Diplomarbeit

Entwicklung eines auf standardisierten Managementkonzepten basierenden Anwendungsmanagements am Beispiel von X.400

Johann Maurer

INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwicklung eines auf standardisierten Managementkonzepten basierenden Anwendungsmanagements am Beispiel von X.400

Bearbeiter . . . . : Johann Maurer
Aufgabensteller . . . . . . : Prof. Dr. Heinz-Gerd Hegering
Betreuer . . . . : Dr. Sebastian Abeck

Abgabedatum . . . . : 15. Mai 1995

Ehrenwörtliche Erklärung

Ich versichere, daß ich diese Diplomarbeit selbständig verfaßt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Mai 1995

....

(Unterschrift des Kandidaten)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

1.1 Verwaltung verteilter Systeme

In den letzten Jahren hat die Anzahl der vernetzten Rechensysteme stark zugenommen. Dieser Trend wird sich im Rahmen des Down- bzw. Rightsizings, also des Ersetzens von Großrechnern durch verteilte Systeme auf Basis von Workstation-Clustern, weiter fortsetzen.

Mit der Ablösung der zentralen Datenverarbeitung geht allerdings auch die zentrale Überwachbarkeit des Systems verloren. In verteilten Systemen ist es daher notwendig, eine adäquate Lösung zu deren zentraler Verwaltung zur Verfügung zu stellen.

Diese komplexe Aufgabe wird üblicherweise in vier Szenarien gegliedert:

- Netzmanagement — Verwaltung der Komponenten, aus denen das Rechnernetz zusammengesetzt ist (Router, Bridges, Hubs, Protokollstacks der Rechner)
- Systemmanagement — Verwaltung des Ressourcen, die die Rechner zur Verfügung stellen (Rechenzeit, Hintergrundspeicher, Benutzerverwaltung)
- Anwendungsmanagement — Verwaltung der (verteilten) Anwendungen, die im verteilten System betrieben werden
- Enterprisemanagement — Eingliederung dieser drei Szenarien in den betriebswirtschaftlichen Kontext

In der vorliegenden Arbeit werden Aspekte des Anwendungsmanagements am Beispiel eines verteilten Nachrichtenübermittlungssystems untersucht.

1.2 Bedeutung elektronischer Nachrichtenübermittlung

Wissen ist zu einem der wichtigsten Rohstoffe unserer Zeit geworden, schnellem und sicherem Informationsaustausch kommt daher mehr und mehr Bedeutung zu.

Die Vorteile der Nachrichtenübermittlung auf elektronischem Wege liegen auf der Hand: Das System ist ständig verfügbar und die Übertragungszeiten sind im Vergleich zur Briefpost um Größenordnungen geringer, was in Zeit- und Kostenersparnis resultiert. Daten, die in elektronischer Form vorliegen, können ohne Medienbruch vom Absender zum Empfänger übertragen werden, insbesondere ist auch asynchrone Kommunikation zwischen kooperierenden Anwendungen möglich.

Auf Grund dieser Vorteile wird der Informationsaustausch mehr und mehr auf elektronischer Basis durchgeführt. EMail-Systeme werden für den Geschäftsverkehr mit Kunden ebenso benutzt wie zur Abwicklung der Informationsflüsse innerhalb des Unternehmens.

Dauerhafte Verfügbarkeit und Zuverlässigkeit elektronischer Nachrichtenübermittlungssysteme ist daher zu einem Erfolgsfaktor geworden, was zentrale und integrierte Managementlösungen für solche Systeme notwendig macht.

1.3 Aspekte des Anwendungsmanagements

Im Netz- und Systemmanagement reicht in vielen Fällen die Verwaltung jeder einzelnen Ressource für sich aus. Der Kontext, in dem diese Komponenten interagieren, wird selten in seiner vollen Breite in der Managementlösung modelliert, stattdessen wird meistens entsprechendes Hintergrundwissen beim Verwaltungspersonal vorausgesetzt.

Um eine verteilten Anwendung angemessen verwalten zu können, reicht es nicht aus, nur die einzelnen Bausteine, aus denen sich die Applikation zusammensetzt, zu betrachten. Zwischen diesen Ressourcen existiert ein komplexes Netz von Abhängigkeiten, weshalb eine Gliederung gefunden werden muß, die auch die komponentenübergreifenden Aspekte und Zusammenhänge der Applikation berücksichtigt.

In der vorliegenden Arbeit wird zu diesem Zweck ein vierstufiges Managementmodell entwickelt:

- Bausteinebene — Verwaltung der einzelnen Ressourcen, aus denen sich die Anwendung zusammensetzt
- Anwendungsebene — Modellierung der anwendungsweiten (ressourcenübergreifenden) Aspekte
- Diensteebene — Betrachtung der von der Anwendung erbrachten Dienste
- Geschäftsebene — Einordnung der Anwendung in den betriebswirtschaftlichen Kontext

1.4 Zusammenfassung

Im nächsten Kapitel werden die verschiedenen Aspekte eines Nachrichtenübermittlungssystems am Beispiel der Recommendations X.400ff der ITU beschrieben. Neben einem Überblick über das Nachrichtenformat werden insbesondere Architektur und Dienstestruktur vorgestellt.

In Kapitel 3 werden zunächst die Anforderungen an eine Managementlösung aufgeführt. Im Anschluß daran wird ein vierstufiges Managementkonzept entwickelt, das Aspekte der Anwendung und ihrer Bausteine ebenso berücksichtigt wie die angebotenen Dienste und das die Applikation in einen betriebswirtschaftlichen Kontext einordnet.

Im folgenden Kapitel werden bestehende Managementansätze untersucht, wobei sowohl proprietäre Lösungen wie auch herstellerübergreifende Konzepte berücksichtigt werden. Insbesondere wird ein vom IAB genormter Ansatz vorgestellt und an Hand der im letzten Kapitel beschriebenen Anforderungen bewertet.

Um die Übertragbarkeit der erarbeiteten Konzepte auf die Praxis zu überprüfen, werden in Kapitel 5 zwei existierende Einsatzszenarien vorgestellt. Zuerst wird das firmeninterne MHS der BASF AG beschrieben, das durch seine weltweite Ausdehnung und der daraus resultierenden Managementproblematik von besonderem Interesse ist, anschließend wird der X.400-Mailverbund des DFN-Vereins vorgestellt, der Hochschulen und Forschungseinrichtungen in Deutschland verbindet.

Um festzustellen, inwieweit die erarbeiteten Konzepte mit heute verfügbaren Mitteln realisierbar sind, wird ein Prototyp spezifiziert, der im Rahmen dieser Arbeit zum Teil implementiert wurde.

Kapitel 6 gibt einen Überblick über die Entwicklungsumgebung. Zum einen wird Nexors Implementierung eines X.400-MTAs beschrieben, der von der BASF AG in dem in Kapitel 5 beschriebenen Szenario erfolgreich eingesetzt wird. Zum anderen wird die Managementplattform Spectrum von Cabletron vorgestellt, wobei insbesondere die objektorientierten und generischen Ansätze dieses Systems hervorgehoben werden.

In Kapitel 7 wird zunächst ein allgemeines Modell für eine Managementlösung vorgestellt, das anschließend für das gegebene Entwicklungsumfeld verfeinert wird. Desweiteren werden Ansätze für ein Ausbau des Prototypen und die damit verbundenen Probleme aufgezeigt.

Kapitel 8 gibt schließlich einen Ausblick auf neuartige Konzepte im Problemkreis Anwendungsmanagement, die in den nächsten Jahren eine zunehmende Rolle spielen dürften.

Kapitel 2

Message Handling nach X.400

Electronic Mail ist in den letzten Jahren zunehmend ein Erfolgsfaktor im kommerziellen und akademischen Bereich geworden. Die Möglichkeit zur schnellen und unkomplizierten Kommunikation innerhalb des Arbeitsbereichs und auch über diesen hinaus resultiert oft in einer Effizienzsteigerung.

Für den Nachrichten- und Datenaustausch existieren eine Vielzahl von frei verfügbaren und kommerziellen Softwarelösungen. Es existieren Angebote für Großrechner ebenso wie Programmsysteme für den Einsatz auf Workstation- oder auch PC-Netzen. Leider sind dadurch ein große Anzahl von proprietären Protokollen und Nachrichtenformaten entstanden, die untereinander nur bedingt bzw. überhaupt nicht kompatibel sind.

Um diesem Mißstand abzuhelfen, hat die ITU 1984 erstmals eine Reihe von Empfehlungen mit den Bezeichnungen X.400 und folgende veröffentlicht, die einen Standard für Meldungsübermittlungssysteme sowie die dafür vorgesehenen Dienste und Protokolle auf Basis des OSI-Referenzmodells der ISO veröffentlicht. Seitdem werden im Vierjahresabstand überarbeitete und erweiterte Versionen dieser Dokumente herausgegeben. Während 1988 die Architektur in großem Rahmen geändert wurde, wurden 1992 nur noch Detailkorrekturen und Erweiterungen vorgenommen.

Seit Veröffentlichung des Standards ist eine große Anzahl von Produkten auf den Markt gekommen. Trotzdem hat sich X.400 bisher weder im akademischen Bereich, wo größtenteils RFC822-basierte Mailsysteme eingesetzt werden, noch im den Abteilungsbereichen der Industrie, in denen nach wie vor proprietäre Produkte vorherrschen, in größerem Maße durchsetzen können.

Der Haupteinsatzbereich von X.400-basierten Nachrichtenübermittlungssystemen ist daher hauptsächlich im Backbone-Bereich zu finden, da

- dank der Standardierung die Interoperabilität der verschiedenen Produkte gewährleistet ist[1].
- für nahezu jedes proprietäre EMail-System Gateway-Software für den Übergang in das X.400-MHS existiert.

X.400 kann daher durchaus als Esperanto der Nachrichtenübermittlung bezeichnet werden.

Im folgenden werden die verschiedenen Aspekte eines Nachrichtenübermittlungssystems, das auf den X.400-Recommendations von 1992 basiert, beschrieben.

