch sicherheitsrelevanter Ereignisse, Komprimierung der zugehörigen Daten (*Logging, Log Management*)

Diese Liste könnte natürlich noch erheblich erweitert werden. Es handelt sich hier nur um die dringlichsten Aufgaben, die vorrangig behandelt werden müssen.

Kapselung der Basisdienste

Die ersten Schritte bestehen nun darin, die in den Managementsystemen vorhandenen Basisdienste sukzessive in CORBA zu überführen, während die Plattform-Infrastruktur in ihrem ursprünglichen Zustand bleibt. Die Verwendung von IDL-Wrappern ist, wie wir in Abschnitt 4.2.4 anhand der Kapselung der Plattformmodule zur Ereignis- bzw. Topologieverwaltung aufgezeigt haben, ein einfaches und zugleich effektives Mittel zur schnellen Migration von „legacy"-Code in objektorientierte Umgebungen. Wendet man dies auf alle in einer Plattform vorhandenen Dienstkomponenten an, so erhält man eine CORBA-basierte – unter Umständen auf mehrere Systeme verteilbare – Managementplattform, wie sie in Abbildung 6.10 dargestellt ist.

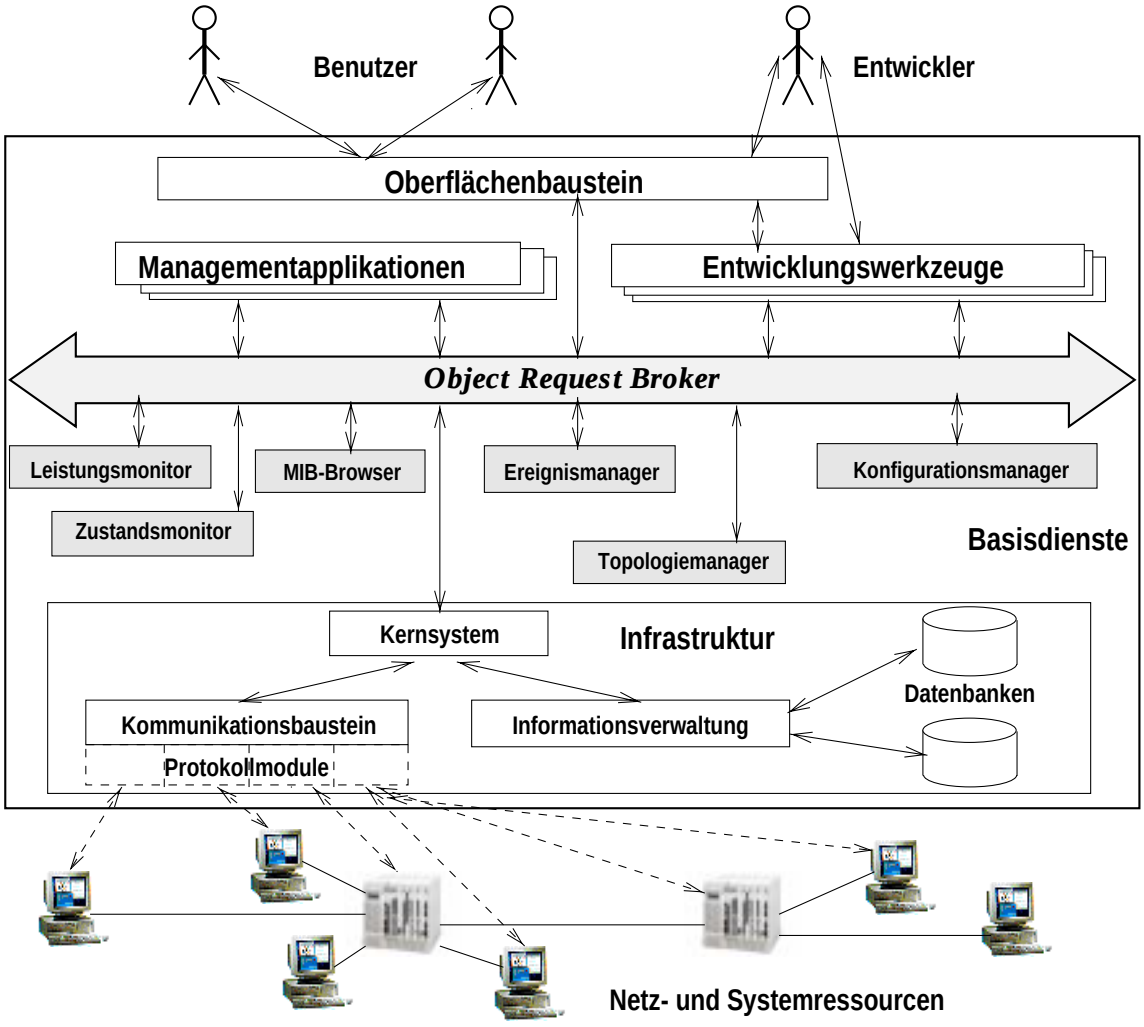


Abbildung 6.10: CORBA-basierte verteilte Managementplattform

Hierbei kommunizieren sämtliche Basisdienste über einen Object Request Broker miteinander, wobei dieser jedoch nicht von Außen zugreifbar ist, d.h. von Agenten bzw. ande-

ren Managementsystemen. Die Interaktionen mit anderen Systemen geschehen ausschließlich über eine klar definierte Schnittstelle – den Kommunikationsbaustein der Plattform. Dies trägt der Tatsache Rechnung, daß trotz der Standardisierung des *CORBA Security Service* heutige kommerzielle ORB-Implementierungen nur in seltenen Fällen über ein tragfähiges Sicherheitskonzept verfügen. Ebenso erweist sich die Migration der Datenbestände in eine CORBA-Umgebung oft als schwierig, da hierfür häufig Produkte von Drittherstellern (wie z.B. relationale Datenbanksysteme) eingesetzt werden, deren Migrationsstrategie sich oft nicht mit der des Plattformherstellers deckt. Ferner finden sich bisher kaum Implementierungen des *Licensing Service*, der objektbezogene, und damit feingranulare Abrechnungsmechanismen bereitstellt. Dies stellt jedoch in Umgebungen mit unterschiedlichen Dienstanbietern bzw. -nutzern eine kritische Einschränkung dar.

Offenlegung der Basisdienste und der Infrastruktur

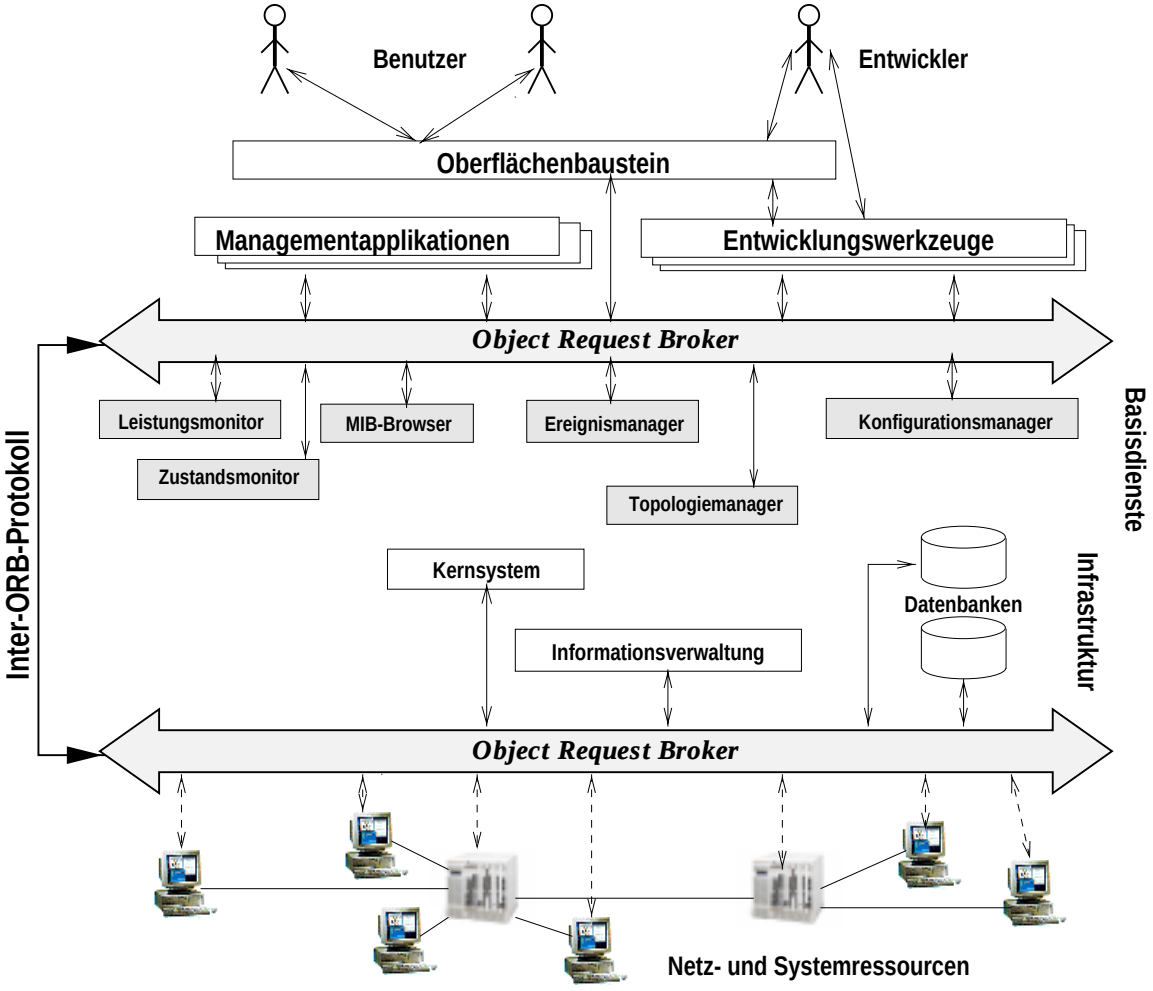


Abbildung 6.11: Verteiltes CORBA-basiertes Management

Sind diese Voraussetzungen jedoch erfüllt, besteht der zweite Schritt nun darin, die IDL-Schnittstellen sowohl der Basisdienste als auch der Plattform-Infrastruktur vollständig offenzulegen: Die Basisdienste sind dann für alle (Agenten-) Systeme verfügbar, die ihrerseits mit einem zum CORBA 2.0-Standard konformen ORB ausgestattet sind. Hierdurch wird eine vollständige Verteilung des Managementsystems erreicht, wie in Abbildung 6.11 dargestellt. Eine Konsequenz hieraus ist, daß die Managementdienste durch entsprechende Replikationsmechanismen in einem verteilten System dort plaziert werden können, wo sie am häufigsten benötigt werden. Alternativ dazu können die in Abschnitt 6.2 besprochenen Mechanismen zur dynamischen Delegierung der Managementdienste an Agentensysteme dazu verwendet werden, um eine möglichst dezentrale Verarbeitung der Managementinformation zu erreichen und so die Skalierbarkeit des Gesamtsystems zu erhöhen.

Ein letzter Schritt in Richtung eines verteilten CORBA-basierten Managements besteht nun darin, eine Umkehr des bislang verwendeten Push-Modells für die Delegierung vorzunehmen. Bisher liegt die Entscheidung, Managementfunktionalität an Agentensysteme zu Delegieren, ausschließlich beim Manager, d.h. dieser delegiert aufgrund eines lokal ablaufenden Entscheidungsprozesses explizit Dienste an Agentensysteme, ohne daß diese über Möglichkeiten zur Einflußnahme verfügen.

Verwendet man jedoch das Pull-Modell, werden die Agentensysteme dahingehend erweitert, um sich die von ihnen benötigten Dienste selbständig aus einem „Dienstepool“ zu beziehen. Man erhält so die in Abschnitt 2.3.2 angesprochenen „Selbststeuernden Systeme“.

KAPITEL 7

Zusammenfassung der Ergebnisse und Ausblick

Die hohe Praxisrelevanz und Aktualität des Enterprise Managements ergibt sich aus der Notwendigkeit, die Administration der Netze für Sprach- und Datenkommunikation auf die Ziele des Betreibers hin auszurichten. Dies impliziert nicht zuletzt einen Wechsel von der bisherigen „Bottom-up"-Sichtweise hin zu einer von den Zielvorgaben ausgehenden, ganzheitlichen „Top-down"-Betrachtung der Managementproblematik.

7.1 Ergebnisse dieser Arbeit

Das gegenwärtige frühe Stadium des unternehmensweiten integrierten Managements führt dazu, daß Begriffe wie „Enterprise Management", „Managementsystem", „Interoperabilität" häufig inkonsistent, mehrdeutig und oft sogar widersprüchlich verwendet werden. Wir haben daher zunächst die für die vorliegende Arbeit relevanten Begriffe eingeführt und so den begrifflichen Kontext für die weiteren Kapitel festgelegt.

Um die Tragweite des im Rahmen dieser Arbeit behandelten Problembereichs abschätzen zu können, haben wir anhand mehrerer Szenarien sowohl aus dem Daten- als auch aus dem Telekommunikationsumfeld systematisch die Anforderungen an das Management verteilter kooperativer Managementsysteme abgeleitet. Hierbei haben wir unter anderem die Erkenntnis gewonnen, daß Ablaufumgebungen für verteilte Anwendungen („Middleware") und Managementarchitekturen logisch geschichtet sind und Anforderungen wie Offenheit, Flexibilität und Verteilungstransparenz folglich nicht nur notwendige Charakteristiken einer Managementarchitektur sind, sondern insbesondere Eigenschaften der Middleware sind, mit der diese letztlich implementiert werden. Eine wichtige Implikation aus der logischen Schichtung der Rahmenwerke für verteilte Anwendungen sowie für das Management ist, daß eine Managementarchitektur eine Vielzahl wichtiger Dienste bereits von der Middleware übernehmen kann und nicht mehr neu definieren muß: Dienste zur Zustellung asynchroner Ereignismeldungen sind das offensichtlichste Beispiel für die sich hierbei ergebenden Möglichkeiten der Wiederverwendung. Folglich reicht es nicht

aus, unsere Betrachtungen auf reine Managementarchitekturen zu beschränken; vielmehr ist es notwendig, auch Rahmenwerke für verteilte Umgebungen anhand der aus den Managementszenarien gewonnenen Anforderungen hinsichtlich ihrer Eignung für das Enterprise Management zu evaluieren.

Bei der Analyse des Status Quo im Bereich der Architekturen für das Management sowie für verteilte Verarbeitung hat sich gezeigt, daß mit CORBA und dem RM-ODP leistungsfähige Rahmenwerke vorliegen, auf denen wir unseren Lösungsansatz für das Enterprise Management aufbauen können, auch wenn wichtige Dienste derzeit noch nicht spezifiziert sind bzw. noch keine vollständigen Implementierungen vorliegen:

- Die Vorteile des Einsatzes von CORBA für das Management liegen insbesondere darin, daß *eine einzige* Architektur sowohl für die Entwicklung als auch für das Management eines verteilten Systems verwendet werden kann. Dies bedeutet, daß nicht nur die Beschreibung sowohl der Nutz– als auch der Managementobjekte des Systems in einer identischen Notation erfolgt, sondern ebenfalls Nutz– und Managementdaten einer Applikation mit demselben Kommunikationsmittel (nämlich dem Object Request Broker) übertragen werden. Demzufolge kann das breite Spektrum verfügbarer Werkzeuge zur Softwareentwicklung genutzt werden, da sowohl Nutz- als auch Managementdaten identisch modelliert und implementiert werden.
- Das ODP-Referenzmodell bildet mit seinen Viewpoint Languages ein solides und umfangreiches Beschreibungsmodell für generische Klassen verteilter Anwendungen, die wir als Basis für unsere Objektmodelle verteilter kooperativer Managementsysteme einsetzen können.

Die Analyse bestehender Forschungsansätze führte zu dem Ergebnis, daß zwar in verwandten Problembereichen dieser Arbeit bereits erste Resultate vorliegen, die uns Hilfestellungen bei unserem Lösungskonzept geben können. Die Kernfragestellung der vorliegenden Arbeit ist bislang jedoch weder angegangen noch gelöst worden. Auch kommerzielle Produkte, die einen inhaltlich mit dieser Arbeit bestenfalls verwandten Themenkomplex behandeln, sind von ihrer technischen Ausgestaltung entweder auf einen zu kleinen Bereich ausgelegt oder stützen sich auf Basisarchitekturen ab, deren Unzulänglichkeiten die Effektivität der Lösungen beeinträchtigen. Insbesondere mußte festgestellt werden, daß bislang sämtliche kommerziellen Produkte von der Prämisse einer homogenen Managementinfrastruktur ausgehen.