2.1 Diensteklassen

Innerhalb der X.400-Recommendations wird ein zweistufiges Dienstekonzept verfolgt (vgl. Abbildung 2.1). Basisdienst ist der Message Transfer Service, der die sichere Übermittlung einer Nachricht beliebigen Inhalts vom Absender zum Adressaten anbietet. Auf diesen Dienst setzen verschiedene komplexere Dienste auf, die Nachrichten eines bestimmten Formats übertragen und auf dieser Basis erweiterte Funktionalität anbieten.

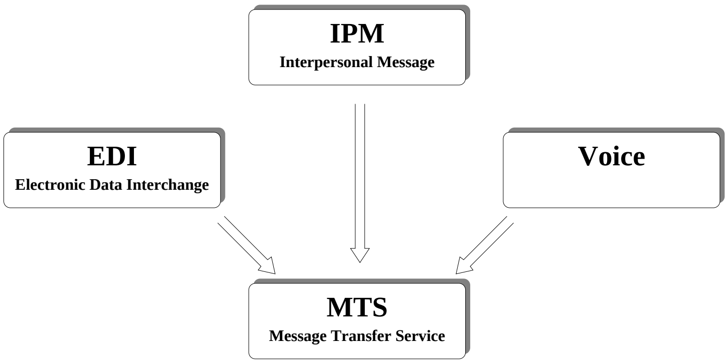


Abbildung 2.1: Diensteklassen in X.400

Im einzelnen sind das

- IPM für die Übermittlung von strukturierten Nachrichten (Text, Grafik, ...) zwischen Anwendern

[1] bzw. sein sollte, tatsächlich gibt es auch im X.400-Bereich Interoperabilitätsprobleme zwischen verschiedenen Herstellern.

- EDI für den Austausch von strukturierten Daten zwischen Anwendungen

- VOICE für den Austausch von Sprachinformation

2.2 Aufbau eines X.400 Message Handling Systems

2.2.1 Funktionales Modell von X.400

Das funktionale Modell beschreibt die Komponenten, die in ihrer Gesamtheit das X.400 Message Handling System (MHS) bilden.

In Abbildung 2.2 ist das Modell eines Message Handling Environments (MHE), wie es in der X.400-Empfehlung von 1992 beschrieben ist, dargestellt.

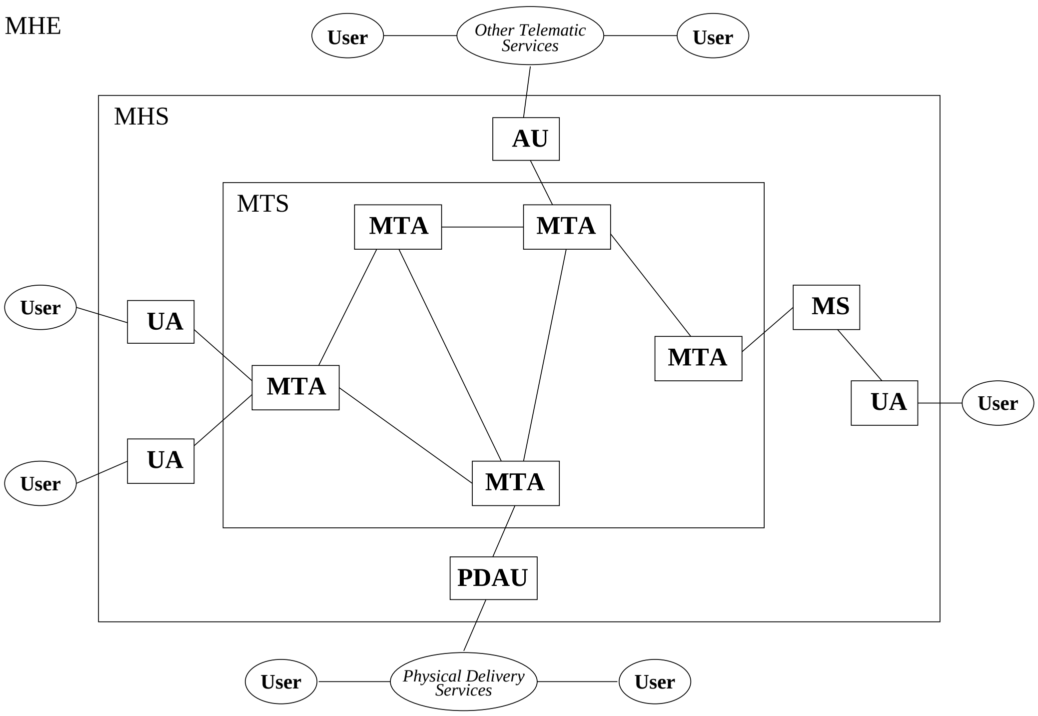


Abbildung 2.2: Funktionales Modell von X.400 (aus [X400])

Komponenten innerhalb des Message Handling Environments sind

User Benutzer können sowohl Menschen als auch Applikationsprozesse sein.

MHS Das Message Handling System stellt den Benutzern Dienste für Verwaltung und Übertragung von Nachrichten zur Verfügung.

Das Message Handling System selbst enthält

UA Ein User Agent ist ein Anwendungsprozeß, der dem Benutzer verschiedene Dienste wie Erstellen, Versand, Empfang und Anzeige von elektronischen Dokumenten an einer seinen Bedürfnissen angepaßten Schnittstelle zur Verfügung stellt.

MS Der Message Store bietet ein Nachrichtenarchiv an, in dem Meldungen für den Empfänger-UA zwischengespeichert werden.

AU Access Units realisieren Schnittstellen zu anderen Telematikdiensten wie Telex und Teletex.

PDAU Die Physical Delivery Access Unit realisiert den Übergang zur Briefpost.

MTS Das Message Transfer System hat die Aufgabe, Meldungen beliebiger Art vom Absender-UA zum Empfänger-UA (bzw. -MS) zu übertragen, wobei Zwischenspeicherung in den auf dem Versandweg liegenden Message Transfer Agents möglich ist.

Das Message Transfer System besteht aus mehreren

MTA Der Message Transfer Agent nimmt Nachrichten von User Agents, Message Stores oder anderen MTAs entgegegen und leitet sie an den nächsten MTA bzw. den Empfänger-UA oder -MS weiter.

2.2.2 Verwaltungsbereiche

Eine Gruppe von Komponenten, die mindestens einen MTA enthält und von derselben Organisation betrieben wird, bildet eine Management Domain (MD).

Management Domains, die von öffentlichen oder privaten Anbietern von MHS-Diensten betrieben werden, heißen Administration Management Domain (ADMD). Solche Anbieter sind entweder Mitglied der ITU oder wurden von einem Mitglied autorisiert. Innerhalb eines Landes kann es mehrere solche Anbieter geben, deren ADMD-Namen aber national eindeutig sein müssen.

Der Betreiber einer ADMD kann seinen Kunden verschiedene Dienste anbieten:

- Benutzung eines vom Betreiber gestellten User Agents
- Zugang mit privatem User Agent zum Betreiber-MTA
- Zugang mit privatem User Agent zum Betreiber-MS
- Zugang mit privatem MTA zum Betreiber-MTA
- Benutzung einer vom Betreiber gestellten Access Unit

Management Domains, die von anderen Organisationen betrieben werden, heißen Private Management Domains (PRMD). Eine PRMD unterhält Verbindungen zu einer oder mehreren ADMDs auf Basis einer MTA-MTA-Kommunikation. Eine PRMD kann als Übergang zu weiteren Management Domains dienen oder länderübergreifend sein, wenn nationale Bestimmungen und das Einverständnis aller Teilnehmer dies erlauben. Der Name einer PRMD muß national bzw. zumindestens innerhalb der angeschlossenen ADMD eindeutig sein. Ist eine PRMD an mehrere ADMDs angeschlossen, kann sie entsprechend mehrere Namen haben.

Ein Beispielszenario ist in Abbildung 2.3 angegeben.

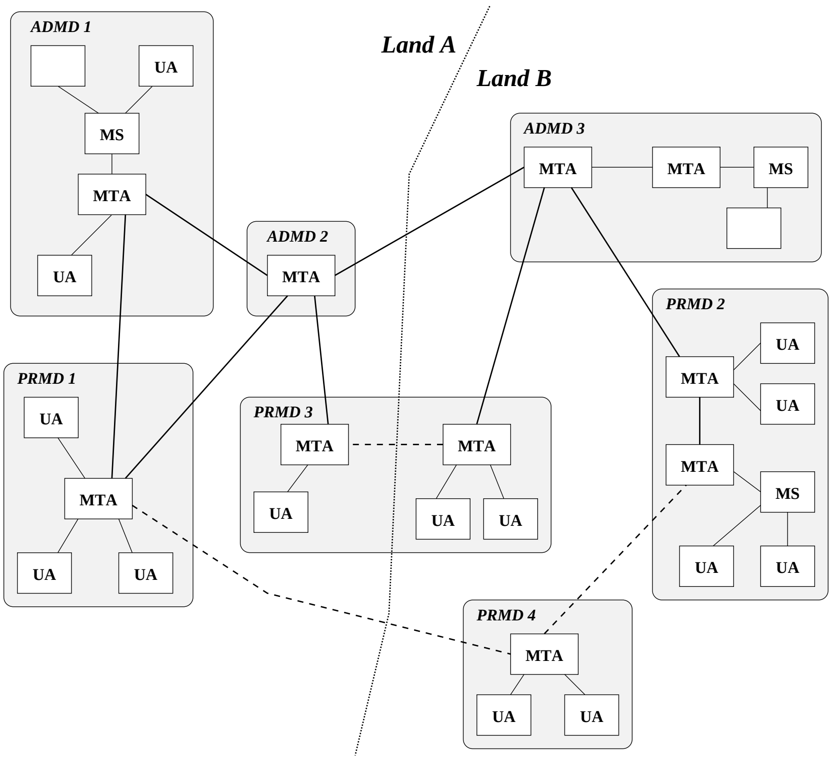
2.3 Nachrichten im X.400-MHS

2.3.1 Adreßformat

Zur Identifikation von Absender und Empfänger werden Originator/Recipient Names verwendet. Ein O/R Name besteht aus einem Directory Name, einer O/R Address oder beidem.

Directory Name

Ein Verweis auf den Directory-Eintrag des Empfängers. Dieser Eintrag enthält die O/R Address des Adressaten. Einen Überblick über ein Directory-System nach X.500 gibt Abschnitt 2.5.