Diese Annahme ist jdeoch in heutigen Kommunikationsnetzen nicht mehr gültig: Gerade das Zusammenwachsen der Daten- und Telekommunikation bringt das Problem der **Heterogenität des Managements** mit sich, da traditionell für das Management dieser bisher getrennten Netze unterschiedliche Managementarchitekturen verwendet wurden. Zusätzlich wurden in jüngster Zeit neben proprietären Ansätzen mehrere Managementarchitekturen standardisiert, die überwiegend in Konkurrenz zueinander stehen. Mit dem Vorhandensein einer Vielzahl heterogener Managementarchitekturen ist Enterprise Management nur dann machbar und aussichtsreich, wenn andere Managementarchitekturen in

die Enterprise Management Architektur möglichst nahtlos integriert werden können. Gefordert ist also ein **Umbrella Management**, welches Übergänge zwischen der Enterprise Management Architektur und anderen Architekturen schafft und somit eine *einheitliche* Sichtweise auf alle Netzkomponenten, Systeme und Anwendungen eines Kommunikationssystems *unabhängig* von ihrer ursprünglichen Architektur gewährleistet.

Nicht zuletzt aus diesem Grund haben wir die Mechanismen zur Abstraktion von den Spezifika der einzelnen Managementarchitekturen behandelt und die hierfür denkbaren Alternativen (multiarchitektureller Manager, multiarchitektureller Agent, Management-Gateway) anhand ihrer technischen Ausprägungen sowie in bezug auf ihre praktische Einsetzbarkeit evaluiert. Aufgrund unserer Implementierungserfahrungen waren wir in der Lage, eine umfassende Bewertung der Alternativen vorzunehmen. Dies ist umso wichtiger, als bisher keine Betriebserfahrungen mit Implementierungen zur Sicherstellung der Interoperabilität vorliegen.

Der vielversprechendste (aber auch sehr aufwendige) Ansatz für das Umbrella Management besteht zweifellos in der Bereitstellung von Management-Gateways, da sich die hierfür notwendigen Transformationsmechanismen gegenwärtig in der Standardisierungsphase befinden und unsere Implementierungen somit das zentrale Kriterium der Offenheit erfüllen. Der wohl wichtigste Vorteil von Management-Gateways besteht darin, weder Modifikationen an Managern noch an Agenten vornehmen zu müssen, was sich in der Unabhängigkeit von konkreten Produkten und somit in einem breiten Anwendungsspektrum äußert.

Außerdem haben wir festgestellt, daß Management-Gateways aufgrund ihrer Agent/Manager-Doppelrolle als ein Spezialfall verteilter kooperativer Managementsysteme angesehen werden können: Aus unseren Erfahrungen bei der Implementierung der CMIP/SNMP sowie der CORBA/SNMP Management-Gateways ergibt sich, daß insbesondere auch Management-Gateways überwacht und gesteuert werden müssen.

Zusammenfassend läßt sich sagen, daß mit dem Gateway-basierten Umbrella Management die technische Grundlage zur Etablierung eines *vollständig* CORBA-basierten Enterprise Managements gelegt wird, da so die „klassischen“ Managementarchitekturen fast nahtlos in CORBA integriert werden können. Demzufolge können wir bei der Konzeption der von uns entworfenen Objektmodelle verteilter kooperativer Managementsysteme von einer *homogenen*, CORBA-basierten Umgebung ausgehen, ohne dadurch die Anwendbarkeit unserer Methodik in *heterogener* Umgebung zu gefährden.

Beim Entwurf dieser Objektmodelle sind folgende beiden Fälle zu unterscheiden:

- Falls bestehende Managementinformation und -dienste aus anderen Architekturen übernommen werden sollen, muß dafür gesorgt werden, daß ein möglichst großer Teil an Managementinstrumentierung wiederverwendet werden kann. Hierfür haben wir einen werkzeugunterstützten Ansatz zur Gewinnung neuer CORBA-konformer Agenten aus bestehenden SNMP-Implementierungen entwickelt, der gleichzeitig die mit der Verwendung von CORBA einhergehenden Vorteile der Nutzung moderner Softwareentwicklungssysteme wie z.B. CASE-Tools illustriert.

- Falls bisher in keiner Managementarchitektur geeignete Managementinformation vorhanden ist, muß darauf geachtet werden, daß das Objektmodell möglichst einfach um neue Anwendungs- und Diensteklassen erweiterbar ist, was eine klare Trennung zwischen generischer und anwendungsspezifischer Managementinformation impliziert.

Um diese Trennung zu erreichen, haben wir eine auf dem ODP-Referenzmodell aufbauende Methodik vorgestellt, die darauf abzielt, generische Management-Objektklassen zu entwerfen, um bereits so nahe wie möglich an der Wurzel der Vererbungshierarchie ein Maximum an Managementinformation und -diensten zu definieren. Der Vorteil einer hohen Informationsdichte an der Spitze der Vererbungshierarchie bietet uns neben der Eigenschaft einer übersichtlichen Darstellung des Management-Objektmodells verteilter kooperativer Managementsysteme insbesondere den Zusatznutzen, konkrete Vorgaben an den Minimalumfang einer geeigneten Instrumentierung machen zu können. Ferner stellen diese Objektklassen bereits eine beträchtliche Anzahl an Managementinformation und -diensten bereit, die grundsätzlich von jeder verteilten Anwendung (und damit auch von verteilten kooperativen Managementsystemen) erbracht werden müssen. Den grundsätzlichen Forderungen nach aussagekräftigen Basisklassen und einem möglichst hohes Maß an Wiederverwendbarkeit wurde somit Rechnung getragen. Die Allgemeingültigkeit dieser generischen Basisklassen haben wir durch die Implementierung von Managementagenten für unterschiedliche Anwendungsklassen (z.B. Transaktionsmonitore) und Dienste (z.B. NFS, NIS und WWW) nachgewiesen.

Unsere Objektmodelle verteilter kooperativer Managementsysteme, die sowohl eigenständige Managementplattformen als auch Management-Gateways umfassen, stützen sich auf die generischen Basisklassen ab und verfügen somit einen reichhaltigen Informations- und Funktionsumfang, den wir anhand prototypischer Implementierung in der Praxis testen konnten.

Die vorliegende Arbeit leistet somit einen Beitrag zur Etablierung eines auf CORBA basierenden Enterprise Managements durch die Entwicklung einiger hierzu unabdingbarer Konzepte.

Neben dem bisher Gesagten wurden die folgenden weiteren Ergebnisse erreicht:

- Die Erkenntnis, daß die Common Object Request Broker Architecture nachweislich (durch unsere Prototypen) ein solides konzeptionelles Fundament für das Enterprise Management bietet; um im realen Betrieb eingesetzt werden zu können, müssen jedoch noch einige spezifische Managementdienste standardisiert werden.
- Diese spezifischen Managementdienste bestehen zu einem gewissen Teil bereits in anderen Architekturen (insbesondere OSI/TMN); es wurde ein Verfahren entwickelt und vorgestellt, um diese Dienste in die OMG–Referenzarchitektur einzubetten. Für das Management von Managementsystemen zusätzlich erforderliche Managementinformation und -dienste wurden aus den Anforderungen abgeleitet, anschließend objektorientiert modelliert und in CORBA implementiert.

7.2 Zukünftige Forschungsfragestellungen

Die in dieser Arbeit verfolgte Betrachtung des Managements als eine spezifische Ausprägungsform einer verteilten Anwendung hat durch die Verwendung zeitgemäßer Technologien für die objektorientierte Analyse und Design einen neuen Entwicklungsansatz für Managementagenten vorgestellt, der zu einem späteren Zeitpunkt im Rahmen der *Web-based Enterprise Management (WBEM)* Initiative auf ähnliche Weise für Managementzwecke verwendet worden ist.

- Der CIM-Ansatz, unterschiedliche Architekturen in ein einheitliches Informationsmodell zu transformieren und dann jeweils wieder auf unterschiedliche Architekturen abzubilden mag zwar plausibel klingen, ist jedoch aufgrund der fundamentalen semantischen Unterschiede der Managementarchitekturen kaum praktikabel. Kapitel 4 hat aufgezeigt, daß die Integration unterschiedlicher Managementarchitekturen alle Teilmodelle einer Managementarchitektur umfaßt und sich bei weitem nicht auf ein Compilerbauproblem reduzieren läßt, wie es CIM unterstellt: Beispielhaft seien hier die sogenannten „Pushbutton“-Variablen des SNMP-Managementmodells aufgeführt, aus deren syntaktischer Beschreibung nicht ersichtlich ist, daß sie bei der Überführung in ein objektorientiertes Informationsmodell zu Methoden einer Objektklasse werden müßten. Ein entsprechender Übersetzer würde daraus Attribute einer Objektklasse generieren. Hierbei ist auch das CIM „Scratchpad“, das semantische Unterschiede abdecken helfen soll, keine Hilfe bei der automatischen Transformation. Ferner bleiben dabei die Spezifika einzelner Managementarchitekturen (wie das OSI Scoping und Filtering) sowie die architekturspezifischen Mechanismen zur Ereignisfilterung und -propagierung vollständig unbeachtet.

- Die formale Überführung von Behaviour-Definitionen ist auch weiterhin als ungelöstes Problem anzusehen, da diese bisher lediglich in informeller Weise spezifiziert werden. Hier könnten gegenwärtige Aktivitäten der ISO helfen, die darauf abzielen, Behaviour-Definitionen in einer formalen Notation (Z) zu definieren. Alternative Ansätze wie die Einbeziehung von KI-Sprachen (wie z.B. Prolog) wären ebenfalls eine Alternative.

- Ein weiterer Punkt, der im Rahmen unserer Arbeit in Ermangelung entsprechender generischer Dienste lediglich durch die Abstützung auf proprietäre Merkmale der von uns verwendeten CORBA-Entwicklungsumgebungen erreicht werden konnte, ist die **dynamische Erweiterbarkeit** von Management-Gateways. Um neue Ressourcenklassen zur Laufzeit den Gateways und somit auch dem Umbrella Management verfügbar zu machen, müssen jeweils

  1. neue, für diese Ressourcen zuständige Gateways erzeugt werden. Dies besteht aus der MIB-Übersetzung und einer anschließenden Neucompilierung des Gateways, was die Anwendbarkeit dieses Verfahrens in der Umgebung von Netzbe-

treibern signifikant einschränkt, da hierfür ein vollständiges Entwicklungssystem beim Anwender vorhanden sein muß.

2. Eine Alternative dazu besteht darin, diese Gateways zur Laufzeit um neue Managementinformationen erweitern zu können.

Die bisher allgemein verfügbaren Lösungen implizieren die erste Alternative, d.h. das Vorhandensein statischer Gateways. Wünschenswert ist jedoch die zweite Alternative, nämlich **dynamische Gateways**, die neben den oben genannten Vorteilen insbesondere durch folgende Aspekte begründet sind. Häufig sind die Objektmodelle von Managementagenten Modifikationen unterworfen: Datentypen von Attributen eines Agenten können geändert bzw. neue Ressourcen einem System hinzugefügt werden. Ein Management-Gateway steht dann vor dem Problem, mit Managementobjekten kommunizieren zu müssen, deren Typen zu seiner Übersetzungszeit unbekannt waren. Es ist daher notwendig, Informationen bereitzustellen, welche die Typen der Managementojekte beschreiben. Ein Management-Gateway kann dann zur Laufzeit ermitteln, über welche Datentypen Managementagenten verfügen. Solche Metainformation wird von sog. *Typagenten* angeboten, die in der neuesten Version des TMN/C++ API durch die *Metadata Services* realisiert werden. Diese standen uns jedoch bei der Implementierung unseres CMIP/SNMP Gateways nicht zur Verfügung. Das CORBA-Analogon hierzu besteht im *Meta-Object Service*, der sich jedoch ebenfalls noch in der Standardisierungsphase befindet.

- Die dynamische Erweiterbarkeit von Managementagenten um neue Funktionalität konnte von uns in Ermangelung geeigneter Entwicklungswerkzeuge nicht abschließend prototypisch implementiert werden. Unsere Erfahrungen mit der OMG *Mobile Agents Facility Interoperability* Spezifikation haben jedoch aufgezeigt, daß diese ein praktikabler Lösungsweg ist. Ferner bietet das *Java Dynamic Management Kit (JDMK)* in seiner neuesten Version ebenfalls die Möglichkeit, bestehende Systeme zur Laufzeit um neue Managementfunktionalität zu ergänzen.

- Kombiniert man die im vorigen Punkt angesprochenen Technologien zur dynamischen Delegierung von Managementdiensten mit dem *Pull*-Modell, so erreicht man damit eine Erweiterung bisheriger Agenten zu **selbststeuernden Systemen**, die autonom in Abhängigkeit des Auftretens von Fehlersituationen entscheiden, welche Art von Managementdiensten zur Problemdiagnose und -behebung erforderlich sind. Man erreicht so ein vollständig dezentrales Management, das eine spürbare Verbesserung hinsichtlich der Skalierbarkeit des Managements mit sich bringt.

Diese Gesichtspunkte sind in zukünftige Überlegungen einzubeziehen, die darauf abzielen, die in dieser Arbeit entwickelten Konzepte fortzuführen.

Anhang A

Common Object Request Broker Architecture (CORBA)

Wir stellen in diesem Anhang kurz einige grundlegende Begriffe und Mechanismen der in der vorliegenden Arbeit verwendeten Basisarchitektur vor. Sie sind einerseits für das Verständnis dieser Arbeit unumgänglich und können andererseits aufgrund des geringen Alters und der damit einhergehenden hohen Änderungsdynamik von CORBA nicht als selbstverständlich vorausgesetzt werden. Aufgrund der allgemein gehaltenen Ausrichtung von CORBA als objektorientierte Architektur für verteilte Verarbeitung finden sich hier keine für das Management spezifischen Aspekte; diese werden vielmehr in Abschnitt 3.1.2 behandelt. Der an einführenden Informationen zu CORBA interessierte Leser findet diese beispielsweise in [Sieg 96], [OrHE 95], [OrHa 98] und [VoDu 98].

A.1 Das OMG-Referenzmodell

Die **Object Management Architecture (OMA)** [Sole 95] der OMG für verteilte objektorientierte Programmierung erlaubt ortsungebundene Kooperation von Objekten in verteilter Umgebung. Je nach Implementierung können diese Objekte die Rolle eines Clients, eines Servers oder beide Rollen zugleich übernehmen. Im Gegensatz zum Manager/Agent-Paradigma, das im OSI-Management und im SNMP-basierten Management verwendet wird, liegen hier symmetrische Kooperationsbeziehungen zwischen einzelnen Objekten vor. Die OMA bildet den Rahmen für die fünf folgenden Teilaspekte und deren Architekturen (siehe auch Abbildung A.1).

Der **Object Request Broker (ORB)** bildet das Kommunikationsmedium für verteilte Objekte und leitet Anfragen bzw. Resultate in heterogener Umgebung von einem Objekt zu einem anderen weiter. Seine Hauptaufgabe besteht in der Abstraktion von Systemspezifika wie die verwendeten Netzprotokolle, Betriebssysteme und Programmiersprachen, in denen Objekte implementiert werden und verschattet somit die Heterogenität der einzelnen Ablaufumgebungen. Die Architektur, die die dafür notwendige Infrastruktur spezifiziert, wird **Common Object Request Broker Architecture (CORBA)** [CORBA 2.2] genannt.

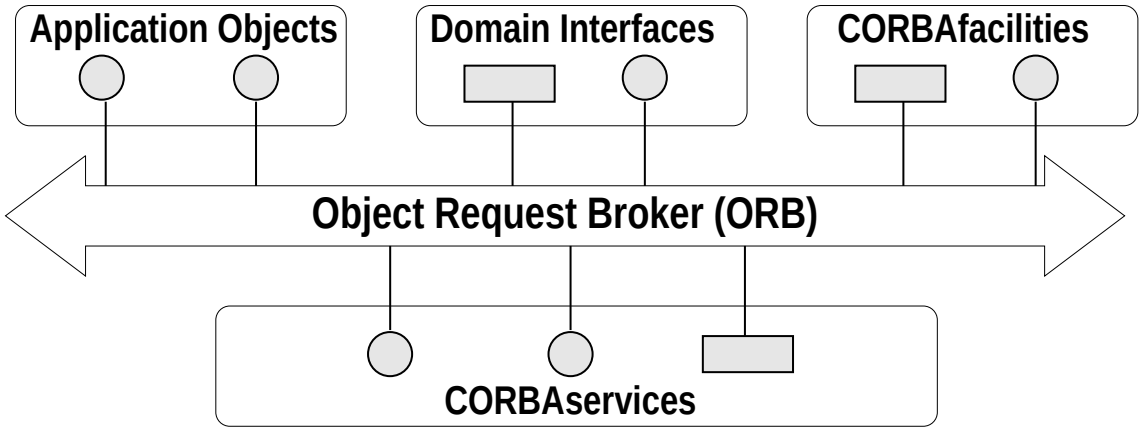


Abbildung A.1: Object Management Architecture

Die **Object Services Architecture (OSA)** [OMG 95-1-47] umfaßt grundlegende Dienste, die Basisfunktionalität bereitstellen, um Objekte in verteilter Umgebung überhaupt nutzen zu können. Hierzu zählen z.B. Dienste zur Instantiierung, Namensgebung und zur dauerhaften Speicherung von Objekten sowie zum Empfang bzw. Versand von Ereignismeldungen. **CORBAservices**[1] werden von der OMG genormt und sind für CORBA-konforme Systeme verbindlich. Bisher sind im Rahmen der *Common Object Services Specification (COSS)* [OMGSS 97] insgesamt 18 CORBAservices spezifiziert worden. Weitere elementare Objektdienste befinden sich in der Normierungsphase. Nützliche Managementfunktionalität für einzelne Objekte wird beispielsweise von den Diensten *Property*, *Query*, *Licensing* oder *Change Management* bereitgestellt.

Die darauf aufbauenden **CORBAfacilities** (früher: Common Facilities) sind universell verwendbare höherwertige Dienste, die für eine Vielzahl von Applikationen notwendig sind. Sie werden innerhalb der **Common Facilities Architecture (CFA)** [OMG 95-1-2] spezifiziert, die insgesamt vier Arten von **CORBAfacilities** beinhaltet. Sie werden in die Bereiche *User Interface*, *Information Management*, *Task Management* und *Systems Management* eingeteilt. Letztere bilden die Grundlage, um Managementdienste in das Rahmenwerk der OMG einzubringen; bestehende und neue Dienste, die für unsere Aufgabenstellung relevant sind, werden in Abschnitt 3.1.2 sowie Kapitel 5 vorgestellt.

Domain Interfaces (kurz: CORBAdomains) sind Dienste, die lediglich innerhalb eines bestimmten Anwendungsbereichs relevant sind. Beispiele hierfür sind Dienste für das Gesundheitswesen, die Telekommunikation, Echtzeitanwendungen oder die Finanzwelt. Die Dienstkategorie **CORBAdomains** wurde im Zuge der OMG-Restrukturierung im Juni 1995 von den CORBAfacilities getrennt, um Endanwendern im Rahmen sogenannter „Domain Task Forces“ (DTF) eine geeignete Plattform zur Spezifikation von Diensten zu bie-

[1] Im Juni 1995 wurde die bisher gültige Bezeichnung *Object Services* in *CORBAservices* geändert, um den unmittelbaren Bezug dieser Dienste zu CORBA zu verdeutlichen. In der Literatur werden gegenwärtig beide Begriffe synonym verwendet.

ten, die ihre bereichsspezifischen Anforderungen berücksichtigen. Der Erfolg dieser Maßnahme zur Einbeziehung von Anwendern läßt sich an der hohen Zahl der DTFs ablesen, die gegenwärtig ein Dutzend übersteigt. Nicht zuletzt ist die objektorientierte Analyse- und Designmethodik *Unified Modeling Language (UML)*, die als Nachfolger von OMT gedacht ist, mit dem die in dieser Arbeit vorgestellten Objektmodelle entworfen wurden, zu einem OMG-Standard herangereift. Naturgemäß ist die Zahl der CORBAdomains ist im Vergleich zu den beiden vorigen Dienstarten a priori unbeschränkt, da sich in ihnen die Vielfalt der möglichen Anwendungsbereiche widerspiegelt. CORBAdomains durchlaufen einen ähnlichen Standardisierungsprozeß wie CORBAservices und CORBAfacilities, sind jedoch optional für CORBA-konforme Implementierungen.

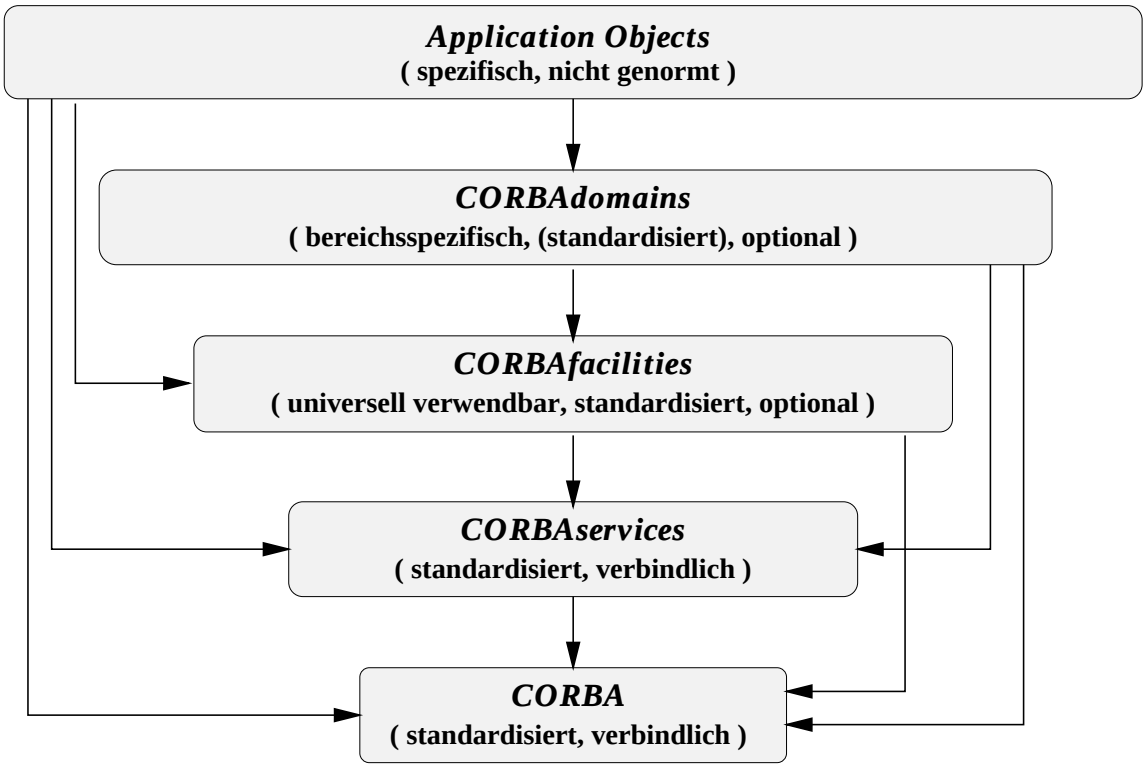


Abbildung A.2: Vererbungsbeziehungen zwischen OMA-Dienstekategorien

Für das integrierte Management besonders interessant ist die Telecom DTF, welche sich zum Ziel gesetzt hat, CORBA-basierte Dienste für den Bereich der Telekommunikation zu definieren. Diese umfassen sowohl die Erweiterung von CORBA zur Überwachung von Audio/Videoströmen als auch Dienste zur Ereignisfilterung oder Topologieverwaltung. In jüngster Zeit bildet diese Task Force das Zentrum der Bemühungen, CORBA für den Bereich der intelligenten Netze zu nutzen und Übergänge zwischen CORBA und TMN zu schaffen.

Den letzten Teilaspekt, der unter dem Gesichtspunkt der OMA behandelt wird, stellen die **Application Objects** dar. Sie unterliegen aufgrund ihrer Vielfalt keiner Normierung

durch die OMG und bilden somit auch keine eigenständige Teilarchitektur der OMA. Sie sind die eigentlichen Anwendungen wie zum Beispiel CAD-Systeme, CASE-Tools und Netzmanagement-Plattformen.

Zwischen den OMA-Teilarchitekturen bestehen Abhängigkeitsbeziehungen (siehe Abb. A.2), die sich dank der Vererbungseigenschaften objektorientierter Systeme einfach implementieren lassen. Die Wurzel der Vererbungshierarchie stellen die CORBAservices dar, da sie die grundlegenden Funktionen bereitstellen, um Objekte in einer CORBA-Umgebung überhaupt nutzen zu können. Darauf bauen die CORBAfacilities auf, die die zugrundeliegenden Objektdienste verwenden und um höhere Funktionalität erweitern. Natürlich können auch zwischen Objektklassen, die zur gleichen Dienstkategorie gehören, Vererbungsrelationen bestehen. Die höchste Kategorie besteht aus den Application Objects, die sich aus CORBAfacilities und gegebenenfalls zusätzlichen CORBAservices sowie dem eigentlichen Programmcode zusammensetzen. Die Abgrenzungen zwischen Application Objects, CORBAdomains, CORBAfacilities und CORBAservices sind natürlich nicht exakt festlegbar. Häufig in Applikationen vorkommende Softwaremodule sind potentielle Kandidaten für neue CORBAdomains oder CORBAfacilities; ebenso ist es möglich, daß Funktionalität von den CORBAfacilities hin zu den CORBAservices verlagert wird.

A.2 Die Bestandteile von CORBA

Die Schnittstellen von CORBA-konformen Objekten werden in einer programmiersprachenunabhängigen Notation, der sogenannten *OMG Interface Definition Language, (OMG IDL)*[2] definiert. Hierdurch wird sichergestellt, daß sämtliche Objekte eines CORBA-Systems auf einheitliche Art angesprochen werden können. Clients rufen Methoden auf Server-Objekten auf, ohne beachten zu müssen, auf welcher Systemarchitektur und unter welchem Betriebssystem diese laufen; die Frage, in welcher Programmiersprache diese Objekte implementiert sind, ist für das Aufrufen von Methoden (und damit für die Kommunikation zwischen Objekten) ebenfalls irrelevant.

Jede Objektklasse wird durch eine OMG IDL-Schnittstelle beschrieben, die sowohl die (privaten) Attribute der Klasse als auch deren (öffentliche) Methoden beschreibt; Objektklassen können Eigenschaften von einer oder mehrerer anderer Klassen erben. Neue Server-Objekte werden dem Object Request Broker (ORB) durch deren Eintragung im **Interface Repository**, einem ORB-weit gültigen Speicher für Schnittstellendefinitionen, bekanntgegeben. Die OMG legt über standardisierte *language mappings* die Abbildung von IDL auf konkrete Programmiersprachen wie C, C++, Java, Ada oder Smalltalk fest. Anhand dieser Abbildungsvorschrift generiert ein IDL-Compiler das **Server Skeleton** in

[2]Der Präfix OMG wird verwendet, um eine Verwechslung mit der Schnittstellenbeschreibungssprache von DCE, *DCE IDL* auszuschließen. Während DCE IDL sich an der Programmiersprache C orientiert, verwendet OMG IDL Konstrukte von C++.

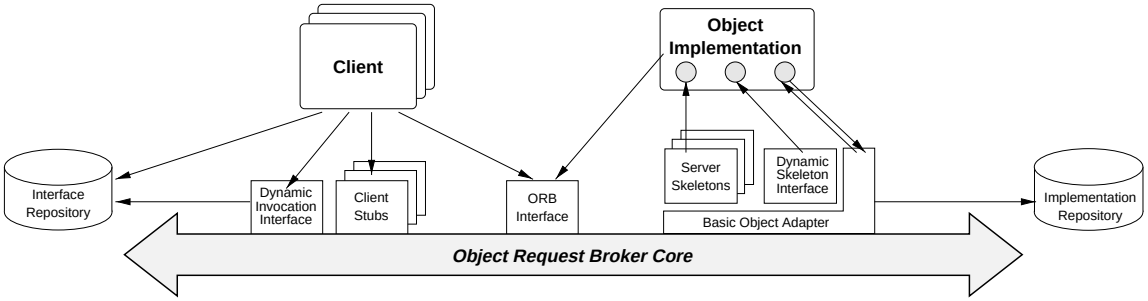


Abbildung A.3: Die Komponenten von CORBA

der jeweiligen Implementierungssprache. Der Benutzer eines Objekts erzeugt ebenfalls mit Hilfe eines IDL-Compilers den **Client Stub**, der ein Zugriff auf das entfernte Server-Objekt wie auf eine lokale Klasse ermöglicht.

Vom IDL-Compiler generierte Client Stubs und Server Skeletons bilden eine statische Schnittstelle zur Nutzung von Diensten (siehe auch Abbildung A.3. Damit eine Anwendung auch dynamisch zur Laufzeit Dienste von Server-Objekten in Anspruch nehmen kann, deren IDL-Beschreibung zur Zeit der Übersetzung nicht vorlag, definiert CORBA das **Dynamic Invocation Interface (DII)**, über das Abfragen z.B. hinsichtlich der Eigenschaften von Methodenparametern zur Laufzeit oder dem Typ von Objekten gestellt werden. Zu allen im System registrierten Serverobjekten enthält das Interface Repository die IDL-Beschreibungen ihrer Schnittstellen. Mit den Typinformationen aus dieser Datenbank kann ein Client dynamisch über das DII einen Methodenaufruf an einen Server absetzen. Die höhere Flexibilität des DII im Vergleich zu den Client Stubs besteht darin, daß ein Client das Zielobjekt über das DII *zur Laufzeit*, d.h. dynamisch adressiert, während dies beim Client Stub zur *Übersetzungszeit* geschieht; der Preis für diese Flexibilität liegt in der komplizierteren Implementierung von Client-Aufrufen, da vor dem Aufruf eine oder mehrere Abfragen an das Interface Repository gestellt werden müssen. Der angesprochene Server kann DII-Aufrufe nicht von Stub-Aufrufen unterscheiden.

Auf Server-Seite ist das **Dynamic Skeleton Interface (DSI)** das Äquivalent zum DII. Über diese Schnittstelle können Aufrufe an dynamisch erzeugte Objektimplementierungen übergeben werden. Diese Funktionalität wird u.a. für Gateways zu anderen Objektsystemen oder für generische Brücken zwischen ORBs benötigt.

Gewöhnlich sind Aufrufe über das statische Interface synchron: Der Client blockiert, bis er vom ORB über den Stub ein Ergebnis oder eine Fehlermeldung (Exception) erhält. Werden Operationen in IDL mit dem Schlüsselwort *oneway* versehen, benutzt CORBA hierfür eine asynchrone und unzuverlässige (*best effort*) Aufrufsemantik, bei der keine Ergebnisse oder Ausführungsbestätigungen zurückgegeben werden. Über das DII können Methoden zusätzlich auch asynchron *mit* Ergebnisrückgabe ausgeführt werden (*deferred synchronous*). Der Client blockiert hierbei nicht. Über einen Polling-Mechanismus wird überprüft, ob ein Ergebnis vorliegt.

Clients identifizieren das Objekt, auf dem sie eine Methode aufrufen möchten, gegenüber dem ORB durch eine eindeutige Objektreferenz (**Inter-operable Object Reference (IOR)**), ohne über den physikalischen Ort des Objekts Kenntnis besitzen zu müssen. Der ORB unterhält mit dem **Implementation Repository** ein Verzeichnis, welches u.a. die Abbildung von IOR auf reale Objekte erlaubt. Damit dieser Mechanismus funktioniert, müssen sich Server-Objekte über den **Basic Object Adapter (BOA)** beim ORB registrieren. Der BOA generiert für ein neues Objekt die IOR, die der ORB in seinem Verzeichnis ablegt. Auch das Aktivieren von registrierten, aber nicht gestarteten Servern bei Eintreffen einer Client-Anfrage beim ORB kann der BOA übernehmen. Clients können die Objektreferenzen zu benötigten Diensten auf verschiedene Weise ermitteln: Neben Verzeichnis- (Naming Service) und Vermittlungsdiensten (Trading Service) können auch Dateien, die IORs als Zeichenketten (*stringified object references*) enthalten, als Informationsquellen dienen, ebenso weitere, proprietäre Mechanismen. Hinter dem **ORB Interface** verbergen sich verschiedene nützliche Funktionen sowohl für Clients als auch für Server (z.B. zur Erzeugung von stringified IORs und zur Konvertierung unterschiedlicher Datentypen).