Eine gestrichelte Linie bedeutet, daß ein solcher Link ggf. nicht erlaubt ist

Abbildung 2.3: Beispiel für Management Domains (aus [X400])

O/R Address

O/R-Adressen sind Attributlisten. Ein Attribut besteht aus einem Typ, etwa Vorname oder Land, und dem zugehörigen Wert. [X402] definiert eine Reihe von Standardattributen, daneben können aber auch innerhalb einer Management Domain domain-private Attribute definiert werden.

Die gängigsten Standardattributtypen sind country-name, administration-domain-name, private-domain-name, organization-name, organizational-unit-name und personal-name. Es existieren weitere Attributtypen, um Teilnehmer eines elektronischen Nachrichtenübermittlungssystems zu adressieren, sowie Attributtypen wie etwa street-address, die im Zusammenhang mit PDAUs verwendet werden.

Es existieren fünf verschiedene Darstellungsarten für O/R-Adressen, mnemonische, numerische, formatierte und unformatierte postalische sowie terminal-spezifizierende Adressen. Jede dieser Darstellungsarten setzt die Existenz einer bestimmten Teilmenge von Attributtypen voraus. Ein Beispiel für eine mnemonische O/R-Adresse ist

given-name=Hans
sur-name=Maurer
organizational-unit-name=Informatik
private-domain-name=TU-Muenchen
administration-domain-name=D400
country-name=DE

bzw. etwas kürzer

`G=Hans;S=Maurer;OU=Informatik;PRMD=TU-Muenchen;ADMD=D400;C=DE`

Enthält ein O/R Name beide Angaben, wird die Nachricht an Hand der O/R Address transportiert. Auf den Directory Name wird nur im Fehlerfall zurückgegriffen.

2.3.2 Nachrichtenformate

Das MTS kennt drei verschiedene Typen von Nachrichten.

Messages sind die Nachrichten im eigentlichen Sinn. Sie dienen dem Transport der Nutzdaten. Eine Message gliedert sich in zwei Komponenten (vgl. Abbildung 2.4), den Envelope und den Content.

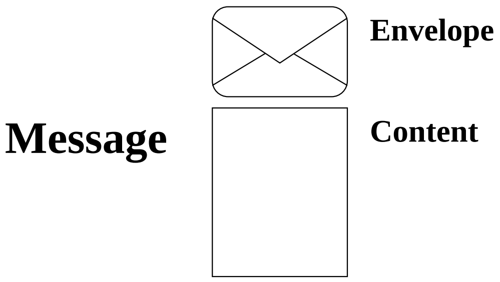


Abbildung 2.4: Aufbau einer Nachricht (aus [X400])

Der Envelope variiert von Transmittal Step (siehe Abschnitt 2.4.1) zu Transmittal Step. Er enthält insbesondere die Adresse des Absenders und die Adressen der potentiellen Empfänger und gibt an, in welchem Format der Inhalt der Nachricht abgelegt ist.

Der Content enthält die eigentlichen Nutzdaten. Er wird vom MTA abgesehen von Konvertierung nicht verändert.

Probes sind Nachrichten, die administrativen Zwecken dienen. Sie bestehen nur aus dem Envelope, besitzen also keinen Content. Probes durchlaufen abgesehen von Delivery und DL-Expansion dieselben Transmittal Steps wie Messages, mit ihrer Hilfe kann also festgestellt werden, ob ein Adressat (bzw. eine Verteilerliste) erreichbar ist.

Reports sind Statusmeldungen des MTS an den Absender von Messages oder Probes, die die Auslieferung einer Nachricht oder die Nichtauslieferbarkeit anzeigen. Ensprechend gibt es Delivery-Reports, die auf Wunsch des Absenders während der Schritte Delivery, Export und DL-Expansion, in jedem Fall aber bei Affirmation versandt werden, sowie Non-Delivery-Reports, die beim Durchführen von Non-Delivery und Non-Affirmation erzeugt werden.

Während der Message Transfer Service keine weiteren Formatvorgaben macht, spezifizieren die in Abschnitt 2.1 beschriebenen Dienste der oberen Ebene eine Dokumentarchitektur für den Content einer Nachricht. Dieser Bestandteil wird also weiter strukturiert, was am Beispiel des Interpersonal Messaging Service beschrieben werden soll.

Bei Interpersonal Messages besteht der Content (siehe Abbildung 2.5) aus

- dem Header, der verschiedene zusätzliche Metadaten bzgl. der Nachricht enthält, unter anderem den Kurztitel der Nachricht sowie eine Beschreibung des Typs und der Struktur der Bodyparts, und
- dem Body, der seinerseits aus mehreren (ggf. hierarchisch angeordneten) Bodyparts besteht. Bodyparts können von verschiedenem Typ sein, etwa Text, Sprache, Binärdatei, Fax oder Grafik.

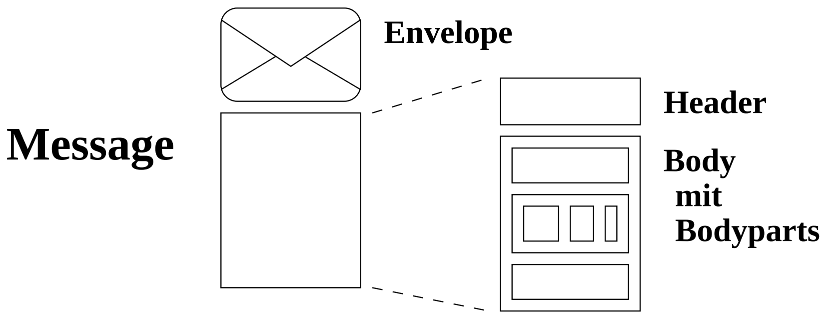


Abbildung 2.5: Aufbau einer Interpersonal Message (aus [X400])

2.4 Transport von Nachrichten

2.4.1 Abläufe beim Versand einer Nachricht

Der Weg, den einer Nachricht vom Erstellen durch den Absender bis zur Anzeige beim Empfänger durchläuft, läßt sich in verschiedene Teilabschnitte gliedern. Die X.400-Recommendation unterscheidet zwischen Transmittal Steps, innerhalb derer eine Nachricht von einer Komponente des MHS zu einer benachbarten transportiert wird, und Transmittal Events, bei denen die Nachricht innerhalb einer Komponente bearbeitet wird.

Der Ablauf einiger Transmittal Steps, im einzelnen Indirect Submission, Submission, Transfer, Delivery und Retrieval, sowie die dafür eingesetzten Protokolle sind genormt, um Interoperabilität zwischen den Komponenten verschiedener Hersteller zu gewährleisten, während die Implementierung der übrigen Steps dem einzelnen Anbieter überlassen bleibt. Transmittal Steps sind:

Origination Der Benutzer erstellt mit Hilfe des User Agent eine Nachricht.

Indirect Submission Der User Agent übermittelt die Nachricht an einen Message Store zur Weiterleitung in das Message Transfer System.

Submission Der User Agent bzw. Message Store übergibt die Nachricht an einen angeschlossenen Message Transfer Agent.

Import Eine (Physical Delivery) Access Unit leitet eine Nachricht aus einem externen Nachrichtenübermittlungssystem an einen angeschlossenen Message Transfer Agent weiter.

Transfer Message Transfer Agents tauschen Nachrichten untereinander aus.

Export Der Message Transfer Agent übermittelt eine für ein externes Nachrichtenübermittlungssystem bestimmte Nachricht an die zuständige (Physical Delivery) Access Unit.

Delivery Der dem Ziel nächste Message Transfer Agent leitet die Nachricht an den entsprechenden User Agent bzw. Message Store weiter.

Retrieval Die Nachricht wird von Message Store zum User Agent übertragen.

Receipt Der Empfänger sieht mit Hilfe des User Agent die Nachricht ein.

Im Gegensatz zu einem Transmittal Step ist bei einem Transmittal Event nur eine Komponente des MHS, nämlich der MTA, involviert. Daher spielt die Implementierung von Transmittal Event für die Interoperabilität keine Rolle, weswegen in den Recommendations nur deren Funktionalität, aber keine Implementierungsvorgaben beschrieben sind. Das Ergebnis eines Transmittal Events kann Einfluß auf die Auswahl und den Ablauf des nächsten Transmittal Steps haben. Es existieren folgende Eventtypen:

Splitting Eine zu übertragende Nachricht kann mehrere potentielle Empfänger haben. Ein Konflikt ergibt sich dann, wenn für diese Empfänger der nächste Transmittal Step oder Event variiert, etwa, weil diese Empfänger über verschiedene Partner-MTAs zu erreichen sind oder weil für einen Empfänger eine Konvertierung des Inhalts vorgenommen wird. Dieses Problem wird dadurch aufgelöst, daß die Nachricht mehrfach kopiert wird und die Empfänger so auf Envelopes verteilt werden, daß für jede Kopie der nächste Bearbeitungsschritt eindeutig ist.

Joining Dieser Event ist die Umkehrung von Splitting.

Name Resolution Der Directory Name des Empfängers wird zur O/R Address aufgelöst.

DL Expansion Eine Verteilerliste wird expandiert.

Eine Verteilerliste ist ein Mechanismus, der es erlaubt, komfortabel mit einer Nachricht mehrere User anzusprechen. Dazu wird eine neue O/R Address auf einem MTA eingerichtet, der eine Liste von Empfängern (bzw. deren O/R Names) zugewiesen wird.

Nimmt der MTA eine Nachricht entgegen, die an eine Verteilerliste gerichtet ist, wird geprüft, ob der Absender die Berechtigung besitzt, diese Verteilerliste zu benutzen. Ist das der Fall, wird die Nachricht an alle in der Liste aufgeführten O/R Names weitergeleitet. Probes, die an die Liste gerichtet sind, gelten als erfolgreich ausgeliefert, werden also nicht an die Listenteilnehmer weitergeleitet.

Redirection Eine Empfängeradresse wird durch eine neue ersetzt, die Nachricht somit umgeleitet.

Conversion Der MTA konvertiert den Content der Nachricht in eine andere Codierung, beispielsweise, weil der nächste Transmittal Step ein bestimmtes Format voraussetzt.