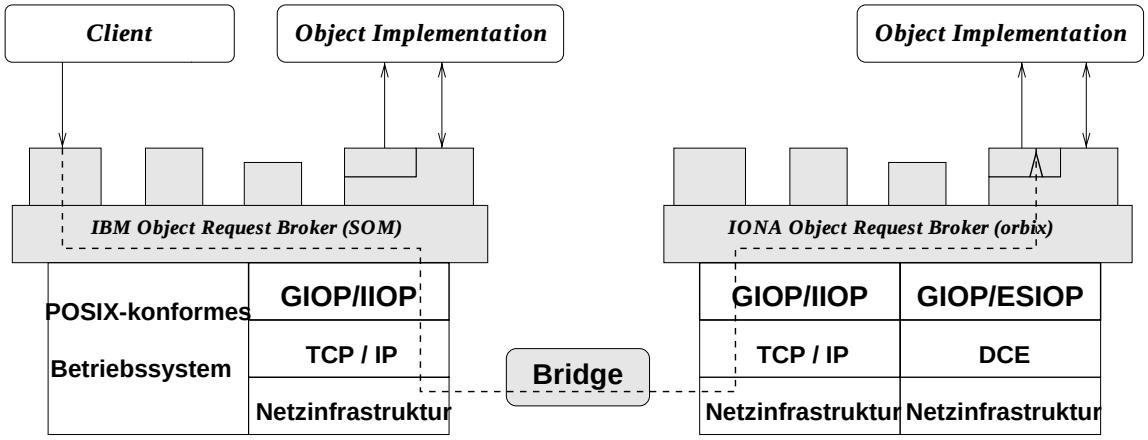


Abbildung A.4: Standardisierte Protokollstacks zur CORBA-Interoperabilität

Seit der Version 2.0 wird die Interoperabilität zwischen ORBs verschiedener Hersteller durch das **General Inter-ORB Protocol (GIOP)** gewährleistet, dessen Abbildung auf unterschiedliche Protokolle standardisiert wurde. Jeder ORB muß mindestens die Kooperation auf Basis von TCP/IP über das **Internet Inter-ORB-Protocol (IIOP)** unterstützen. Eventuell vorhandene proprietäre ORB-Protokolle werden mit Hilfe sogenannter *Bridges* integriert, die die Konvertierung des proprietären Protokolls in IIOP vornehmen. Darüberhinaus wurde bisher unter dem Namen **DCE Common Inter-ORB-Protocol (DCE-CIOP)** ein **Environment-specific Inter-ORB Protocol (ESIOP)** standardisiert, das eine Abbildung von CORBA auf DCE realisiert. Weitere ESIOPs für SNA, OSI und SS.7 sind zwar angedacht, jedoch nicht in den Standardisierungsprozeß eingebracht worden. Abbildung A.4 stellt die Mechanismen zur Interoperabilität graphisch dar. Andere verteilte Objektumgebungen wie Java RMI von *JavaSoft* oder *Microsoft* DCOM/ActiveX können

mit Hilfe von Gateways in CORBA integriert werden. Obwohl Mechanismen zum *Interworking* mit DCOM Bestandteil des CORBA-Standards ist, sind bisher nur sehr wenige Implementierungen erfolgt.

Anhang B

Transformation von Management-Informationsmodellen

Während Abschnitt 4.4.3 aus Platzgründen die Eigenschaften und Regeln der Transformation von Management-Informationsmodellen lediglich skizziert, beschreibt dieser Anhang anhand einiger charakteristischer Elemente der MIB-II [rfc1213] detailliert die Überführung des Internet-Informationsmodells in das OSI-Informationsmodell mit Hilfe des IIMC-Algorithmus (Abschnitt B.1) bzw. in IDL-Beschreibungen von CORBA-Objekten unter Zuhilfenahme des JIDM-Algorithmus (Abschnitt B.2).

B.1 Direkte Übersetzung von Internet–SMI nach GDMO

In 1993 entwarf die Arbeitsgruppe *ISO-Internet Management Coexistence (IIMC)* des Network Management Forums einen Algorithmus [NMF 26] zur Übersetzung von Internet MIB-Beschreibungen in OSI-konforme MOCs. Das im Rahmen der vorliegenden Arbeit entwickelte CMIP/SNMP-Gateway arbeitet beruht auf den dort festgelegten Übersetzungsregeln.

B.1.1 Registrierung und Namensgebung

Registrierung und Namensgebung sind für die eindeutige Identifikation von Managementinformation entscheidend. Erstere sichert dabei die globale Eindeutigkeit der Managementinformation während letztere für die Zusammensetzung des *Distinguished Name (DN)* einer Instanz und damit für die Lokalisierung dieser Instanz innerhalb einer MIB verantwortlich ist. Bei der Übersetzung von Internet–MIBs in OSI–Objektbeschreibungen entstehen neue Objektklassen, Attribute und Beschreibungen von Notifikationen, die zunächst registriert werden müssen. Außerdem müssen auch die *NAME BINDINGs*, die für den Aufbau der Containment Hierarchie maßgeblich sind, registriert werden.

Grundsätzlich wird in Internet–SMI Managementinformation mit Hilfe eines ASN.1 *Object Identifier (OID)* registriert. Ein OID bezeichnet dabei einen eindeutigen Pfad im Internet–Registrierungsbaum (Präfix: 1.3.6.1 bzw. iso.org.dod.internet) bzw. in einem proprietären Teilbaum des ISO–Registrierungsbaumes. Die Eindeutigkeit der Registrierung von Internet-Managementinformation wird dadurch ausgenutzt, daß die daraus resultierende Managementinformation in GDMO in einem anderen Teilbaum des ISO-Registrierungsbaumes mit der Internet-OID abgespeichert wird. Das folgende Beispiel zeigt die Registrierung:

- Registrierungsteilbäume für die übersetzten Managementinformationen :
  Für Klassen und Attribute : iimcAutoObjAndAttr = .1.2.124.360501.14.1
  Für NAME BINDINGs : iimcAutoNameBinding = .1.2.124.360501.14.2
  Für Naming Attribute : iimcAutoNaming = .1.2.124.360501.14.3
- Internet-Managementinformation : sysContact
  Registriert unter : .1.3.6.1.2.1.1
- Übersetzte GDMO-Managementinformation : sysContact (Attribut)
  Registriert unter : .1.2.124.360501.14.1.1.3.6.1.2.1.1
  (iimcAutoObjAndAttr konkateniert mit der Internet OID)

Name Bindings sowie die dazugehörigen *Naming Attribute* (s.u.) werden entsprechend unter *iimcAutoNameBinding* bzw. *iimcAutoNaming* registriert.

Die Namensgebung für ein OSI-Managementobjekt erfolgt durch ein gesondert gekennzeichnetes Attribut, das sogenannte *Naming Attribute*. Der Name des Attributs und sein Inhalt ergeben zusammen den *Relative Distinguished Name (RDN)* der Instanz. Der IIMC Algorithmus spezifiziert dabei für das *Naming Attribute* jeweils eine Regel für die Namensgebung, die Registrierung, die Werte-Definition und die automatische Generierung des *Naming Attribute* für jede MOC, die aus der Übersetzung einer Internet–MIB entsteht:

- Namensgebung des Naming Attributes:
  An den Namen derjenigen Objekt-Klasse, zu welcher das Attribut gehört, wird der Suffix „Id“ angefügt.
- Die Registrierung des Naming Attributes erfolgt im Teilbaum *iimcAutoNaming* mit dem Internet-OID derjenigen Objektklasse, zu der das Attribut gehört.
- Dem Wert des Attributes wird entweder ASN.1 NULL oder der Inhalt der Internet-MIB-Variable *INDEX* aufgrund folgender Regeln zugewiesen:
  - Für Managementobjekte, von denen nur eine Instanz pro Internet-MIB existieren darf – beispielsweise diejenigen OSI-MOCs, die aus den Internet MIB-II Gruppen system, tcp, ip usw. entstanden sind – wird der Wert des Attributes auf ASN.1 NULL gesetzt.

– Für Managementobjekte, von denen mehrere Instanzen pro Internet MIB existieren können – beispielsweise diejenigen OSI-MOCs, die Tabellenzeilen einer Internet-MIB repräsentieren (Beispiel MIB-II: tcpConnEntry, ipAddrEntry – wird dem Wert des Attributes der jeweilige *INDEX* der SNMP-Tabellenzeile zugewiesen.

B.1.2 Der IIMC-Algorithmus

Die Übersetzung einer Internet-MIB in OSI-MOCs vollzieht sich in mehreren Etappen, die jeweils vom zu übersetzenden Datentyp abhängen und nachfolgend wiedergegeben sind.

Übersetzen von Gruppen

Internet Gruppen (zum Beispiel aus der MIB-2: system, ip ,tcp etc.) werden zu OSI-MOCs übersetzt:
Gruppe system der MIB-II:

```
-- groups in MIB-II
...
system       OBJECT IDENTIFIER ::= { mib-2 1 }
...
```

Übersetzte OSI-MOC internetSystem; die Umbenennung erfolgte, um die Namenskollision mit der Wurzel der OSI Containment Hierarchie auszuschließen:

```
internetSystem MANAGED OBJECT CLASS
        DERIVED FROM
           "CCITT Rec. X.721 (1992) |
                ISO/IEC 10165-2 : 1992": top;
        CHARACTERIZED BY internetSystemPkg PACKAGE
        BEHAVIOUR
           internetSystemPkgBehaviour BEHAVIOUR
           DEFINED AS
           !BEGINPARSE
           REFERENCE
              !!This managed object class maps to the
              Internet system group with object id
              {mib-2 1} in RFC 1213.!!;
           DESCRIPTION
              !!If an agent is not configured to have a
              value for any of these variables, a string
              of length 0 is returned.
              [...]
```

```
   ENDPARSE!;;
ATTRIBUTES
   internetSystemId              GET,
   sysDescr                      GET,
   sysObjectID                   GET,
   sysUpTime                     GET,
   sysContact                    GET-REPLACE,
   sysName                       GET-REPLACE,
   sysLocation                   GET-REPLACE,
   sysServices                   GET;
NOTIFICATIONS
         "IIMC MIB Translation":internetAlarm;;;
REGISTERED AS
   {1 2 124 360501 14 1 1 3 6 1 2 1 1};
```

Übersetzen von Tabellen

Hier erfolgt keine direkte Übersetzung der Tabellendefinition (zum Beispiel tcpConnTable, ipAddrTable, usw.) selbst. Es werden lediglich die Tabellen*zeilen* (zum Beispiel tcpConnEntry, ipAddrEntry, usw.) in OSI-MOCs übersetzt.

Das folgende Beispiel der Tabellenzeilendefinition ipAddrEntry der MIB-II soll dies verdeutlichen:

```
...
ipAddrEntry OBJECT-TYPE
      SYNTAX  IpAddrEntry
      ACCESS  not-accessible
      STATUS  mandatory
      DESCRIPTION
         "The addressing information for one of this
         entity's IP addresses."
      INDEX   { ipAdEntAddr }
      ::= { ipAddrTable 1 }

IpAddrEntry ::=
      SEQUENCE {
          ipAdEntAddr
              IpAddress,
          ipAdEntIfIndex
              INTEGER,
          ipAdEntNetMask
              IpAddress,
```

```
                ipAdEntBcastAddr
                    INTEGER,
                ipAdEntReasmMaxSize
                    INTEGER (0..65535)
            }
        ...
```

Die OSI-MOC ipAddrEntry lautet:

```
ipAddrEntry MANAGED OBJECT CLASS
        DERIVED FROM
           "CCITT Rec. X.721 (1992) |
                ISO/IEC 10165-2 : 1992": top;
        CHARACTERIZED BY ipAddrEntryPkg PACKAGE
        BEHAVIOUR
           ipAddrEntryPkgBehaviour BEHAVIOUR
           DEFINED AS
           !BEGINPARSE
           REFERENCE
           !!This managed object class maps to the
           ipAddrEntry object with object id {ipAddrTable 1}
           in RFC 1213.!!;
           DESCRIPTION
              !!The addressing information for one of this
              entity's IP addresses.!!;
              INDEX RFC1213-MIB.ipAdEntAddr;
           ENDPARSE!;;
        ATTRIBUTES
           ipAddrEntryId            GET,
           ipAdEntAddr              GET,
           ipAdEntIfIndex           GET,
           ipAdEntNetMask           GET,
           ipAdEntBcastAddr         GET,
           ipAdEntReasmMaxSize      GET;;;
        REGISTERED AS
           { 1 2 124 360501 14 1 1 3 6 1 2 1 4 20 1};
```

Skalare MIB-Variablen

Alle skalaren MIB-Variablen einer Internet-MIB, werden – wie hier am Beispiel der MIB-II Variablen sysContact aus der system-Gruppe der MIB-II gezeigt wird – zu MOC Attributen in derjenigen Klasse, die aus der übergeordneten Gruppe entstanden ist:

```
[...]
sysContact OBJECT-TYPE
      SYNTAX  DisplayString (SIZE (0..255))
      ACCESS  read-write
      STATUS  mandatory
      DESCRIPTION
         "The textual identification of the
         contact person for this managed node,
         together with information on how to
         contact this person."
::= { system 4 }
[...]
```

Das entsprechende MOC-Attribut sysContact lautet:

```
sysContact ATTRIBUTE
        WITH ATTRIBUTE SYNTAX
                IIMCRFC12131354ASN1.DisplayString;
        BEHAVIOUR
           sysContactBehaviour BEHAVIOUR
           DEFINED AS
              !BEGINPARSE
              REFERENCE
              !!This attribute maps to sysContact with
              object id {system 4} in RFC1213.!!;
              DESCRIPTION
                 !!The textual identification of the
                 contact person for this managed node,
                 together with information
                 on how to contact this person.!!;
              ENDPARSE!;;
        REGISTERED AS
           {1 2 124 360501 14 1 1 3 6 1 2 1 1 4};
```

Übersetzung asynchroner Ereignismeldungen

Ereignismeldungen werden in der Internet-Managementarchitektur nicht im Informationsmodell, sondern im *Kommunikationsmodell* definiert[1]. Dies impliziert, daß Traps, die von einem SNMP-Agenten zu einem SNMP-Manager ausgesandt werden, nicht Teil der MIB, sondern ein Teil des SNMP-*Protokolls* sind. In der OSI-Managementarchitektur werden dagegen Notifikationen von speziellen Managementobjekten, sogenannten *Event Forwarding Discriminators (EFD)*, ausgelöst. Sie sind damit Teil des Informationsmodells und werden mit dem Notification Template spezifiziert.

Aufgrund dieser Unterschiede kann ein SNMP-Trap häufig keinem OSI-Managementobjekt eindeutig zugeordnet werden. Daher definiert der Algorithmus eine generische Notifikation, auf die *sämtliche* SNMP-Traps abgebildet werden. Generische Notifikationen sollen daher von der Instanz der jeweiligen MIB-Wurzel (hier: die MOC internetSystem) ausgesendet werden.

Generieren von NAME BINDING Templates

Für jede erstellte OSI-MOC muß zusätzlich ein NAME BINDING Template definiert werden. Das NAME BINDING Template definiert die Lage einer Instanz in der Containment-Hierarchie. Dabei dient die aus der übergeordneten SNMP-Gruppe entstandene OSI-MOC als *SUPERIOR OBJECT CLASS*. Der Internet-Registrierungsbaum wird somit auf die OSI Containment Hierarchie abgebildet.

Nachfolgend ist das NAME BINDING Template der OSI-MOC internetSystem in Bezug auf die übergeordnete OSI-MOC mib2 aufgeführt:

```
internetSystem-mib2NB  NAME BINDING
        SUBORDINATE OBJECT CLASS
           internetSystem  AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS
           mib2;
        WITH ATTRIBUTE internetSystemId;
        BEHAVIOUR
           internetSystem-systemNBBehaviour BEHAVIOUR
           DEFINED AS
           !BEGINPARSE
           INDEX     NULL;
           ENDPARSE!;;
        REGISTERED AS
           {1 2 124 360501 14 4 1 3 6 1 2 1 1 };
```

[1] Diese Aussage ist für das Internet Management-Framework in der zweiten Version von 1996 nicht mehr gültig: Dort werden Traps nämlich mit Hilfe des NOTIFICATION TYPE Makros definiert (siehe dazu auch Abschnitt B.2); demzufolge sind Ereignismeldungen neuerdings auch Teil des Informationsmodells.

B.2 Umsetzung von Internet-SMI in OMG IDL

B.2.1 Namenskonventionen

Im Gegensatz zu ASN.1-Namen wird bei den entsprechenden IDL-Identifikatoren nicht zwischen Groß- und Kleinbuchstaben unterschieden; andererseits umfaßt das Alphabet von ASN.1 einen wesentlich größeren Zeichenvorrat (darunter auch einige Sonderzeichen) als das von OMG IDL. Außerdem müssen die drei Namensräume von ASN.1 (Typreferenzen, Wertreferenzen und Identifikatoren) auf einen einzigen IDL-Namensraum abgebildet werden. Infolgedessen muß bei der Umsetzung darauf geachtet werden, daß keine Namenskonflikte auftreten. Damit die Abbildung nach IDL so einfach und direkt wie möglich bleibt, werden die Namen übernommen und müssen systematisch ergänzt werden, sodaß sie eindeutig sind.

B.2.2 Der JIDM-Algorithmus

Umsetzung der ASN.1-Datentypen

Tabelle B.1 zeigt die Umsetzung von (einigen) ASN.1-Datentypen in die Datentypen von OMG IDL. Da sie elementare Datentypen definieren, werden sie in einer Datei „ASN1Types.idl“ abgelegt und müssen in jede übersetzte MIB-Datei eingebunden werden.

ASN.1-Datentyp	IDL-Datentyp
`NULL`	typedef octet ASN1_Null;
`BOOLEAN`	typedef boolean ASN1_Boolean;
`INTEGER`	typedef long ASN1_Integer;
`REAL`	typedef double ASN1_Real;
`ENUMERATED`	enum { item1, ... ,itemn };
`BIT STRING`	typedef sequence<octet> ASN1_BitString;
`OCTET STRING`	typedef sequence<octet> ASN1_OctetString;
`STRING`	typedef sequence<octet> ... ; typedef string ... ; typedef sequence<long> ... ; typedef sequence<unsigned short> ... ;
`OBJECT IDENTIFIER`	typedef string ASN1_Objectidentifier;
`CHOICE`	union/switch
`SET,SEQUENCE`	struct
`SET,SEQUENCE OF type`	typedef sequence<type> ... ;

Tabelle B.1: Abbildung der Datentypen von ASN.1 nach IDL

Der Grund, weshalb der ASN.1-Datentyp `STRING` mehrere IDL-Entsprechungen hat, sind die verschiedenen Varianten dieses Typs: Ein `STRING` kann in ASN.1 eine einfache Zeichenkette oder eine Sequenz eines bestimmten Datentyps (z.B. ein 32-Bit-Wert) sein, wofür jeweils ein entsprechender IDL-Datentyp definiert werden muß.

Aus der obigen Liste der ASN.1-Datentypen sind in der Version 1 von Internet-SMI [rfc1155] nur die Datentypen `NULL, INTEGER, OCTET STRING` und `OBJECT IDENTIFIER` sowie die Datentypen `SEQUENCE, SEQUENCE OF` zugelassen. Sie werden verwendet, um die Datentypen des Internet-SMI in der Version 1 (Zähler, IP-Adresse etc.) zu definieren. Für diese gilt der obere Teil der Umsetzungstabelle B.2.

SNMP	IDL
`IpAddress`	sequence<octet, 4> IpAddressType;
`Counter`	typedef unsigned short CounterType;
`Gauge`	typedef unsigned short GaugeType;
`TimeTicks`	typedef unsigned short TimeTicksType;
`Opaque`	sequence<octet> OpaqueType;
Neue SNMPv2 Datentypen:	
`Integer32`	typedef long Integer32Type;
`Counter32`	typedef unsigned long Counter32Type;
`Gauge32`	typedef unsigned long Gauge32Type;
`TimeTicks`	typedef unsigned long TimeTicksType;
`Counter64`	typedef long Counter64Type;
`Unsigned32`	typedef unsigned long Unsigned32Type;

Tabelle B.2: Umsetzung der Datentypen von SNMP in OMG IDL

Das SNMPv2-Informationsmodell [rfc1902] erweitert die erste Version von Internet-SMI um die im zweiten Teil der Tabelle B.2 aufgeführten ASN.1-Datentypen.

Umsetzung von MIB-Modulen

Ein SNMPv2-Informationsmodul enthält die Beschreibung einer MIB mit Strukturierungs- und/oder Informationsknoten. Beispiel für ein solches Modul ist die MIB-II [rfc1213]:

```
RFC1213-MIB DEFINITIONS ::= BEGIN
IMPORTS
        mgmt, NetworkAddress, IpAddress, Counter, Gauge,
                TimeTicks
            FROM RFC1155-SMI
        OBJECT-TYPE
                FROM RFC-1212;
```

```
mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }

-- textual conventions
DisplayString ::=
    OCTET STRING
PhysAddress ::=
    OCTET STRING
-- groups in MIB-II

system       OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }
at           OBJECT IDENTIFIER ::= { mib-2 3 }
ip           OBJECT IDENTIFIER ::= { mib-2 4 }
icmp         OBJECT IDENTIFIER ::= { mib-2 5 }
tcp          OBJECT IDENTIFIER ::= { mib-2 6 }
udp          OBJECT IDENTIFIER ::= { mib-2 7 }
egp          OBJECT IDENTIFIER ::= { mib-2 8 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp         OBJECT IDENTIFIER ::= { mib-2 11 }

-- Formale Beschreibung der einzelnen Gruppen

END
```

Jedes Informationsmodul wird auf ein IDL-Modul mit gleichem Identifikator (im Beispiel `RFC1213-MIB`) abgebildet. Der Dateiname entspricht dem Modulnamen, der um das Suffix „.idl“ ergänzt wird. In diese Datei werden alle den ASN.1-Makros des Moduls entsprechenden IDL-Definitionen geschrieben; sie sieht für das obige Beispiel folgendermaßen aus:

```
// rfc1213.idl
#ifndef RFC1213_MIB_idl
#define RFC1213_MIB_idl
#include “SNMPMgmt.idl”
module RFC1213_MIB {
#include “rfc1155.idl”
typedef RFC1155_SMI::TimeTicksType TimeTicksType;
typedef RFC1155_SMI::GaugeType GaugeType;
typedef RFC1155_SMI::CounterType CounterType;
typedef RFC1155_SMI::IpAddressType IpAddressType;
typedef RFC1155_SMI::NetworkAddressType NetworkAddressType;
#define mgmt RFC1155_SMI::mgmt;
const ASN1_ObjectIdentifier mib_2 = “mgmt.1”;
```

```
const ASN1_ObjectIdentifier system = “mib_2.1”;
const ASN1_ObjectIdentifier interfaces = “mib_2.2”;
const ASN1_ObjectIdentifier at = “mib_2.3”;
const ASN1_ObjectIdentifier ip = “mib_2.4”;
const ASN1_ObjectIdentifier icmp = “mib_2.5”;
const ASN1_ObjectIdentifier tcp = “mib_2.6”;
const ASN1_ObjectIdentifier udp = “mib_2.7”;
const ASN1_ObjectIdentifier egp = “mib_2.8”;
const ASN1_ObjectIdentifier transmission = “mib_2.10”;
const ASN1_ObjectIdentifier snmp = “mib_2.11”;
// IDL-Beschreibung der Gruppen
}
#endif
```

Zu beachten ist, daß sämtliche Objektidentifikatoren als Konstanten deklariert werden; sie werden für die Adressierung der Managed Objects benötigt.

Abbildung von Gruppen und Tabellen

Wie oben bereits angedeutet, ist das SNMPv2-Informationsmodell umfangreicher als das von SNMP(v1) und schließt dieses zur Gewährleistung der Abwärtskompatibilität mit ein. MIB-Module von SNMP(v1) können unter Verwendung von [rfc1908] in SNMPv2-Module umgewandelt werden.

Analog zum IIMC-Algorithmus (vgl. dazu Abschnitt B.1) bildet der JIDM-Algorithmus [JIDM 97] zusammengesetzte Elemente von Internet-SMI folgendermaßen nach OMG IDL ab:

- Für jede SNMPv2-Tabellenzeile wird eine IDL-Objektklasse erzeugt. Die Spalteneinträge der ursprünglichen Tabelle werden zu Attributen dieser Klasse. Eine Instanz einer solchen Objektdefinition entspricht also genau einer Zeile der ursprünglichen Tabelle.
- Eine SNMP-Gruppe wird zu einer IDL-Objektklasse („Group Object“). Die in dieser Gruppe enthaltenen MIB-Variablen werden zu den Attributen der Klasse.

Jede neue Schnittstellendefinition wird von der Objektklasse SmiEntry abgeleitet, die die Wurzel der Vererbungshierarchie darstellt (s. Abb. B.1). Sie dient als Platzhalter für Attribute und Operationen, die einen generischen Zugriff auf Managementobjekte implementieren, welche durch den JIDM-Algorithmus in CORBA-konforme Objektklassen überführt wurden.

Gruppen- und Tabellenzeilenobjekte befinden sich in der Vererbungshierarchie der IDL-Schnittstellen auf der gleichen Ebene; somit ist die ursprüngliche Zugehörigkeit von einer Tabelle zu einer bestimmten SNMP-Gruppe (wie es beispielsweise der Fall für die

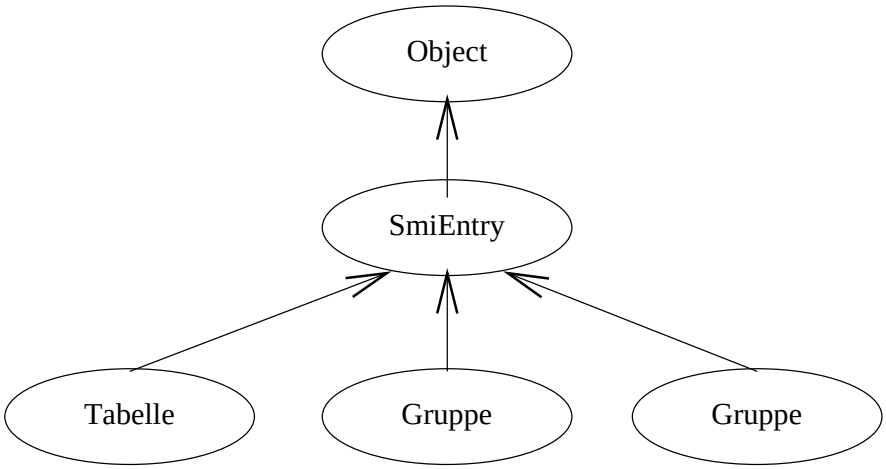


Abbildung B.1: Vererbungshierarchie für übersetzte Internet-SMI Tabellenzeilen und Gruppen

`ifTable` ist, die Bestandteil der `interfaces`-Gruppe ist) nicht mehr unmittelbar sichtbar. Die Zugehörigkeit von IDL-Zeilenobjekten und IDL-Gruppenobjekten ist durch die mit übernommenen Objektidentifikatoren allerdings weiterhin eindeutig.

Anhand der erwähnten SNMP-Gruppe `interfaces` soll nun beispielhaft der Transformationsvorgang detailliert besprochen werden. Ausgangspunkt ist die nachfolgende Definition dieser SNMP-Gruppe in [rfc1213]:

```
-- the Interfaces group

ifNumber OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
          ''The number of network interfaces ... ''
     ::= { interfaces 1 }

ifTable OBJECT-TYPE
     SYNTAX SEQUENCE OF IfEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
          '' A list of interface entries. ...''
     ::= { interfaces 2 }

ifEntry OBJECT-TYPE
```

```
     SYNTAX IfEntry
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
          ‘‘An interface entry containing objects ... ‘‘
     INDEX { ifIndex }
     ::= { ifTable 1 }

IfEntry ::=
     SEQUENCE {
          ifIndex INTEGER,
          ifDescr DisplayString,
          ...
          ifSpecific OBJECT IDENTIFIER
     }

ifIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
          ‘‘A unique value for each interface. ...’’
     ::= { ifEntry 1 }

ifDescr OBJECT-TYPE
     ...
     ::= { ifEntry 2 }

...

ifSpecific OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
          ‘‘A reference to MIB definitions specific ...’’
     ::= { ifEntry 22 }

-- the Address Translation group
...
```

Diese Gruppe besteht aus einer skalaren Variable (`ifNumber`) und einer Tabelle (`ifTable`) mit 22 Spalten (`ifEntry 1-22`). Durch den JIDM-Algorithmus werden somit je eine Schnittstellendefinition für ein Gruppenobjekt und für ein Tabellenzeilenobjekt erzeugt. Beide Klassen werden von der Klasse SmiEntry, die im Modul SNMPMgmt definiert wurde, abgeleitet.

Der Name der Klasse für das Gruppenobjekt ergibt sich aus dem Namen der SNMP-Gruppe und dem Suffix Group. Die skalare MIB-Variable `ifNumber` wird auf ein Attribut dieser Klasse abgebildet. Dabei werden der Name aus dem Deskriptor der Variablen, die Zugriffsarten aus der ACCESS-Klausel und der Datentyp aus der SYNTAX-Klausel der Variablen bestimmt. Da die SNMP-Gruppe keine weiteren skalaren Objekte besitzt, enthält die IDL-Definition nur dieses eine Attribut; die Tabelle der SNMP-Gruppe wird dabei ignoriert, da hierfür eine eigene IDL-Schnittstellendefinition vorgesehen ist. Die OID der SNMP-Variablen wird als Konstante definiert. Da diese Konstante den gleichen Namen trägt wie die Variable bzw. ihr entsprechendes Attribut, muß dies, um Namenskonflikte zu vermeiden, außerhalb der interface-Deklaration erfolgen:

```
...
interface InterfacesGroup:SNMPMgmt::SmiEntry {
/* DESCRIPTION:
“The number of network interfaces ...” */

readonly attribute ASN1_Integer ifNumber;

};
const ASN1_ObjectIdentifier ifNumber=”interfaces.1”;
```

Da eine Zeile einer SNMP-Tabelle eine Liste (SEQUENCE) von einzelnen skalaren Variablen ist (siehe [rfc1902]), ergibt sich der Name einer solchen Klasse aus dem Namen dieser Liste (und nicht aus dem Namen der Tabelle) sowie dem Suffix Object. In obigem Beispiel besteht eine Zeile der Tabelle `ifTable` aus den in der Liste `ifEntry` angeführten Spaltenelementen. In der zu erzeugenden Schnittstelle IfEntryObject werden die Spalteneinträge, also Variablen einfacher skalarer Typen, zu Attributen dieser neuen Klasse. Analog dazu werden Namen, Zugriffsrechte und Datentypen von den Variablen übernommen, die Objektidentifikatoren von `ifTable`, `ifEntry` und von den Spaltenelementen werden auf Strings abgebildet:

```
...
interface IfEntryObject:SNMPMgmt::SmiEntry {
/* INDEX ifEntry */
//————————————————-
      /* DESCRIPTION:
      “A unique value for each interface. ...” */

      readonly attribute ASN1_Integer ifIndex;
//————————————————-
      /* DESCRIPTION:
      “ A textual string ... “ */
      typedef sequence <DisplayStringType, 255> IfDescrType;

      readonly attribute IfDescrType ifDescr;
// ————————————————
      ...
      DESCRIPTION:
      ”A reference to MIB definitions specific ...” */

readonly attribute ASN1_ObjectIdentifier ifSpecific;
};
const ASN1_ObjectIdentifier ifTable=”interfaces.2”;
const ASN1_ObjectIdentifier ifEntry=”ifTable.1”;
const ASN1_ObjectIdentifier ifIndex=”ifEntry.1”;
const ASN1_ObjectIdentifier ifDescr=”ifEntry.2”;
...
const ASN1_ObjectIdentifier ifSpecific=”ifEntry.22”;
```

Abbildung asynchroner Ereignismeldungen

Ereignismeldungen sind beim Internet-Management SNMP-PDUs, die von einem SNMP-System (Manager oder Agent) asynchron an ein anderes System gesendet werden. Im Gegensatz dazu entspricht die Meldung eines Events in der CORBA-Umgebung im Prinzip dem Aufruf einer bestimmten (*Push*- oder *Pull*-)Methode eines Objektes (vgl. hierzu Abschnitt 2.1 und 4.2.4). So ruft etwa bei der *Push*-Eventkommunikation ein Objekt, das ein Ereignis einem anderen Objekt melden will, mit einem CORBA-Request eine *Push*-Methode auf diesem Objekt auf. Um diesen Mechanismus zu gewährleisten, werden für jedes SNMP-Modul, in dem Ereignismeldungen vorgesehen sind, genau zwei IDL-Schnittstellenbeschreibungen definiert, die jeweils *Push*- bzw. *Pull*-Methoden bereitstellen. Diese lauten:

- SnmpNotifications: *Push*-Methoden für die typisierte *push*-Eventkommunikation;
- PullSnmpNotifications: *Pull*-Methoden für die typisierte *pull*-Kommunikation.