Non-Delivery Der Message Transfer Agent stellt fest, daß eine Message nicht ausgeliefert werden kann, und schickt einen Non-Delivery-Report an den Absender.

Non-Affirmation Der Message Transfer Agent erkennt, daß eine Probe nicht an ihr Ziel gelangen kann, und schickt einen Non-Delivery-Report an den Absender.

Affirmation Der Message Transfer Agent ermittelt, daß das Ziel einer Probe ein lokaler Nutzer ist, und schickt einen Delivery-Report an den Absender.

Routing Der Message Transfer Agent wählt die Nachbar-Instanz im MHS aus, an die die Nachricht weitergeleitet wird.

Ein Beispiel dafür, welche Steps und Events eine Nachricht vom Absender zum Empfänger durchläuft, ist in Abbildung 2.6 aufgeführt.

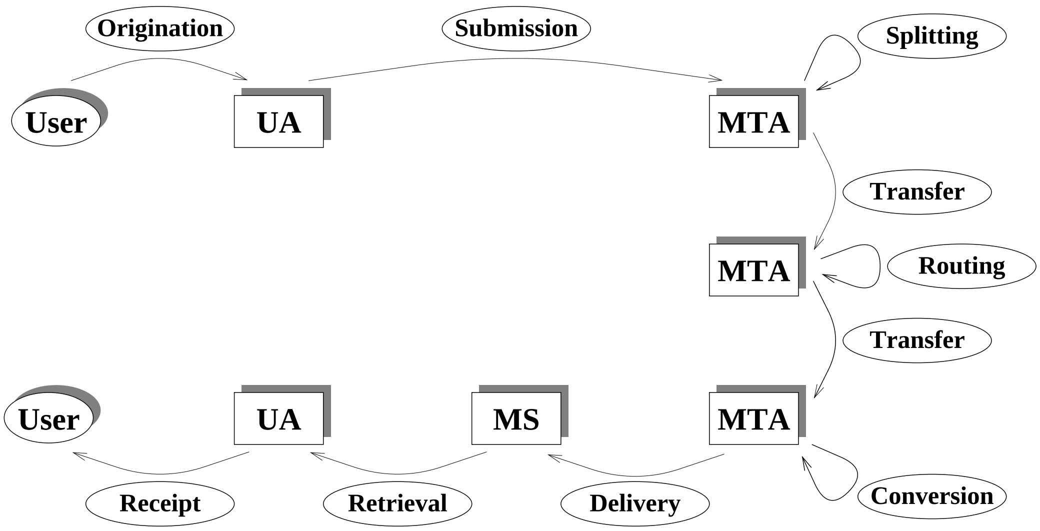


Abbildung 2.6: Auslieferungsweg einer Nachricht

2.4.2 Protokolle

In den X.400 Recommendations werden drei Protokolle spezifiziert:

P1 Mit diesem Protokoll tauschen Message Transfer Agents die zu übermittelnden Nachrichten untereinander aus ([X482]). Dieses Protokoll wird also vom Transmittal Step Transfer benutzt.

P3 User Agents bzw. Message Stores kommunizieren unter Verwendung dieses Protokolls mit Message Transfer Agents ([X483]). Submission und Delivery verwenden dieses Protokoll.

P7 Dieses Protokoll wird zum Informationsaustausch zwischen User Agent und Message Store benutzt ([X484]). Mit diesem Protokoll werden Indirect Submission und Retrieval abgewickelt.

Kommunikationsbeziehungen, die in Abbildung 2.2 angedeutet sind, aber von diesen drei Protokollen nicht abgedeckt werden, werden vom Hersteller durch proprietäre Protokolle realisiert.

Hinweis: Im Modell von 1984 war eine Untergliederung der Schicht 7 des OSI-Modells in eine Message Transfer Layer (MTL) und eine User Agent Layer (UAL) spezifiziert. Entsprechend existierte eine Klasse von UA-UA-Protokollen, etwa das in [X481] definierte P2-Protokoll, das einen Mechanismus zum Austausch von Interpersonal Messages vorsah. Mit der Abschaffung der Schichtenteilung im Standard von 1988 wurden diese Protokolle obsolet und durch entsprechende äquivalente Dokumentarchitekturen ersetzt.

2.5 Verzeichnissysteme nach X.500

In den Recommendations X.500ff und folgende hat die ITU erstmals 1988 die verschiedenen Aspekte eines verteilten Verzeichnisdienstes spezifiziert, um zu ermöglichen, daß Systeme verschiedener Hersteller unter verschiedem Management, verschiedener Komplexität und unterschiedlichen Alters zusammenarbeiten können.

Eine der Zielsetzungen der Definition war es, Objekte innerhalb des Netzes mit „benutzerfreundlichen“ Namen zu benennen, die nur von organisatorischen, nicht aber von technischen Gegebenheiten abhängen. Eine weitere Funktion ist die Abbildung eines solchen Namens auf die tatsächliche Adresse des bezeichneten Objekts. Somit stehen für Objekte, die ihre Adresse dynamisch ändern, statische Namen zur Verfügung, womit eine weitere Abstraktionsstufe eingeführt wird.

2.5.1 Struktur der Datenbasis

Die Datenbasis des Verzeichnisses, die sog. Directory Information Base (DIB) besteht aus einzelnen Einträgen (Entries), von denen jeder verschiedene Informationen über das beschriebene Objekt enthält. Jeder Eintrag ist aus

Attributen zusammengesetzt, die ihrerseits aus einem Typ (z.B. Name, O/R Address) und einem oder mehreren Werten bestehen. Welche Attribute ein bestimmter Entry enthalten muß bzw. kann, hängt von der Klasse ab, der das beschriebene Objekt angehört.

Die DIB ist baumförmig als Directory Information Tree (DIT) strukturiert, wobei die einzelnen Einträge die Knoten darstellen. Einträge, die weit oben im Baum stehen, repräsentieren oft Länder oder Organisationen, während tieferliegende Einträge meist Personen, Rollen oder Applikationsprozesse beschreiben.

Ein Eintrag wird eindeutig identifiziert durch seinen Distinguished Name (DN), der aus dem Pfad von der Wurzel zu diesem Knoten abgeleitet wird. Der DN eines Knotens ergibt sich durch Konkatenation des DN seines Vaterknotens mit dem Relative Distinguished Name des Knotens selbst, der aus einem oder mehreren ausgewählten Attributen und Werten dieses Knotens zusammensetzt. Die Wurzel des Baumes besitzt keinen (bzw. einen leeren) Distinguished Name. Grundvoraussetzung für diese Namensbildung ist natürlich, daß alle Kinder eines jeden Knoten paarweise verschiedene Relative Distinguished Names besitzen.

Neben Einträgen, die unmittelbar ein Objekt repräsentieren, können Blattknoten im Baum auch sogenannte Aliaseinträge sein. Ein solcher Entry enthält einen Verweis auf einen anderen Eintrag innerhalb des Baums. Mit diesem Mechanismus ist es möglich, ein Objekt im Directory durch verschiedene Namen anzusprechen.

Ein Beispiel für einen solchen Directory Information Tree ist in Abbildung 2.7 dargestellt, der Distinguished Name des hervorgehoben Entries ist `C=DE@O=Technische Universitaet Muenchen@OU=informatik`.

2.5.2 Das verteilte Verzeichnis

Das Directory System besteht aus einem oder mehreren Directory System Agents (DSA). An definierten Punkten erlaubt das Directory Lese- und ggf. Schreibzugriff durch User, die zu diesem Zweck einen Directory User Agent verwenden (vgl. Abbildung 2.8).

Jeder DSA beinhaltet ein Fragment des Directory Information Tree. Jeder Eintrag wird von genau einem DSA-Administrator verwaltet und ist eindeutig einem DSA innerhalb des Directory zugeordnet. Allerdings ist vorgesehen, daß Kopien in anderen DSA angelegt bzw. die Ergebnisse von Verzeichnisanfragen gecachet werden.

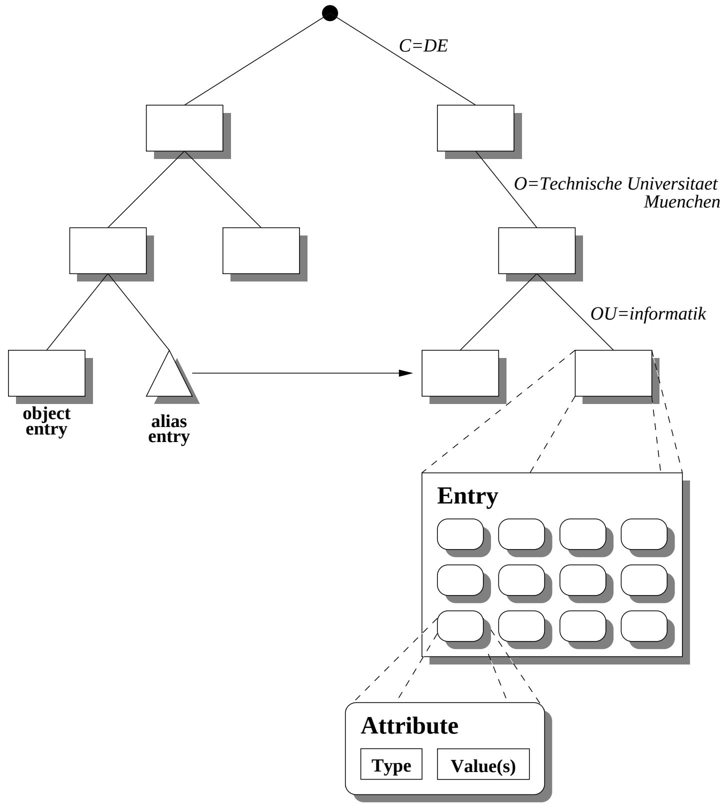


Abbildung 2.7: Struktur der Directory Information Base (nach [X500])

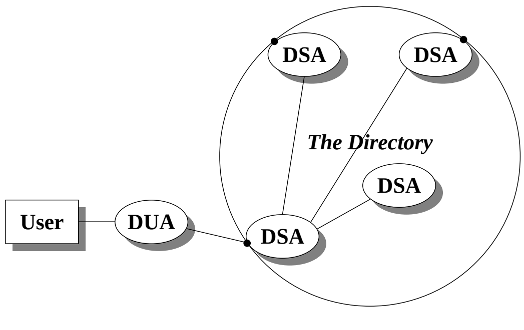


Abbildung 2.8: Struktur eines Verzeichnissystems (aus [X500])

Um den Standort eines Eintrags innerhalb des verteilten Verzeichnisses zu ermitteln, existieren verschiedene Suchverfahren, die zum Teil auf Konzepten der Tiefen- bzw. Breitensuche basieren, zum Teil aber auch mit Referenzen arbeiten. In letzterem Fall wird an den Kanten des DIT, an denen die Verwaltung an einen anderen DSA übergeht, ein Verweis auf diesen DSA abgelegt.