Die *Push*- bzw. *Pull*-Methoden ergeben sich aus den NOTIFICATION-TYPE Makros eines MIB-Moduls, da mit diesem festgelegt wird, welche Managementinformation in einem SNMP-Trap enthalten ist.

Für jeden NOTIFICATION-TYPE Makro werden diese Schnittstellen um insgesamt drei Methoden erweitert und zwar:

- die Schnittstelle SnmpNotifications um die Funktion <notification_name>, wobei <notification_name> der Identifikator des NOTIFICATION-TYPE-Makros ist,
- die Schnittstelle PullSnmpNotifications um die beiden Funktionen try_<op> und pull_<op>. <op> ist der Name der korrespondierenden Operation in SnmpNotifications.

Die Parameter enthalten jeweils die OID und den Kontext der Quellinstanz sowie den Zeitpunkt des Traps. Folgendes Beispiel des NOTIFICATION-TYPE `linkUp` erläutert die Vorgangsweise genauer:

```
...

linkUp NOTIFICATION-TYPE
     OBJECTS { ifIndex }
     STATUS current
     DESCRIPTION
     ‘‘A linkUp trap signifies that the SNMPv2 entity ...’’
     ::= { snmpTraps 4 }
...
```

Damit stehen zunächst die Namen der Funktionen fest: linkUp für die Schnittstelle SnmpNotifications einerseits und pull_linkUp sowie try_linkUp für die Schnittstelle PullSnmpNotifications andererseits. Mit der Funktion try_linkUp kann abgefragt werden, ob bei einem Supplier eine solche Ereignismeldung vorliegt. Der Rückgabewert der Methode ist entweder *true* oder *false*. Die beiden anderen Methoden liefern keine Werte zurück.

In der optionalen OBJECTS-Klausel werden jene MIB-Variablen aufgelistet, deren Werte zusätzlich mit der Ereignismeldung verschickt werden. Im obigen Beispiel besteht diese Liste nur aus der MIB-Variable `ifIndex`, also dem Index des entsprechenden Zeileneintrags in der `Interfaces`-Tabelle. Die OBJECTS-Klausel wird auf eine IDL-Struktur abgebildet. Ihr Name ergibt sich aus dem Deskriptor des NOTIFICATION-TYPE-Makros, der um das Suffix Type ergänzt wird. Die Elemente entsprechen den MIB-Variablen in der OBJECTS-Liste; deren Datentyp wird aus der SYNTAX-Klausel des jeweiligen OBJECT-TYPE-Makros bestimmt. Zusätzlich werden der IDL-Struktur ein weiterer in-Parameter

der Methode linkUp sowie jeweils ein weiterer out-Parameter der Methoden pull_linkUp und try_linkUp hinzugefügt. Es sei darauf hingewiesen, daß bei der generischen Event-Kommunikation nur ein Parameter vom Typ *any* erlaubt ist. Dies war mitunter ein Grund, für SNMP-Traps typisierte Events vorzusehen. Dennoch sind im Modul SNMPMgmt für generische Eventkommunikation die Schnittstellen GenericNotification und PullGenericNotification mit den Methoden snmp_notification bzw. pull_snmp_notification und try_snmp_notification definiert.

Schließlich wird der OBJECT IDENTIFIER des NOTIFICATION-TYPE-Makros als konstante Zeichenkette übernommen.

Für obigen NOTIFICATION TYPE Makro erhält man somit das folgende IDL-Äquivalent :

```
struct IfIndexType {
    ASN1_ObjectIdentifier var_name;
    ASN1_ObjectIdentifier var_value;
};
struct LinkUpType {
    ASN1_Integer ifIndex;
};
const ASN1_ObjectIdentifier linkUp = “snmpTraps.4”;
/* DESCRIPTION:
    “A linkUp trap signifies that the SNMPv2 entity, ...” */

interface SnmpNotifications {
    void linkUp(
        in ASN1_ObjectIdentifier srcParty,
        in ASN1_ObjectIdentifier snmpContext,
        in SNMPv2TC::TimeStampType eventTime,
        in LinkUpType notification_info);
};
interface PullSnmpNotifications {
    void pull_linkUp(
        out ASN1_ObjectIdentifier srcParty,
        out ASN1_ObjectIdentifier snmpContext,
        out SNMPv2TC::TimeStampType eventTime,
        out LinkUpType notification_info);
    boolean try_linkUp(
        out ASN1_ObjectIdentifier srcParty,
        out ASN1_ObjectIdentifier snmpContext,
        out SNMPv2TC::TimeStampType eventTime,
        out LinkUpType notification_info);
};
```

Die IDL-Schnittstellen SnmpNotifications und PullSnmpNotifications werden für jedes Modul nur einmal definiert. Falls mehrere NOTIFICATION-TYPE-Makros in einem MIB-Modul vorhanden sind, so werden diese Schnittstellen um entsprechende Methoden ergänzt.

ABKÜRZUNGEN

ACID	Atomicity, Consistency, Isolation, Durability
ACSE	Association Control Service Element
AIX	Advanced Interactive Executive
API	Application Programming Interface
ASE	Application Service Element
ASN.1	Abstract Syntax Notation Number One
BOA	Basic Object Adapter
CASE	Computer Aided Software Engineering
CCF	Connection Control Function
CCITT	International Telegraph and Telephone Consultative Committee (jetzt ITU)
CFA	Common Facilities Architecture
CI	Component Interface
CICS	Customer Information Control System
CIM	Common Information Model
CIMOM	CIM Object Manager
CLEC	Competitive Local Exchange Carrier
CMF	Common Management Facilities
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Element
CMOT	Common Management Information Protocol over TCP/IP
CNM	Customer Network Management
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
DCE	Distributed Computing Environment

DCM	Device Communication Manager
DCOM	Distributed Common Object Model
DFN	Deutsches Forschungsnetz
DII	Dynamic Invocation Interface
DLL	Dynamic Link Library
DME	Distributed Management Environment
DMI	Definition of Management Information
DMI	Desktop Management Interface
DMTF	Desktop Management Task Force
DN	Distinguished Name
DNS	Domain Name System
DPE	Distributed Processing Environment
DPI	Distributed Protocol Interface
DSI	Dynamic Skeleton Interface
DSOM	Distributed System Object Model
DTF	Domain Task Force
DV	Datenverarbeitung
EFD	Event Forwarding Discriminator
EMS	Event Management Services
ENC	European Networking Center
ESIOP	Environment-specific Inter-ORB Protocol
ESPRIT	European Specific Research and Technological Development Programme in the Field of Information Technology
EUI	End User Interface
EURESCOM	European Institute for Research and Strategic Studies in Telecommunications
GDMO	Guidelines for the Definition of Managed Objects
GEMA	Generic Management Architecture
GEO	Geosynchronous Earth Orbit
GIOP	General Inter-ORB Protocol
GMOC	Generic Managed Object Classes
GRM	General Relationship Model
GTM	Generalized Topology Manager
GUI	Graphical User Interface
HMMP	Hypermedia Management Protocol
HMMS	Hypertext Management Scheme
HP	Hewlett Packard

HTTP	Hypertext Transfer Protocol
IAB	Internet Architecture Board
IBM	International Business Machines Corporation
ICMP	Internet Control Message Protocol
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IIMC	ISO-Internet Management Coexistence
IIOP	Internet Inter-ORB Protocol
ILEC	Incumbent Local Exchange Carrier
IN	Intelligent Network
IOP	Inter-ORB-Protocol
IOR	Interoperable Object Reference
IP	Internet Protocol
IR	Interface Repository
ISO/IEC	International Organization for Standardization International Electrotechnical Committee
ISP	Internet Service Provider
IT	Information Technology
ITU	International Telecommunications Union
IV	Informationsverarbeitung
JDMK	Java Dynamic Management Kit
JIDM	Joint Inter-Domain Management
JMAPI	Java Management Application Programming Interface
JNI	Java Native Interface
KI	Künstliche Intelligenz
LAN	Local Area Network
LCF	Log Control Function
LEC	Local Exchange Carrier
LEO	Low Earth Orbit
M2M	Manager-to-Manager
MAN	Metropolitan Area Network
MAScOTTE	Management Services for Object-oriented Distributed Systems
MASIF	Mobile Agent System Interoperability Facilities Specification
MbD	Management by Delegation
MEO	Middle Earth Orbit
MF	Mediation Function
MHS	Message Handling Systems

MI	Management Interface
MIB	Management Information Base
MIF	Management Information Format
MIT	Management Information Tree
MLM	Mid-level Manager
MO	Managed Object
MOC	Managed Object Class
MOF	Managed Object Format
MOI	Managed Object Instance
MTA	Message Transfer Agent
MVS	Multiple Virtual Storage
NEF	Network Element Function
NFS	Network File System
NIS	Network Information System
NM	Netzmanagement
NMF	Network Management Forum
NRIM	Network Resource Information Model
NSM	Network Services Monitoring
OAM	Operation, Adminstration, and Maintenance
ODM	Object Data Manager
ODMA	Open Distributed Management Architecture
ODP	Open Distributed Processing
OID	Object Identifier
OMA	Object Management Architecture
OMF	Object Management Function
OMG	Object Management Group
OMNIPoint	Open Management Interoperability Points
OMT	Object Modeling Technique
OOA	Object-Oriented Analysis
OOD	Object-Oriented Design
ORB	Object Request Broker
ORS	Object Registration Services
OS	Operations Systems
OSA	Object Services Architecture
OSF	Open Software Foundation
OSF	Operations Systems Function
OSI	Open Systems Interconnection

PDU	Protocol Data Unit
QAF	Q-Adapter Function
QoS	Quality of Service
RDN	Relative Distinguished Name
RFC	Request For Comments
RFP	Request For Proposal
RMON	Remote Network Monitoring
RM-ODP	Reference Model of Open Distributed Processing
ROSE	Remote Operations Service Element
RPC	Remote Procedure Call
SAP	Service Access Point
SCF	Service Control Function
SDF	Service Data Function
SIA	System Information Agent
SIG	Special Interest Group
SIOP	SS.7 Inter-ORB-Protocol
SM	Systemmanagement
SMAE	Systems Management Application Entity
SMAP	Systems Management Application Process
SMASE	Systems Management Application Service Elements
SMF	Service Management Function
SMFs	Systems Management Functions
SMFAs	Specific Management Functional Areas
SMI	Structure of Management Information
SMIT	System Management Information Tool
SNMP	Simple Network Management Protocol
SOM	System Object Model
SP	Service Provider
SQL	Structured Query Language
SSF	Service Switching Function
SSP	Service Switching Point
SS.7	Signalling System Number 7
StP	Software through Pictures
SW	Software
TCP/IP	Transmission Control Protocol / Internet Protocol
TINA	Telecommunication Information Networking Architecture
TK	Telekommunikation

TME	Tivoli Management Environment
TMN	Telecommunications Management Network
UML	Unified Modeling Language
UUID	Universal Unique Identifier
VNM	Virtual Network Machine
WAN	Wide Area Network
WBEM	Web-based Enterprise Management
WSF	Work Station Function
WWW	World Wide Web
XCMF	X/Open Common Management Facilities
XML	Extensible Markup Language
XMP	X/Open Management Protocol
XOM	X/Open OSI-Abstract-Data Manipulation API
XoJIDM	Joint X/Open-NM Forum Inter-Domain Management
XoTGSysMan	X/Open Systems Management Working Group

ABBILDUNGEN

TABELLEN

LITERATUR

[AbCH 93] ABECK, SEBASTIAN, ALEXANDER CLEMM und ULF HOLLBERG: *Simply Open Network Management: An approach for the Integration of SNMP into OSI Management Concepts.* In: HEGERING, HEINZ-GERD und YECHIAM YEMINI [INM-III 93], Seiten 361–375.

[ADKL 97] APOSTOLESCU, VICTOR, GABI DREO-RODOSEK, THOMAS KAISER, MICHAEL LANGER, STEFAN LOIDL und MICHAEL NERB: *Einführung eines Customer Network Managements für das B-WIN*. In: *27. Betriebstagung des DFN-Vereins, Berlin*, Oktober 1997.

[AiPl 94] AIDAROUS, SALAH und THOMAS PLEVYAK (Herausgeber): *Telecommunications Network Management into the 21st Century: Techniques, Standards, Technologies and Applications*. IEEE Press, 1994.

[AiPl 98] AIDAROUS, SALAH und THOMAS PLEVYAK (Herausgeber): *Telecommunications Network Management: Technologies and Implementations*. IEEE Press, 1998.

[AMS2.0] *Application Management Specification (AMS), Version 2.0*. Technischer Bericht IBM Corporation, November 1997.

[Bapa 98] BAPAT, SUBOTH: *Java in Network Management*. In: SAHIN, VELI [NOMS 98].

[Beie 93] BEIER, B.: *Die Integration von SNMP-Management in OSI-Netzmanagement*. Diplomarbeit, Universität Karlsruhe, 1993.

[BlSt 97] BLAIR, GORDON S. und JEAN-BERNARD STEFANI: *Open Distributed Processing and Multimedia*. Addison-Wesley, 1997.

[CCITT Q.700] *Introduction to CCITT Signalling System No. 7*. Recommendation Q.700, CCITT, März 1993.

[CCSH 97] CHATT, TOM R., MICHAEL A. CURRY, JUHA SEPPÄ und ULF HOLLBERG: *TMN/C++: An object-oriented API for GDMO, CMIS, and ASN.1*. In: LAZAR, AUREL A. et al. [INM-V 97], Seiten 177–191.

[ChKo 97] CHEN, G. und Q. KONG: *Integrated TMN service provisioning and management environment*. In: LAZAR, AUREL A. et al. [INM-V 97], Seiten 99–112.

[Cohe 94] COHEN, ROBERTA: *The Telecommunications Management Network (TMN)*, Kapitel 9, Seiten 217–242. In: SLOMAN, MORRIS [Slom 94], Juni 1994.

[CORBA 2.2] *The Common Object Request Broker: Architecture and Specification*. OMG Specification Revision 2.2, Object Management Group, Februar 1998.

[Cros 97] CROSS, TERENCE: *ORB Instrumentation*. OMG CORBA Management Workshop, Dublin, September 1997.

[DiBo 97] DINI, P. und R. BOUTABA: *Deriving Variable Pollig Frequency Policies for Pro-active Management in Networks and Distributed Systems*. In: LAZAR, AUREL A. et al. [INM-V 97], Seiten 541–552.

[DivB 97] DINI, P. und G. VON BOCHMANN: *Agent based Management of Distributed Systems with Variable Polling Frequency Profiles*. In: LAZAR, AUREL A. et al. [INM-V 97], Seiten 553–564.

[DMTF 96] DESKTOP MANAGEMENT TASK FORCE: *Desktop Management Interface Specification (Version 2.0)*, März 1996.

[DMTF 98] *Common Information Model (CIM) Version 2.0*. Specification, März 1998.

[DNWI 95] DREO, G., B. NEUMAIR, R. WIES, A. BÖTTCHER und M. WERNER: *Management von LEO/MEO-Satellitennetzen: Anforderungen und Netzdarstellung*. In: FRANKE, K., U. HÜBNER und W. KALFA (Herausgeber): *Kommunikation in verteilten Systemen, GI/ITG-Fachtagung*, Seiten 270–284. Springer-Verlag, Februar 1995.

[Domb 97] DOMBACH, T.: *Erstellung und Anwendung eines Kriterienkataloges zur Klassifikation von Managementplattformen*. Diplomarbeit, Ludwig-Maximilians-Universität München, Juni 1997.

[DSOM 96] SARACCO, ROBERTO (Herausgeber): *Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*, Scuola Superiore G. Reiss Romoli, L'Aquila, Italy, Oktober 1996. IFIP.

[Eust 96] EUBA, A. und D. STRICKER: *Implementierung eines Werkzeuges zur graphischen Darstellung von WWW-Link-Strukturen*. Fortgeschrittenenpraktikum, Technische Universität München, Juli 1996.

[FeHN 96] FERIDUN, M., L. HEUSLER und R. NIELSEN: *Implementing OSI Agents for TMN*. IBM Research Report RZ 2759, IBM Research Division, Zurich Research Laboratory, 1995.

[FKWW 95] FUENTE, L.A. DE LA, M. KAWANISHI, M. WAKANO, T. WALLES und C. AURRECOECHEA: *Application of the TINA-C Management Architecture*. In: RAYNAUD, YVES et al. [INM-IV 95], Seiten 424–435.

[GeDi 98] GERVAIS, MARIE-PIERRE und ALIOUNE DIAGNE: *Enhancing Telecommunications Service Engineering with Mobile Agent Technology and Formal Methods*. IEEE Communications Magazine, 36(7):38–43, Juli 1998.

[GlHa 96] GLITHO, RICH H. und STEPHEN HAYES: *Approaches for Introducing TMN in Legacy Networks: A Critical Look.* IEEE Communications Magazine, 34(9), September 1996.

[Gold 96] GOLDSZMIDT, GERMAN: *Distributed Management by Delegation*. Dissertation, Columbia University, 1996.

[Gome 98] CID, GONZALO GOMEZ: *CORBA in the Telco Environment: The Case at Telefonica I & D*. orbix Journal, (2), April 1998.

[GoYe 95] GOLDSZMIDT, GERMAN und YECHIAM YEMINI: *Distributed Management by Delegation*. In: *Proceedings of the 15th International Conference on Distributed Computing Systems*, Juni 1995.

[GoYe 98] GOLDSZMIDT, GERMAN und YECHIAM YEMINI: *Delegated Agents for Network Management*. IEEE Communications Magazine, 36(3):66–70, März 1998.

[GrBy 98] GREENBERG, MICHAEL S. und JENNIFER C. BYINGTON: *Mobile Agents and Security*. IEEE Communications Magazine, 36(7):76–85, Juli 1998.

[GrPa 97] GRIFFIN, DAVID und GEORGE PAVLOU: *Realizing TMN-like Management Services in TINA.* Journal of Network and Systems Management; Special Issue on TINA, 5(4), Dezember 1997.

[GuNe 95] GUTSCHMIDT, M. und B. NEUMAIR: *Integration von Netz- und Systemmanagement: Ziele und erste Erfahrungen*. In: *Proceedings der 3. Fachtagung Arbeitsplatzrechensysteme (APS'95), Hannover*, Mai 1995.

[Guts 95] GUTSCHMIDT, M.: *Ein Objektmodell für ein integriertes Management von Systemdiensten mit Client/Server-Struktur*. Dissertation, Ludwig-Maximilians-Universität München, September 1995.

[HAB 91] HEGERING, H.-G., S. ABECK und TH. BÖHNKE: *Converting MIB-Descriptions into MIB-Implementations*. In: *Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*, Santa Barbara, CA, USA, Oktober 1991. .

[Hama 97] HAMADA, TAKEO: *An overview of the TINA Architecture*. Journal of Network and Systems Management; Special Issue on TINA, 5(4), Dezember 1997.

[Harr 97] HARRINGTON, DAVID: *The Evolution of Architectural Concepts in the SNMPv3 Working Group*. The Simple Times, 5(1):1–7, Dezember 1997.

[HeAb 93] HEGERING, HEINZ-GERD und SEBASTIAN ABECK: *Integriertes Netz– und Systemmanagement*. Addison-Wesley, 1993.

[HeAN 99] HEGERING, HEINZ-GERD, SEBASTIAN ABECK und BERNHARD NEUMAIR: *Integriertes Management vernetzter Systeme — Konzepte, Architekturen und deren betrieblicher Einsatz*. dpunkt-Verlag, 1999.

[Hoel 96] HÖLLER, T.: *Konzeption und Realisierung eines CORBA / SNMP Gateways*. Diplomarbeit, Technische Universität München, August 1996.

[Hoel 97] HÖLSCHER, G.: *Entwurf und CORBA-basierte Implementierung eines Objektmodells für das Management von ATM-Switches*. Diplomarbeit, Technische Universität München, November 1997.

[Hump 95] HUMPHREY, WATTS S.: *A Discipline for Software Engineering*. Addison-Wesley, 1995.

[IBMNVAR 95] IBM CORPORATION, Research Triangle Park, NC 27709-12195: *IBM NetView for AIX Version 4: Administrator's Reference*, Erste Auflage, Juli 1995. Order Number: SC31-8169-00.

[IBMNVPR 95] IBM CORPORATION, Research Triangle Park, NC 27709-12195: *IBM NetView for AIX Version 4: Programmer's Reference*, Erste Auflage, Juli 1995. Order Number: SC31-8165-00.

[IBMSOM94] IBM CORPORATION, INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION, Research Triangle Park, NC 27709-2195: *SOMobjects: A Practical Introduction to SOM and DSOM*, Juli 1994. Order Number: GG24-4357-00.

[IBMSVSz95] IBM CORPORATION, INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION, Research Triangle Park, NC 27709-2195: *IBM System View for AIX: Sizing Considerations*, Mai 1995. Order Number: GG24-2586-00.

[IBMSysMon94] IBM CORPORATION, INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION, Research Triangle Park, NC 27709-2195: *IBM Systems Monitor: Anatomy of a Smart Agent*, Dezember 1994. Order Number: GG24-4398-00.

[IBMTMNGI 96] IBM CORPORATION, Research Triangle Park, NC 27709-2195: *IBM TMN Products for AIX Release 2: General Information*, Release 2 Auflage, März 1996. Order Number: GC31-8016-01.

[IBMTMNSF 96] IBM CORPORATION, Research Triangle Park, NC 27709-2195: *IBM NetView TMN Support Facility for AIX Release 2: User's Guide*, März 1996. Order Number: SC31-8017-01.

[IBMTMNWB 96] IBM CORPORATION, Research Triangle Park, NC 27709-2195: *IBM TMN WorkBench for AIX Release 2: Managed Object/Agent Composer User's and Programmer's Guide*, März 1996. Order Number: SC31-8006-01.

[ICODP 97] ROLIA, JEROME, JACOB SLONIM und JOHN BOTSFORD (Herausgeber): *Proceedings of the IFIP/IEEE International Conference on Open Distributed Processing and Distributed Platforms*. IFIP, Chapman and Hall, Mai 1997.

[IITB 96] USLÄNDER, THOMAS ET AL.: *The CORBA-Assistant*. White Paper, Fraunhofer-IITB, Oktober 1996.

[INM-II 91] KRISHNAN, I. und W. ZIMMER (Herausgeber): *Proceedings of the 2nd International Symposium on Integrated Network Management*, Washington D.C., USA, April 1991. IFIP, North-Holland.

[INM-III 93] HEGERING, HEINZ-GERD und YECHIAM YEMINI (Herausgeber): *Proceedings of the 3rd International Symposium on Integrated Network Management*, San Francisco, CA, USA, April 1993. IFIP, North-Holland.

[INM-IV 95] RAYNAUD, YVES, ADARSHPAL SETHI und FABIENNE FAURE-VINCENT (Herausgeber): *Proceedings of the 4th International Symposium on Integrated Network Management*, Santa Barbara, CA, USA, Mai 1995. IFIP, Chapman and Hall.

[INM-V 97] LAZAR, AUREL A., ROBERTO SARACCO und ROLF STADLER (Herausgeber): *Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management*, San Diego, CA, USA, Mai 1997. IFIP, Chapman and Hall.

[IONA 97] IONA TECHNOLOGIES PLC.: *OrbixManager*. White Paper, IONA Technologies PLC., Mai 1997.

[IsNi 98] ISHIKAWA, HIROSHI und KEN ICHI NISHIMURA: *Impact and Preliminary Results of Telecommunications Deregulation in Japan*. IEEE Communications Magazine, 36(7):100–104, Juli 1998.

[ISO 10040] *Information Technology – Open Systems Interconnection – Systems Management Overview*. IS 10040, International Organization for Standardization and International Electrotechnical Committee, 1992.

[ISO 10164-11] *Information Technology – Open Systems Interconnection – Systems Management – Part 11: Metric Objects and Attributes*. IS 10164-11, International Organization for Standardization and International Electrotechnical Committee, 1994.

[ISO 10164-13] *Information Technology – Open Systems Interconnection – Systems Management – Part 13: Summarization Function*. IS 10164-13, International Organization for Standardization and International Electrotechnical Committee, 1995.

[ISO 10164-16] *Information Technology – Open Systems Interconnection – Systems Management – Part 16: Management Knowledge Management Function*. IS 10164-16, International Organization for Standardization and International Electrotechnical Committee, Juli 1997.

[ISO 10164-2] *Information Technology – Open Systems Interconnection – Systems Management – Part 2: State Management Function*. IS 10164-2, International Organization for Standardization and International Electrotechnical Committee, 1993.

[ISO 10164-5] *Information Technology – Open Systems Interconnection – Systems Management – Part 5: Event Report Management Function*. IS 10164-5, International Organization for Standardization and International Electrotechnical Committee, 1993.

[ISO 10164-6] *Information Technology – Open Systems Interconnection – Systems Management – Part 6: Log Control Function*. IS 10164-6, International Organization for Standardization and International Electrotechnical Committee, 1993.

[ISO 10164-x] *Information Technology – Open Systems Interconnection – Systems Management – Management Functions*. IS 10164-x, International Organization for Standardization and International Electrotechnical Committee, 1991-97.

[ISO 10165-1] *Information Technology – Open Systems Interconnection – Structure of Management Information – Part 1: Management Information Model*. IS 10165-1, International Organization for Standardization and International Electrotechnical Committee, 1993.

[ISO 10165-4] *Information Technology – Open Systems Interconnection – Structure of Management Information – Part 4: Guidelines for the Definition of Managed Objects*. IS 10165-4, International Organization for Standardization and International Electrotechnical Committee, 1992.

[ISO 10165-5] *Information Technology – Open Systems Interconnection – Structure of Management Information – Part 5: Generic Management Information*. IS 10165-5, International Organization for Standardization and International Electrotechnical Committee, 1994.

[ISO 10165-6] *Information Technology – Open Systems Interconnection – Structure of Management Information – Part 6: Requirements and Guidelines for Implementation Conformance Statement Proformas Associated with OSI Management*. IS 10165-6, International Organization for Standardization and International Electrotechnical Committee, 1997.

[ISO 10165-9] *Information Technology – Open Systems Interconnection – Structure of Management Information – Part 9: Systems Management Protocol Machine Objects*. WD 10165-9, International Organization for Standardization and International Electrotechnical Committee, Mai 1996.

[ISO 10746-1] *Open Distributed Processing – Reference Model – Part 1: Overview*. DIS 10746-1, International Organization for Standardization and International Electrotechnical Committee, 1995.

[ISO 10746-2] *Open Distributed Processing – Reference Model – Part 2: Foundations*. IS 10746-2, International Organization for Standardization and International Electrotechnical Committee, 1995.

[ISO 10746-3] *Open Distributed Processing – Reference Model – Part 3: Architecture*. IS 10746-3, International Organization for Standardization and International Electrotechnical Committee, 1995.

[ISO 10746-4] *Open Distributed Processing – Reference Model – Part 4: Architectural Semantics*. DIS 10746-4, International Organization for Standardization and International Electrotechnical Committee, 1995.

[ISO 10746] *Open Distributed Processing – Reference Model*. IS 10746, International Organization for Standardization and International Electrotechnical Committee, 1995.

[ISO 13235] *Open Distributed Processing – ODP Trading Function.* DIS 13235, International Organization for Standardization and International Electrotechnical Committee, Juni 1995.

[ISO 9595] *Information Technology – Open Systems Interconnection – Common Management Information Service Definition*. IS 9595, International

Organization for Standardization and International Electrotechnical Committee, 1991.

[ISO 9596-1] *Information Technology – Open Systems Interconnection – Common Management Information Protocol – Part 1: Specification.* IS 9596-1, International Organization for Standardization and International Electrotechnical Committee, 1991.

[ITOcg 97] *HP OpenView IT/Operations Concepts Guide*. User Manual, Hewlett Packard, August 1997. Order Number: B4249-90011.

[ITU M.3010] *Principles for a Telecommunications Management Network.* Recommendation M.3010, ITU, 1996.

[ITU M.3100] *Generic Network Information Model*. Recommendation M.3100, ITU, 1995.

[ITU X.703] *Open Distributed Management Architecture (ODMA).* Recommendation X.703, ITU, Oktober 1997.

[ITU X.920] *Open Distributed Processing – Interface Definition Language*. Draft Recommendation X.920, ITU, November 1997.

[IWSM-I 93] CHU, WESLEY W. und ALLAN FINKEL (Herausgeber): *Proceedings of the IEEE First International Workshop On Systems Management, Los Angeles*. IEEE, April 1993.

[JIDM 97] *Inter-Domain Management: Specification Translation.* Open Group Preliminary Specification P509, Open Group, März 1997.

[KaSe 93] KALYANASUNDARAM, PRAMOD und ADARSHPAL SETHI: *An Application Gateway Design for OSI-Internet Management*. In: HEGERING, HEINZ-GERD und YECHIAM YEMINI [INM-III 93].

[KaSe 94] KALYANASUNDARAM, PRAMOD und ADARSHPAL SETHI: *Interoperability Issues in Heterogeneous Network Management*. Journal of Network and Systems Management, 2(2):169 – 193, Juni 1994.

[Kell 93] KELLER, A.: *Entwurf und Implementierung von Managementszenarien zu bestehenden Kommunikationsanwendungen*. Diplomarbeit, Technische Universität München, Mai 1993.

[Kemp 98] KEMPTER, B.: *Entwurf eines Java/CORBA-basierten Mobilen Agenten.* Diplomarbeit, Technische Universität München, August 1998.

[KeNe 95] KELLER, A. und B. NEUMAIR: *Systems Management Middleware: Verteilte objektorientierte Technologien für das Systemmanagement*.

In: WALL, DIETER (Herausgeber): *Organisation und Betrieb von DV-Versorgungsstrukturen*, Seiten 83–97, Göttingen, Germany, November 1995. Deutscher Universitäts-Verlag.

[KeNe 97a] KELLER, A. und B. NEUMAIR: *Interoperable Architekturen als Basis eines integrierten Managements*. In: ZITTERBART, M. [KiVS 97], Seiten 405–418.

[KeNe 97c] KELLER, A. und B. NEUMAIR: *Using ODP as a Framework for CORBA-based Distributed Applications Management*. In: ROLIA, JEROME et al. [ICODP 97], Seiten 110–121.

[KiVS 97] ZITTERBART, M. (Herausgeber): *Kommunikation in Verteilten Systemen*, Braunschweig, Germany, Februar 1997. GI/ITG–Fachtagung, Springer Verlag.

[KiVST 95] GI/ITG–FACHTAGUNG: *Tutoriumsband: Kommunikation in verteilten Systemen*, Februar 1995.

[Knoe 99] KNÖCHLEIN, H.: *Management eines Internet Telefonie Servers mittels JDMK* . Diplomarbeit, Technische Universität München, Februar 1999.

[Krie 94] KRIEGER, U.: *Konzeption einer Managementinformationsbasis für das Management von UNIX-Endsystemen*. Diplomarbeit, Technische Universität München, Mai 1994.

[LaLN 98] LANGER, MICHAEL, STEFAN LOIDL und MICHAEL NERB: *Customer Service Management: A more transparent view to your subscribed services*. In: *Proceedings of the 8th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 98)*, Newark, DE, USA, Oktober 1998. .

[Lang 96] LANGER, M.: *Entwurf und Implementierung eines CMIP/SNMP Gateways*. Diplomarbeit, Technische Universität München, November 1996.