2.5.3 Schnittstellen zwischen MHS und DS

Es gibt mehrere Anwendungsbereiche, in denen das MHS Nutzen aus dem Verzeichnissystem ziehen kann:

- „Benutzerfreundliche" Namen

  Wie bereits in Abschnitt 2.3.1 aufgezeigt wurde, ist der Directory Name ein möglicher Bestandteil eines Originator/Recipient Names. Ist keine O/R Address angegeben bzw. eine Auslieferung auf Basis dieser Adresse fehlgeschlagen, wird die O/R Address im Verzeichniseintrag des Benutzers verwendet. Im ersten Fall kann diese Abbildung bereits im UA erfolgen, ansonsten findet sie während des Name-Resolution-Events (vgl. Abschnitt 2.4.1) statt.

- Verteilerlisten

  Dieses Verfahren ist auch bei Verteilerlisten anwendbar. In diesem Fall wird die Liste der Adressaten aus dem Verzeichniseintrag der Liste ermittelt.

- Fähigkeiten des empfangenden User Agents

  Im Verzeichniseintrag des Empfängers wird abgelegt, welche Fähigkeiten der von ihm verwendete User Agent besitzt. Mit dieser Information kann der UA des Absenders die Nachricht derart gestalten, daß der Empfänger sie auswerten kann. Andernfalls kann auch ein MTA auf Basis dieser Information Konvertierungen vornehmen.

- Authentifizierung

  Wenn zwei Komponenten im MHS eine Verbindung zueinander aufbauen, muß jede die Identität der anderen prüfen. Dies kann auf der Basis von Informationen im Verzeichnis erfolgen.

- Konfiguration

  Die verschiedenen Komponenten des MHS können selbst als Application Process einen Eintrag im Directory haben. In einem solchen Eintrag können verschiedene mittel- und langfristig konstante Informationen untergebracht werden, etwa Routingtabellen.

Damit eine Komponente eines MHS auf Informationen im Directory zugreifen kann, muß sie um die Funktionalität eines Directory User Agents erweitert werden. Abbildung 2.9 zeigt ein Beispiel für das Zusammenspiel zwischen MHS und DS.

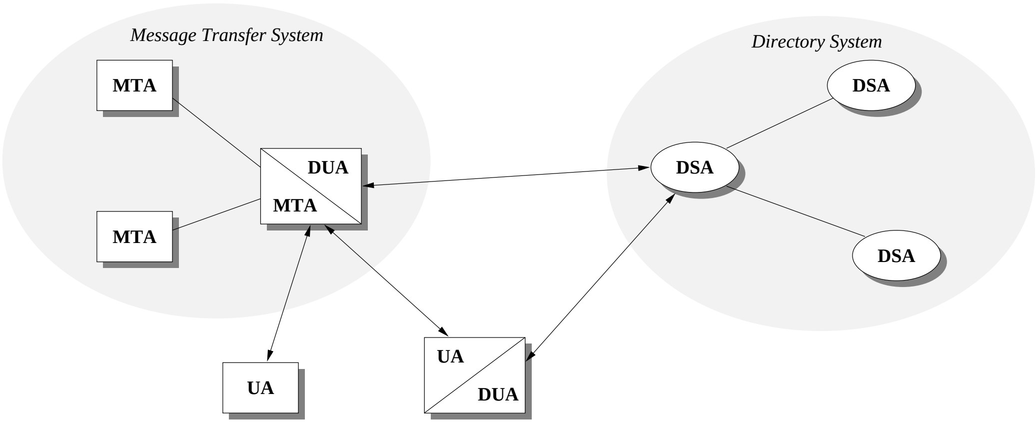


Abbildung 2.9: Zusammenarbeit zwischen MHS und Directory System (aus [X400])

Kapitel 3

Managementaspekte eines Message Handling Systems

In diesem Kapitel werden zunächst die Anforderungen an eine Anwendungsmanagementlösung aufgeführt. Aufgrund der Beschränkungen von Netz- und Systemmanagementmethoden wird anschließend ein umfassendes Konzept für das Management von verteilten Applikationen vorgestellt und am Beispiel eines X.400-MHS vertieft.

3.1 Anforderungen an eine Managementlösung

Eine umfassende Managementlösung für die Verwaltung eines Message Handling Systems muß eine große Zahl von Anforderungen erfüllen. Diese sind in den nächsten Abschnitten aufgeführt, wobei als Gliederungsstruktur die Managementfunktionsbereiche des OSI-Modells gewählt wurden. Aufgrund seiner besonderen Stellung innerhalb des MHS wird der Themenkreis Routing in einem eigenen Abschnitt untersucht.

3.1.1 Konfiguration

Die Funktion jeder Komponente innerhalb des MHS wird durch eine Menge von Parametern beschrieben. Konfigurationsmanagement umfaßt die Registration aller Komponenten innerhalb des MHS und die Abfrage und Manipulation der Parameter.

Jede der Komponenten eines MHS hat ein große Anzahl von Einstellmöglichkeiten. Von der Managementplattform sollten diese Parameter modifiziert werden können, unabhängig davon, ob sie komponentenlokal oder in einem Verzeichnissystem abgelegt sind. Weiterhin sollte die applikationsweite Konsistenz der Einstellungen geprüft werden.

3.1.2 Fehler

Aufgabe des Fehlermanagements ist die Behandlung von anomalen Zuständen im MHS.

Zum einen sollte der Funktionsstatus der Komponenten überwacht werden, zum anderen sollte die Managementstation informiert werden, wenn Fehler auftreten. Dazu gehören Probleme beim Verbindungsaufbau ebenso wie der Verlust von Nachrichten.

3.1.3 Leistung

Während Fehlermanagement die stabile Funktion des Netzes sichert, ist Leistungsmanagement als dessen konsequente Fortsetzung für das „gute" Funktionieren des Netzes verantwortlich.

Zunächst muß der Terminus „gut" definiert werden. Zu diesem Zweck müssen Dienstgüteparameter (Quality of Service, QoS) wie die maximale Laufzeit einer Nachricht oder der maximale Rückstau definiert werden.

Anschließend gilt es, eine Reihe von Verkehrs- und Auslastungsdaten zu erfassen und auszuwerten. Hier bietet sich vor allem das Nachrichtenaufkommen im Gesamtnetz sowie in definierten Teilnetzen bis zu einzelnen Links an. Auch kann die Zahl der in den einzelnen Komponenten zwischengespeicherten Nachrichten und ihre Verweilzeit Aufschluß über Leistungsengpässe geben.

Die erfaßten Daten werden nun mit den Qualitätsvorgaben verglichen. Zeichnet sich eine Überlastung ab, wird die Managementstation benachrichtigt, die ggf. automatisch Steuerungsmaßnahmen ergreift.

Um eine langfristige, strategische Planung zu erlauben, sollten die Leistungsdaten außerdem für statistische Auswertungen archiviert werden.

3.1.4 Sicherheit

Sicherheitsmanagement hat die Aufgabe, die Datensicherheit innerhalb des MHS zu gewährleisten. Diese ist bedroht durch technische Ausfälle sowie durch menschlichen Vorsatz oder Fahrlässigkeit.

Die Sicherheitsproblematik in Message Handling Systems läßt sich grob in die zwei Bereiche Nachrichtenfluß und Managementzugriff aufteilen.

- Die Sicherheit und Integrität der transportierten Nachrichten muß sichergestellt werden.

  Um das zu erreichen, muß unautorisierter Zugriff auf das MHS verhindert werden. Zu diesem Zweck müssen sich die zwei Komponenten beim Verbindungsaufbau gegenseitig authentifizieren. Es sollte möglich sein, die Funktion und die Parameter der Identifizierungsalgorithmen und ggf. Ver- und Entschlüsselungsalgorithmen zu überwachen und zu konfigurieren. Außerdem sollte die Managementstation benachrichtigt werden, falls ein Angriff auf das bzw. eine Sicherheitslücke im MHS festgestellt wird.

- Managementzugriffe auf einzelne MHS-Komponenten dürfen nur autorisiertem Personal erlaubt werden. Im wesentlichen bestehen hier dieselben Managementbedürfnisse wie beim ersten Punkt.

3.1.5 Abrechnung

Der Betreiber des MHS will die durch das Anbieten dieses Dienstes entstehenden Kosten auf die Benutzer dieses Dienstes umlegen.

Zu diesem Zweck muß zunächt eine Accounting-Policy bestimmt werden, in der festgelegt wird, welche Dienste abgerechnet werden und welches Abrechnungsverfahren verwendet wird.

Das Managementsystem legt anschließend die relevanten Accountinginformationen fest (etwa eingehende und ausgehende Nachrichten, deren Absender und Empfänger) und aktiviert die Protokollierung dieser Informationen in den einzelnen Komponenten.

Diese Daten müssen anschließend gesammelt, korreliert und einzelnen Benutzern zugeordnet werden.

3.1.6 Routing

Der Themenbereich Routing wird in einem eigenen Abschnitt behandelt, weil er einerseits zu den wichtigsten Gebieten im MHS-Management gehört und andererseits mit mehreren der oben angeführten Gliederungspunkte interferiert.

Aspekte dieses Bereichs sind:

- Die Topologie des MHS muß konfiguriert bzw. erfaßt werden.
- Die Routing-Algorithmen, die in den einzelnen MTAs verwendet werden, müssen parametrisiert und vor allem aufeinander abgestimmt werden.
- Um Ausfällen oder Leistungseinbrüchen entgegenzuwirken, sollten Alternativrouten verfügbar sein, auf die im Problemfall automatisch zurückgegriffen werden kann. Führt man diesen Ansatz fort, ist auch eine dynamische Wegewahl, die von betreiberdefinierten Parametern abhängig ist, denkbar.

3.2 Integration der Managementlösung

Da sich verteilte Anwendungen auf verschiedene Netz- und Systemressourcen abstützen, reicht es für ein adäquates Management nicht aus, die Applikation isoliert zu betrachten.

Vielmehr müssen Schnittstellen zum Netz- und Systemmanagement vorhanden sein, um eine Korrelation der dort erfaßten Managementinformation mit den Aspekten der verteilten Anwendung zu ermöglichen.

So kann beispielsweise ein User Agent nicht verwendet werden, wenn die Bedienoberfläche des Systems nicht verfügbar ist. Genauso können MTAs nicht miteinander kommunizieren, wenn Probleme im Netzbereich vorliegen.

Im Fall des X.400-MHS müssen auch die Bezüge zum Verzeichnissystem berücksichtigt werden, da die MHS-Funktionen sich zum Teil stark auf dessen Dienste abstützen.

3.3 Beschränkungen von Netz- und Systemmanagementmethoden

Jede Komponente eines X.400-MHS, beispielsweise ein MTA, benötigt für korrekte Funktion bestimmte Netz- und Systemdienste wie Prozessorzeit, Hauptspeicher, Hintergrundspeicher und End-zu-End-Verbindungen zu Partnerinstanzen.

Aufgabe der Netz- und Systemmanagementtools ist es, mittels geeigneter Aktionen diese Voraussetzungen zu schaffen und aufrechtzuerhalten. Funktionierendes Netz- und Systemmanagement ist also eine Grundvoraussetzung für die Funktionsfähigkeit der Anwendung. Anwendungsmanagement ist daher nur auf dieser Basis sinnvoll.

Es stellt sich nun die Frage, welche der in Abschnitt 3.1 gestellten Anforderungen bereits mit Netz- und Systemmanagementmethoden abgedeckt werden können.

Systemmanagementtools können die einzelnen Prozesse, aus denen sich die lokale Komponente der verteilten Anwendung zusammensetzt, überwachen. Sie können feststellen, ob die Prozesse laufen, ob sie Rechenzeit verbrauchen, wieviel Hauptspeicher und Hintergrundspeicher sie belegen. Mit Netzmanagementtools können die Art und Zahl der unterhaltenen Verbindungen und der Verkehr auf diesen Strecken sowie die Zahl der bereits beim Aufbau fehlgeschlagen Verbindungen (und die Ursache für den jeweiligen Fehlschlag) ermittelt und überwacht werden.

Allerdings lassen diese Daten ohne Zusatzwissen keinerlei Rückschlüsse auf die Funktion der Anwendung zu:

- Über bestehende Netzverbindungen müssen nicht notwendigerweise sinnvolle (verwertbare) Daten fließen.

- Prozesse können Systemressourcen verbrauchen, ohne die spezifizierten Dienste bereitzustellen, wie es etwa im Falle eines Deadlockzustands bei einem Datenbankserver vorkommen kann.

Generell muß festgestellt werden, daß mit Netz- und Systemmanagementtools den ermittelten Daten nicht die zugehörige anwendungsbezogene Semantik zugeordnet werden kann. Solche Methoden können also nur dazu dienen, Fehlersituationen aufzudecken, sind aber umgekehrt nicht dazu imstande, eine positive Aussage über das Funktionieren einer Anwendung zu treffen.

3.4 Ein Managementkonzept für X.400

Um den vorgestellten Anforderungen im Rahmen eines Managementmodells gerecht zu werden, ist es notwendig, weitere Gliederungskriterien zu finden, nach denen die Managementaufgaben geordnet werden können. Einen sehr guten Ansatz dafür bietet das im nächsten Abschnitt vorgestellte TMN-Modell.

3.4.1 Das TMN-Modell als Grundlage

Dieses Modell wurde von der ITU zur Beschreibung und Standardisierung der verschiedenen Gesichtspunkte des Netzmanagements bei Telefongesellschaften entwickelt und befaßt sich daher mit der Verwaltung von Telefon- und Datennetzen. Das TMN-Modell gliedert sich in vier Managementebenen, wie sie in Abbildung 3.1 dargestellt werden:

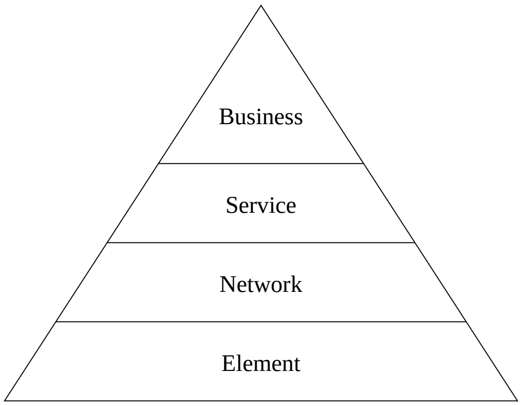


Abbildung 3.1: Ebenen im TMN-Modell (aus [Wil94])

- Die Element-Ebene beschäftigt sich mit allen Aspekten, die bei der Verwaltung einzelner Netzkomponenten auftreten.
- Auf Network-Ebene werden die einzelnen Komponenten zu einer globalen Sicht des Netzes abstrahiert.
- Schwerpunkt der Service-Ebene ist die Verwaltung der vom Netz erbrachten Dienste.
- Die Integration des Netzes in die betriebswirtschaftliche Aufbau- und Ablauforganisation geschieht in der Business-Ebene.

Wie bereits erwähnt, ist dieses Modell für Netzmanagementbelange entwickelt worden. Das Gliederungsschema läßt sich allerdings auch auf Belange des Anwendungsmanagements anwenden. So lassen sich diese Aufgaben ebenfalls ist das TMN-Ebenenschema einordnen.

Während sich die Objekte und Funktionen der Element-Ebene noch stark am Aufbau der Anwendung orientieren, nehmen in den höheren Ebenen abstraktere Objekte und komplexere Funktionen zunehmend größeren Stellenwert ein. Mit zunehmender Höhe im TMN-Modell nimmt also die Bedeutung des Top-Down-Ansatzes zu.

Da sich die höheren Ebenen auf die jeweils tieferliegenden abstützen, liefert das Modell gleichzeitig eine Strategie für eine stufenweise Integration aller betrachteten Belange in ein Managementsystem. Sind alle Aspekte der Element-Ebene integriert, kann darauf aufsetzend die Implementierung der Network-Ebene begonnen werden und so fort.

Im Anschluß wird ein Überblick über die nächsten Abschnitte gegeben, die die einzelnen Ebenen des Modells sowie die spezifische Ausprägung für das MHS-Management beschreiben.

- Application Entity

  Auf dieser Ebene werden die Aspekte der Komponenten beschrieben, aus denen sich die verteilte Anwendung zusammensetzt. Im Fall des X.400-MHS handelt es sich hier um MTA, MS, UA und verschiedene Typen von AUs. Im Anschluß werden einige Kontrollobjekte beschrieben, mit deren Hilfe Zusatzfunktionalität verwaltet wird.

- Application

  Diese Ebene beschreibt anwendungsglobale Managementaspekte. Am konkreten Beispiel werden Konzepte für Adreßraum-, Topologie- und Konfigurationsverwaltung vorgestellt.

- Service

  Diese Ebene beschäftigt sich mit den Diensten, die eine verteilte Anwendung erbringt. Zu diesem Zweck wird zunächst ein Modell für eine Dienstehierarchie entworfen, das anschließend mit den vom MHS angebotenen Diensten gefüllt wird.

- Business

  Diese Ebene bildet die Schnittstelle zum Enterprisemanagement. Aus diesem Grund wird zunächst ein System zur Beschreibung von betrieblichen Arbeitsabläufen vorgestellt, um anschließend den Zusammenhang zum Anwendungsmanagement zu erläutern. Abschließend wird ein Modell für die Überwachung betriebswirtschaftlicher Rahmenbedingungen beschrieben.

3.4.2 Application-Entity-Ebene

Auf dieser Ebene werden die einzelnen Komponenten, aus denen sich eine verteilte Anwendung zusammensetzt, betrachtet. Innerhalb des Managementmodells werden alle Aspekte modelliert, die diese Komponenten unmittelbar betreffen.

3.4.2.1 Überblick

Die verschiedenen Typen von Komponenten eines MHS sind in Abschnitt 2.2.1 aufgelistet. Es handelt sich um Message Transfer Agents, Message Stores, User Agents und verschiedene Arten von Access Units.

Für jeden dieser Typen wird eine Klasse definiert. Die Objekte innerhalb des Managementsystems, die die Komponenten des MHS repräsentieren, sind Instanzen der jeweiligen Klassen.

Es bietet sich an, diejenigen Managementinformationen, die allen Komponenten gemeinsam ist, in eine eigene Klasse einzubetten, und alle komponentenspezifischen Klassen von dieser abzuleiten (vgl. Abbildung 3.2).

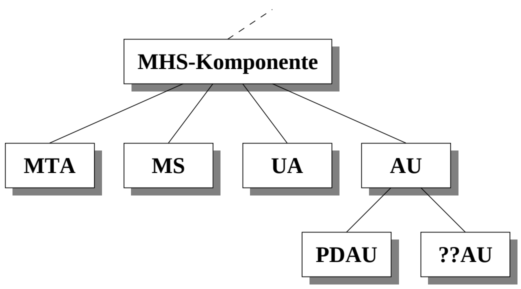


Abbildung 3.2: Klassen auf der Application-Entity-Ebene

3.4.2.2 MHS-Komponenten im allgemeinen

In dieser Klasse werden die Aspekte modelliert, die allen MHS-Komponenten gemeinsam sind. Es ist auch denkbar, die Hierarchie nach oben hin zu erweitern, etwa um eine Klasse für netznahe verteilte Applikationen und weiter um eine Klasse für allgemeine verteilten Anwendungen.

Konfiguration In diesen Bereich fallen der Name, die O/R Address, soweit sie zugeordnet werden kann, und gegebenenfalls der Directory Name.