[LLLM 96] LECLERC, M., C. LINNHOFF-POPIEN, S. LIPPERTS, T. MÜSSENER und H. WEGMANN: *CORBA-based Data Transfer for Financial Risk Management*. In: SPANIOL, OTTO, CLAUDIA LINNHOFF-POPIEN und BERND MEYER (Herausgeber): *Proceedings of the International Workshop Trends in Distributed Systems: CORBA and Beyond (TreDS'96)*, Seiten 136–147, Aachen, Germany, Oktober 1996. .

[MaPo 96] MAGEDANZ, THOMAS und RADU POPESCU-ZELETIN: *Intelligent Networks: Basic Technology, Standards and Evolution*. International Thompson Computer Press, Zweite Auflage, 1996.

[Masc 97] MAScOTTE PARTNERS: *Introduction to MAScOTTE*. White Paper, MAScOTTE Consortium, Mai 1997.

[Mazu 98a] MAZUMDAR, SUBRATA: *Application of Distributed Object Technology in Network Management Domain*. In: SAHIN, VELI [NOMS 98].

[McPo 98] MCKENNA, ROBERT B. und DAN L. POOLE: *Data Communications: Where Regulators clash with Reality*. IEEE Communications Magazine, 36(7):96–99, Juli 1998.

[MEBS 95] MEYER, K., M. ERLINGER, J. BETSER, C. SUNSHINE, G. GOLDSZMIDT und Y. YEMINI: *Decentralizing Control and Intelligence in Network Management*. In: RAYNAUD, YVES et al. [INM-IV 95].

[Mill 98] MILLER, BARRY: *Satellites Free the Mobile Phone*. IEEE Spectrum, März 1998.

[Moun 97] MOUNTZIA, M.-A.: *Flexible Agents in Integrated Network and Systems Management*. Dissertation, Technische Universität München, Dezember 1997.

[Muel 98] MÜLLER, T.: *CORBA-basiertes Management von UNIX-Workstations mit Hilfe von ODP-Konzepten*. Diplomarbeit, Technische Universität München, Februar 1998.

[Murr 93] MURRILL, BRUCE: *OMNIPoint: An Implementation Guide to Integrated Networked Information Systems Management*. In: HEGERING, HEINZ-GERD und YECHIAM YEMINI [INM-III 93], Seiten 405–418.

[Nata 98] NATARAJAN, N.: *TINA Network Resource Information Model*. Journal of Network and Systems Management; Special Issue on Management Information Model Engineering, 6(3), September 1998.

[Neum 98] NEUMAIR, B.: *Distributed Applications Management based on ODP Viewpoint Concepts and CORBA*. In: SAHIN, VELI [NOMS 98], Seiten 559–569.

[Neum 99] NEUMAIR, BERNHARD: *Managementmodelle für Systemdienste und Anwendungen: Anforderungen und Realisierung*. Habilitationsschrift, Ludwig-Maximilians-Universität, 1999. in Vorbereitung.

[NMF 114] NETWORK MANAGEMENT FORUM: *OMNIPoint Integration Architecture*. FORUM Technical Report TR114, 1995.

[NMF 26] NETWORK MANAGEMENT FORUM: *Translation of Internet MIBs to OSI/CCITT GDMO MIBs*. NMF 026, 1993.

[NMF 28] NETWORK MANAGEMENT FORUM: *ISO/CCITT to Internet Management Proxy*. NMF 028, 1993.

[NMF 29] NETWORK MANAGEMENT FORUM: *Translation of Internet MIB-II (RFC 1213) to ISO/CCITT GDMO MIB.* NMF 029, 1993.

[NMF 93] NETWORK MANAGEMENT FORUM (Herausgeber): *Discovering OMNIPoint - A Common Approach to the Integrated Management of Networked Information Systems*. Prentice Hall, 1993.

[NMF 98] *TMN/C++ Application Programming Interface*. TeleManagement Forum Specification 043, Network Management Forum, März 1998.

[NOMS 98] SAHIN, VELI (Herausgeber): *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, Band 1–3, New Orleans Marriott, New Orleans, LA, USA, Februar 1998. IEEE Communications Society, IEEE Press.

[OMG 95-1-2] *Common Facilities Architecture*. OMG TC Document 95-1-2, Revision 4.0, Object Management Group, Januar 1995.

[OMG 95-1-47] *Object Services Architecture*. OMG TC Document 95-1-47, Revision 8.1, Object Management Group, Januar 1995.

[OMG MAF] *Mobile Agent System Interoperability Facilities Specification*. OMG TC Document orbos/97-10-05, Object Management Group, November 1997.

[OMG SNMP] JIDM WORKING GROUP: *CORBA/TMN Interworking - SNMP Part*. TC Document telecom/98-05-03, Object Management Group, Mai 1998.

[OMGCF 98] *CORBA Facilities Architecture Specification*. OMG Specification formal/98-07-10, Object Management Group, Juli 1998.

[OMGSS 97] *CORBAservices: Common Object Services Specification*. OMG Specification, Object Management Group, November 1997.

[OMGTC 98] *CORBA Telecom Specification*. OMG Specification formal/98-07-12, Object Management Group, Juni 1998.

[OrHa 98] ORFALI, ROBERT und DAN HARKEY: *Client/Server Programming with Java and CORBA*. John Wiley & Sons, Inc., Zweite Auflage, 1998.

[OrHE 95] ORFALI, ROBERT, DAN HARKEY und JERI EDWARDS: *The Essential Client/Server Survival Guide*. Van Nostrand Reinhold, 1995.

[OSMsg 96] *Messaging RFP*. OMG Request for Proposal orbos/96-03-16, Object Management Group, März 1996.

[P508 97] BOUJEMAA, FAYCAL ET AL.: *Introduction of Distributed Computing Middleware in Intelligent Networks*. White Paper, EURESCOM Project P508: Evolution, Migration Paths and Interworking to TINA, September 1997.

[P508-D1] EURESCOM PARTICIPANTS IN PROJECT P508: *Initial Assessment of the options for evolving to TINA*. Deliverable 1, EURESCOM Project P508: Evolution, Migration Paths and Interworking to TINA, Februar 1996.

[P508-D2] EURESCOM PARTICIPANTS IN PROJECT P508: *Migration Strategy and Interworking with Legacy Systems*. Deliverable 2, EURESCOM Project P508: Evolution, Migration Paths and Interworking to TINA, April 1997.

[Pavl 93] PAVLOU, GEORGE: *The OSIMIS TMN Platform: Support for Multiple Technology Integrated Management Systems*. In: *Proceedings of the 1st RACE IS&N Conference*, Paris, November 1993. .

[PeMc 97] PERKINS, DAVID und EVAN MCGINNIS: *Understanding SNMP MIBs*. Prentice-Hall, 1997.

[PhKa 98] PHAM, VU ANH und AHMED KARMOUCH: *Mobile Software Agents: An Overview*. IEEE Communications Magazine, 36(7):26–37, Juli 1998.

[PoCh 96] POO, GEE-SWEE und CHYE-GUAN CHEW: *Modeling of the XOM/XMP Application Programming Interface (API)*. IEEE Communications Magazine, 34(8), August 1996.

[Post 94] POSTON, ROBERT M.: *Automated from Object Models*. Communications of the ACM, September 1994.

[Proz 97] PROZELLER, PAUL E.: *TINA and the Software Infrastructure of the Telecom Network of the Future*. Journal of Network and Systems Management; Special Issue on TINA, 5(4), Dezember 1997.

[PTBH 98] PAVON, JUAN, JOSE TOMAS, YVES BARDOUT und LINDA-HELENE HAUW: *CORBA for Network and Service Management in the TINA Framework*. IEEE Communications Magazine, 36(3):72–79, März 1998.

[Rama 98] RAMAN, LAKSHMI: *OSI Systems and Network Management*. IEEE Communications Magazine, 36(3):46–53, März 1998.

[RBPE 91] RUMBAUGH, JAMES, MICHAEL BLAHA, WILLIAM PREMERLANI, FREDERICK EDDY und WILLIAM LORENSEN: *Object-Oriented Modeling and Design.* Prentice-Hall International, Inc., 1991.

[Reil 93] REILLY, J.: *VTT IIMC Notes - Modification to SMIC SNMP MIB Compiler to produce GDMO MIB Definitons*. Technischer Bericht Technical Research Centre of Finland, Telecommunications Laboratory (VTT/TEL), Mai 1993.

[rfc1155] ROSE, M. T. und K. MCCLOGHRIE: *RFC 1155: Structure and identification of management information for TCP/IP-based internets*. RFC, IETF, Mai 1990.

[rfc1213] MCCLOGHRIE, K. und M. T. ROSE: *RFC 1213: Management Information Base for Network Management of TCP/IP-based internets:MIB-II*. RFC, IETF, März 1991.

[rfc1451] CASE, J., K. MCCLOGHRIE, M. ROSE und S. WALDBUSSER: *RFC 1451: Manager-to-Manager Management Information Base*. RFC, IETF, April 1993.

[rfc1514] GRILLO, P. und S. WALDBUSSER: *RFC 1514: Host Resources MIB*. RFC, IETF, September 1993.

[rfc1565] KILLE, S. und N. FREED: *RFC 1565: Network Services Monitoring MIB*. RFC, IETF, Januar 1994.

[rfc1592] WIJNEN, B., G. CARPENTER, K. CURRAN, A. SEHGAL und G. WATERS: *RFC 1592: Simple Network Management Protocol Distributed Protocol Interface Version 2.0*. RFC, IETF, März 1994.

[rfc1902] SNMPV2 WORKING GROUP, J. CASE, K. MCCLOGHRIE, M. ROSE und S. WALDBUSSER: *RFC 1902: Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)*. RFC, IETF, Januar 1996.

[rfc1905] SNMPV2 WORKING GROUP, J. CASE, K. MCCLOGHRIE, M. ROSE und S. WALDBUSSER: *RFC 1905: Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)*. RFC, IETF, Januar 1996.

[rfc1907] SNMPV2 WORKING GROUP, J. CASE, K. MCCLOGHRIE, M. ROSE und S. WALDBUSSER: *RFC 1907: Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)*. RFC, IETF, Januar 1996.

[rfc1908] SNMPv2 WORKING GROUP, J. CASE, K. MCCLOGHRIE, M. ROSE und S. WALDBUSSER: *RFC 1908: Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework*. RFC, IETF, Januar 1996.

[rfc1909] MCCLOGHRIE, K.: *RFC 1909: An Administrative Infrastructure for SNMPv2*. RFC, IETF, Februar 1996.

[rfc2257] DANIELE, M., B. WIJNEN und D. FRANCISCO: *RFC 2257: Agent Extensibility (AgentX) Protocol Version 1*. RFC, IETF, Januar 1998.

[rfc2287] KRUPCZAK, C. und J. SAPERIA: *RFC 2287: Definitions of System-Level Managed Objects for Applications*. RFC, IETF, Februar 1998.

[Rose 94] ROSE, MARSHALL T.: *The Simple Book — An Introduction to Internet Management*. Prentice-Hall, Zweite Auflage, 1994.

[Rose 96] ROSE, MARSHALL T.: *The Simple Book — An Introduction to Networking Management*. Prentice-Hall, Zweite Auflage, 1996.

[Sahi 94] SAHIN, VELI: *Telecommunications Management Network: Principles, Models, and Applications*, Kapitel 3, Seiten 72–121. In: AIDAROUS, SALAH und THOMAS PLEVYAK [AiPl 94], 1994.

[Schr 96] SCHREINER, HANS-GÜNTHER: *Eine generische Managementarchitektur für offene heterognene Rechnernetze*. Dissertation, Universität Karlsruhe, Juli 1996.

[Sido 98] SIDOR, DAVID J.: *TMN Standards: Satisfying Today's Needs While Preparing for Tomorrow*. IEEE Communications Magazine, 36(3):54–64, März 1998.

[Sieg 96] SIEGEL, JON: *CORBA Fundamentals and Programming*. John Wiley & Sons, Inc., 1996.

[Sims 94] SIMS, OLIVER: *Business Objects: Delivering Cooperative Objects for Client-Server*. McGraw-Hill, 1994.

[Slom 94] SLOMAN, MORRIS (Herausgeber): *Network and Distributed Systems Management*. Addison-Wesley, Juni 1994.

[Sole 95] SOLEY, RICHARD MARK (Herausgeber): *Object Management Architecture Guide*. John Wiley & Sons, Inc., Dritte Auflage, Juni 1995.

[SOM 96] IBM CORPORATION: *SOMobjects Developer Toolkit Programmer's Guide Volume 2: Object Services*, Erste Auflage, März 1996.

[Stal 98] STALLINGS, WILLIAM: *SNMP and SNMPv2: The Infrastructure for Network Management*. IEEE Communications Magazine, 36(3):37–43, März 1998.

[StP 34] AONIX: *Software through Pictures/Object Modeling Technique: Creating OMT Models*. Aonix, Inc., 1997. Release 3.4.

[Stri 98] STRICKER, D.: *Nutzung der ODP Viewpoint Languages für das Management der verteilten Anwendung WWW*. Diplomarbeit, Technische Universität München, Mai 1998.

[StSy 94] STRUTT, COLIN und MARK W. SYLOR: *Digital Equipment Corporation's Enterprise Management Architecture*, Kapitel 22, Seiten 581–604. In: SLOMAN, MORRIS [Slom 94], Juni 1994.

[StWe 95] STURM, RICK und JON WEINSTOCK: *Application MIBs: Taming the Software Beast*. Data Communications, November 1995.

[Sun 98] SUN MICROSYSTEMS, INC.: *Java Dynamic Management Kit 3.0 (beta)*, August 1998. Part No. 805-4620-05.

[TAct 96] UNITED STATES OF AMERICA: *Telecommunications Act of 1996*. Public Law 104-104, Februar 1996.

[Thom 98] THOMPSON, J. PATRICK: *Web-Based Enterprise Management Architecture*. IEEE Communications Magazine, 36(3):80–86, März 1998.

[TrMR 97] TRISCHITTA, PATRICK R., ANTONIO JOSE GARCIA MEDINA und ROBERTO COFRE REMEDI: *The Pan American Cable System*. IEEE Communications Magazine, 35(12), Dezember 1997.

[UsBr 97] USLAENDER, THOMAS und HAJO BRUNNE: *Proposal for a CORBA Management Information Base*. OMG CORBA Management Workshop, Dublin, September 1997.

[VoDu 98] VOGEL, ANDREAS und KEITH DUDDY: *Java Programming with CORBA*. John Wiley & Sons, Inc., 2nd Auflage, 1998.

[Vogs 96] VOGS, T.: *Entwurf und Implementierung eines Konzepts zur Anbindung von Object Request Brokern an eine Netzmanagementplattform*. Diplomarbeit, Technische Universität München, Februar 1996.

[Wald 93] WALDBUSSER, STEVEN L.: *The Trend Towards Hierarchical Network Management*. The Simple Times, 2(6):4–6, November 1993.

[Wate 98] WATERMAN, RICHARD H.: *Operations Systems (OS) Unbundling: It's all about Interfaces*. In: SAHIN, VELI [NOMS 98], Seiten 775–784.

[WeAu 96] WELLENS, CHRIS und KARL AUERBACH: *Towards Useful Management*. The Simple Times, 4(3):1–6, Juli 1996.

[WhGu 98] WHITE, MATT und SMITHA GUDUR: *An Overview of the AgentX Protocol*. The Simple Times, 6(1):1–7, März 1998.

[XMP 92] *Systems Management: Management Protocols API (XMP)*. Preliminary Specification P170, X/Open Ltd., März 1992.

[XOM 91] *OSI-Abstract-Data Manipulation API (XOM)*. X/Open CAE Specification C180, X/Open Ltd., November 1991.

[X/Open 95a] *Systems Management: Common Management Facilities Volume 1*. Preliminary Specification P421, X/Open Ltd., Juni 1995.

[YeGY 91] YEMINI, YECHIAM, GERMAN GOLDSZMIDT und SHAULA YEMINI: *Network Management by Delegation*. In: KRISHNAN, I. und W. ZIMMER [INM-II 91], Seiten 95–107.

[Yemi 94] YEMINI, YECHIAM: *A Critical Survey of Network Management Protocol Standards*, Kapitel 2, Seiten 19–71. In: AIDAROUS, SALAH und THOMAS PLEVYAK [AiPl 94], 1994.

[Zele 95] ZELEK, MARK: *Comparing SNMP and CMIP*. CMIP Run, 4(1):1–5, 1995.

INDEX