Falls sich die jeweilige Komponente aus mehreren Bausteinen zusammensetzt, werden diese hier aufgeführt. Desweiteren existiert ein Verweis auf den Rechner, auf dem die Applikation abläuft sowie ggf. die Person, die für die Verwaltung dieser Komponente zuständig ist.

Verbindungsdaten Für jeden Link, den die Komponente innerhalb des MHS unterhält, existiert ein Eintrag, der den Namen und die Netzadresse der Partnerinstanz beinhaltet. Außerdem werden hier entweder der Directory Name des Partners oder Informationen über dessen Fähigkeiten und den gegenseitigen Authentifizierungsablauf festgelegt. Weiterhin existiert ein Eintrag für den Status des Links (Verbindung auf-/abgebaut, Fehler), sowie verschiedene Statistiken über Auslastung und Verkehr auf dem Link, etwa Zähler für die Anzahl und Volumen der transportierten Nachrichten sowie der fehlgeschlagenen Übermittlungsversuche.

Systemdaten Desweiteren wird festgehalten, welche Systemressourcen der Komponente bzw. ihren Bausteinen zugeordnet sind. In diesen Bereich fallen Verweise auf zugrundeliegende Prozesse, verwendete Filesysteme und so fort.

3.4.2.3 Aspekte eines Message Transfer Agent

Ein MTA bietet eine Reihe weiterer Managementinformationen:

- Routinginformationen, mit denen einer Nachricht an Hand der Empfängeradresse eine Partnerinstanz zugeordnet wird, an die sie weiterzuleiten ist.
- Zusätzliche Konfigurationsinformation, wie Timeouts, maximale Anzahl der durchlaufenen Zwischenstationen, bevor eine Nachricht verworfen wird und maximal erlaubte Nachrichtengröße.
- Auslastungsanzeigen wie Füllstand der Nachrichtenwarteschlage, Verweildauer der Nachrichten im MTA und Durchsatz von Nachrichten durch den MTA.

3.4.2.4 Aspekte eines Message Store

Informationen, die der Message Store zur Verfügung stellt, sind

- Liste der Anwender, die von diesem MS versorgt werden

- Füllstand des Nachrichtenarchivs
- Anzahl und Volumen der zwischengespeicherten Nachrichten

3.4.2.5 Aspekte eines User Agent

Der User Agent ist die einzige Komponente im MHS, mit der der Endanwender unmittelbar kommuniziert. Insbesondere ist daher von Interesse, welche Systemressourcen für die Bedienoberfläche benutzt werden.

3.4.2.6 Aspekte einer Access Unit

Da eine große Zahl von (zum Teil proprietären) MHS-Technologien existiert und dementsprechend eine Vielzahl verschiedener Typen von Access Units denkbar sind, ist es nicht sinnvoll, für deren Modellierung nur eine einzige Klasse zu verwenden. Daher wird hier eine eigene Hierarchie von Klassen aufgespannt. Managementinformationen, die die Access Units bereitstellen, sind etwa die Parameter der Funktionen, die die Abbildung von Adressen und Datenformaten auf die Struktur des jeweiligen Fremdnetzes vornehmen.

3.4.2.7 Kontrollobjekte

Es ist abgebracht, neben den im letzten Abschnitt vorgestellten, MHS-spezifischen Objekten eine Reihe weiterer Objekte zu definieren, die für die Kontrolle verschiedener erweiterter Funktionen zuständig sind. Der Katalog der im folgenden vorgestellten Objekte basiert auf den Systems Management Functions ([ISO10164]).

- Alarm und Event Reporting

  Es wird ein Kontrollobjekt angelegt, das entscheidet, welche Ereignisse innerhalb einer Komponente als Event (Ereignishinweis) oder als Alarm an den Manager versandt werden.

  - Eine Anwendung für diesen Bereich sind Alarme, die beim Auftreten von Fehlern ausgelöst werden.
  - Ein Beispiel, das in den Bereich Leistungsmanagement fällt, ist die Schwellwertüberwachung. Dabei werden für bestimmte Auslastungsattribute Schwellwerte und Rücksetzwerte definiert. Überschreitet ein Attributwert den zugehörigen Schwellwert, wird ein Alarm ausgelöst, der beim Unterschreiten des Rücksetzwerts wieder gelöscht wird.

  - Außerdem fällt in diesen Bereich auch der Versand von Alarmen, die beim Auftreten von sicherheitsrelevanten Ereignissen generiert werden, was beispielsweise geschieht, wenn die Integrität der Nachricht nicht garantiert werden kann oder ein unerlaubter Managementzugriff stattfindet.

- Logging

  Log-Objekte kontrollieren, welche sich dynamisch ändernden Informationen und welche Ereignisse protokolliert werden. Gegebenenfalls bieten sie auch eine Schnittstelle zum entfernten Zugriff auf die Protokolldateien. Auch hier sind mehrere Anwendungen denkbar.

  - Um langfristige Planung zu ermöglichen, werden verschiedene Verkehrs-, Auslastungs- und Fehlerdaten mitgeschrieben. Auf Basis dieser Informationen können dann strategische Entscheidungen bzgl. des weiteren Ausbaus getroffen werden.
  - Um die Ursachen einer Sicherheitsverletzung ermitteln zu können, werden eine Reihe von möglicherweise sicherheitsrelevanten Informationen, etwa Verbindungsauf- und -abbau und Managementzugriffe mitprotokolliert.
  - Um das Umlegen der entstehenden Kosten auf die einzelnen Anwender zu ermöglichen, werden die dafür notwendigen Daten (Absender, Empfänger, Größe, ...) jeder bearbeiteten Nachricht mitprotokolliert.

- Zugangskontrolle

  Hier wird definiert, welchen Managern welche Rechte beim Managementzugriff auf die überwachte Anwendung zugeordnet werden.

3.4.3 Application-Ebene

In diesem Abschnitt werden die Informationen, die die einzelnen Komponenten auf der Application-Entity-Ebene zur Verfügung stellen, zusammengefaßt, miteinander in Verbindung gebracht und in einen applikationsweiten Kontext gestellt. Hauptgesichtspunkt auf dieser Ebene sind die Managementaspekte, die die verteilte Anwendung als Ganzes betreffen.

Da ein X.400-MHS keine verteilte Anwendung in dem Sinn ist, daß die einzelnen Komponenten eng zusammenarbeiten und sich stark aufeinander abstützen, sondern es sich eher um ein loses Netz von kooperierenden Bausteinen handelt, gibt es auf dieser Ebene nur einige wenige MHS-weite Aspekte.

3.4.3.1 Organisation des Adreßraums

Wie bereits in Abschnitt 2.2.2 aufgezeigt wurde, zerteilt sich ein X.400-MHS in verschiedene Administration und Private Management Domains. Das in Abschnitt 2.3.1 beschriebene Adreßformat läßt eine feinere Gliederung in kleinere Bereiche (Organization, Organizational Unit; letztere auch mehrfach) zu.

Zwar schreiben die Standarddokumente keine Struktur vor, faktisch wird allerdings immer eine hierarchische Gliederung verwendet. Es bietet sich daher an, diese Gliederung in das Managementmodell zu übernehmen.

Dazu wird eine Klasse MHSOrganization definiert. Für jede Administration und Private Management Domain, jede Organization und Organizational Unit sowie weitere Gliederungsbereiche, die MD-privat definiert sind, wird ein entsprechendes Objekt dieser Klasse angelegt. Diese Objekte und die Objekte, die die MHS-Komponenten repräsentieren, werden durch Relationen in eine hierarchische Struktur eingepaßt, die der Organisation des Adreßraums entspricht.

Da die Klasse MHSOrganization hauptsächlich der Beschreibung und Verwaltung der MHS-Struktur dient, enthält sie den Namen des Organisationsbereichs und Kontaktadressen des Verwalters dieses Bereichs.

Zusätzliche Funktionalität:

- Es sollte möglich sein, aus den O/R-Address-Attributen der einzelnen Komponenten automatisch die Adreßstruktur des Netzes zu ermitteln und den Organisationsbaum zu erzeugen.

- Wird eine Komponente in einen anderen Organisationsbereich verschoben, muß die O/R Address dieser Komponenten entsprechend angepaßt werden. Gegebenenfalls müssen weitere Modifikationen an der Konfiguration verschiedener Komponenten vorgenommen werden (siehe Abschnitt 3.4.3.3).

3.4.3.2 Topologie des Netzes

Ein weiterer Aspekt eines MHS ist die Topologie, nach der die Komponenten im MHS verbunden sind.

Um dies zu modellieren, wird eine Linkklasse definiert. Sämtliche Verbindungen, die im MHS existieren, werden durch Objekte dieser Klasse repräsentiert.

Linkobjekte enthalten

- eine eindeutige Bezeichnung für den Link, Verweise auf die beiden Komponenten an den Endpunkten des Links sowie Authentifizierungsinformation für den Verbindungsaufbau.
- Status und Belastung des Links sowie, falls die Verbindung aufgebaut ist, Verweise auf die zugrunde liegenden Netzmanagementobjekte.
- zusätzliche Daten wie Kostenbewertung.

Zusätzliche Funktionalität:

- Die bestehende Topologie des MHS wird aus den Linkdaten, die in den einzelnen Komponenten abgelegt sind, ermittelt.
- Wird ein Linkobjekt kreiert, modifiziert oder gelöscht, werden in den beiden zugehörigen Komponenten die entsprechenden Linkinformationen angepaßt und umgekehrt.
- Beim Entfernen eines Links müssen gegebenenfalls weitere Modifikationen vorgenommen werden (siehe Abschnitt 3.4.3.3).

3.4.3.3 Konsistenz der Komponentenkonfigurationen

Ein MHS besteht aus einer größeren Anzahl von Komponenten. Diejenige Teilmenge der Konfigurationsdaten, die das Verhalten des gesamten Verbundes beeinflußt, muß auf allen Komponenten aneinander angeglichen werden.

Insbesondere die Routingtabellen der einzelnen MTAs müssen in ihrem Zusammenspiel überwacht werden. Nachrichten müssen von jedem Punkt des MHS an jeden anderen Punkt gelangen. Mögliche Fehlersymptome sind Nachrichtenverlust, wenn eine Nachricht auf einem MTA ankommt, der keinen Eintrag für die Zieladresse besitzt, und Message Loops, bei denen die Nachrichten zwischen mehreren MTAs im Kreis herum wandert.

Hier setzen mehrere Funktionen an:

- Die bestehenden Routingtabellen werden ausgewertet und auf Korrektheit geprüft. Hier finden Algorithmen zur Zyklenerkennung und zur Bildung transitiver Hüllen Anwendung.

- Die Routingtabellen werden automatisch an Hand der Topologieinformation (Abschnitt 3.4.3.2) erzeugt. Hier können Verfahren zur Bestimmung des kürzesten Pfades verwendet werden, wobei die Kanten (Links) oder die Knoten (Komponenten) oder beide bewertet werden. Der Bewertungsfunktion können viele Kriterien zu Grunde liegen. Im allgemeinen ergeben sich diese aus den Betriebsrandbedingungen, die in Abschnitt 3.4.5.3 vorgestellt werden.

Werden Links erzeugt oder entfernt oder Komponenten innerhalb der Organisationsstruktur verschoben, kann mit Hilfe des zweiten Algorithmus automatisch ein Angleich der Routinginformation erfolgen. Mit diesem Verfahren kann auch auf Ausfall oder Überlastung von Komponenten oder Verbindungen reagiert werden.

3.4.4 Service-Ebene

Ziel der Diensteebene ist es, von der verteilten Anwendung bzw. deren Komponenten zu abstrahieren und Managementinformationen für die angebotenen Dienste, die ja der eigentliche Beweggrund für den Einsatz der jeweiligen Anwendung sind, zu ermitteln.

3.4.4.1 Ein Modell für eine Dienstehierarchie

In diesem Modell werden die Dienste beschrieben, auf denen die verteilte Anwendung aufsetzt bzw. die sie selbst erbringt. In Abbildung 3.3 ist eine Gliederung für eine solche Hierarchie dargestellt.

Die verteilte Anwendung setzt auf einer größeren Menge von Diensten auf, die im folgenden beschrieben werden:

- Auf unterster Ebene stehen die Dienste, die Netz und System zur Verfügung stellen.
- Darauf setzen Kommunikationsdienste auf, die beispielsweise den Austausch von Nachrichten oder das entfernte Ausführen von Funktionen (RPC) erlauben.
- Die dritte Ebene beinhaltet grundlegende verteilte Dienste, die einerseits für die Nutzung der Kommunikationsdienste hilfreich sind, andererseits aber auch auf diesen aufsetzen. Ein Beispiel für einen solchen Dienst ist die Abbildung von Systemnamen auf Netzadressen.

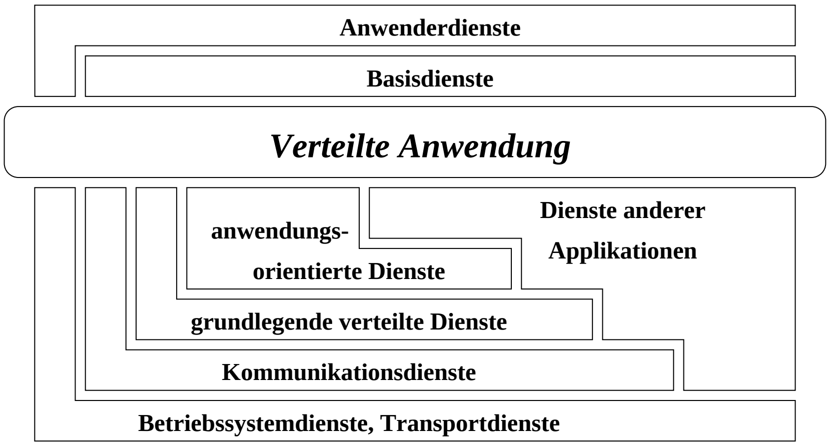


Abbildung 3.3: Dienstehierarchie (zum Teil aus [HNG94])

- Applikationsorientierte Dienste sind die Funktionen, die von einer größeren Zahl von Anwendungen benötigt werden. Ziel ist es, solche Softwarebausteine wiederzuverwenden, um die Erstellung von Applikationen zu vereinfachen.

- Es ist denkbar, daß eine Anwendung zusätzlich zu diesen elementaren Diensten die komplexen Dienste anderer Applikationen verwendet. Als Beispiel können hier die Dienste von Datenbanksystemen dienen, derer sich eine große Zahl von Applikationen bedienen, oder auch der Electronic Data Interchange Service eines MHS.

Die verteilte Anwendung setzt auf dieser Dienstehierarchie auf und bietet ihrerseits dem Anwender verschiedene Dienste an. Je nach Komplexität der Applikation können auch diese in eine Hierarchie eingeordnet werden. Eine mögliche Gliederung ist:

- Die Komponenten der verteilten Anwendung bieten verschiedene Basisdienste an.

- Auf diese Basisfunktionen stützen sich die komplexen Dienste, die dem Endanwender angeboten werden, ab.

Bei großen komplexen Anwendungen ist auch eine mehrstufige Hierarchie denkbar.

Die Objekte der Service-Ebene haben die Aufgabe, diese Dienstehierarchie darzustellen und die Abhängigkeiten der Dienste innerhalb dieser Struktur bis hinunter zu den Netz- und Systemdiensten zu erfassen. Desweiteren muß an den entsprechenden Übergängen ein Bezug zu denjenigen Ressourcen der unteren TMN-Ebenen hergestellt werden, die für die Bereitstellung dieser Dienste verantwortlich sind.

Weiterhin müssen die Dienstobjekte die Managementinformationen der unteren Ebenen erfassen, korrelieren und abstrahieren, um zu entsprechenden Daten über den repräsentierten Dienst zu gelangen.

Entsprechend müssen Dienstobjekte folgende Information und Funktionalität bereitstellen:

- Postition in der Hierarchie

  Jedes Objekt beschreibt, auf welchen in der Hierarchie tieferstehenden Diensten der von ihm repräsentierte Dienst aufsetzt und welche Komponenten unmittelbar diesen Dienst erbringen.

  Im allgemeinen wird es genügen, die zugrundeliegenden Dienste sowie die zugeordneten Komponenten aufzulisten. Bei komplexen verteilten Anwendungen ist aber denkbar, das diese Beschreibungsform nicht ausreicht. Hier kann es notwendig werden, zusätzlich zeitliche und kausale Zusammenhänge zu beschreiben.

- Verfügbarkeit und Leistung

  An Hand dieser Abhängigkeitsbeschreibung kann das Dienstobjekt aus den Verfügbarkeits- und Leistungsdaten der zugrundeliegenden Dienste und zugeordneten Komponenten entsprechende Availability- und Performace-Daten für den von ihm repräsentierten Dienst ermitteln.

- Weitere Managementbereiche

  Je nach Managementanforderung kann es auch notwendig sein, Daten anderer Managementbereiche, etwa Konfiguration und Sicherheit, von der Applikationsebene auf die Diensteebene zu abstrahieren. Umgekehrt ist es auch denkbar, auf Diensteebene Einstellungen vorzunehmen, die mit Hilfe der Hierarchiebeschreibung auf die Applikation bzw. ihre Komponenten abgebildet werden.

3.4.4.2 Dienstehierarchie im MHS

Im Kontext von X.400-Management werden im folgenden die Dienste der oberen Ebenen, die dem Benutzer vom MHS angeboten werden, betrachtet.

Das Diensteangebot eines MHS läßt sich in zwei Ebenen aufgliedern (vgl. Abbildung 3.4):

- Die Basisdienste, die die einzelnen Komponenten anbieten, sind größtenteils bereits in Abschnitt 2.4.1 beschrieben worden. Hier handelt es sich um die verschiedenen Transportdienste sowie die Funktionen, die der MTA ausführt. Dieser Katalog kann um (nicht genormte) Dienste wie das Zwischenspeichen von Nachrichten erweitert werden.

- An Anwenderdiensten bietet das MHS nur einen Dienst an: „Transport einer Nachricht von User A zu User B"

  Je nach Managementanforderungen kann dieser Dienst durch ein einziges Objekt, das den gesamten Transport-Dienst repräsentiert, oder durch eine größere Zahl von Objekten, die bestimmte Einschränkungen bezüglich Absender und Empfänger (etwa auf PRMDs, Abteilungen, einzelne Anwender) definieren, modelliert werden.

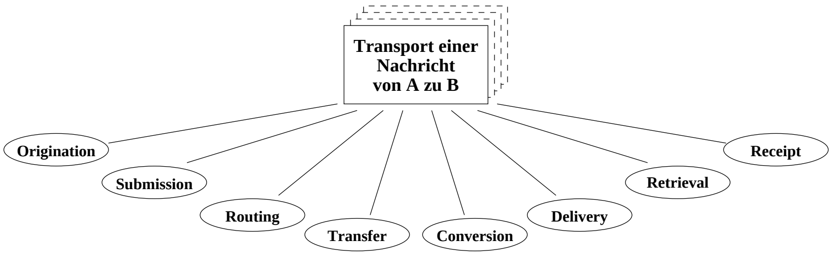


Abbildung 3.4: Dienstehierarchie eines X.400-MHS

[X402] bietet einen Vorschlag für die Implementierung von X.400-Komponenten an, der fünf ASEs, namentlich Message Transfer, Message Submission, Message Retrieval, Message Delivery und Message Administration Service Element, vorsieht. Die ersten vier ASEs dienen zur Unterstützung der entsprechenden Transmittal Steps und bedienen sich dazu des MASEs, das Dienste für X.400-Verbindungsaufbau und -kontrolle enthält. Diese Dienste werden in die Ebene der anwendungsorientierten Dienste eingeordnet. Wurde diese Richtlinie bei der Implementierung berücksichtigt, ist die Hierarchie aus Abbildung 3.4 entsprechend zu erweitern.

Die unteren drei Diensteebenen aus Abbildung 3.3 sind den Netz- und Systemmanagementszenarien zuzuordnen, wobei natürlich innerhalb der X.400-Basisdienstobjekte Verweise auf diese Dienste existieren.

Außerdem müssen auch die Dienste des Verzeichnissystems, auf die sich ein großer Teil der MHS-Funktionen abstützen, in dieses Modell mit eingebunden werden. Auch hier werden in den entsprechenden Objekten Verweise vorgenommen.

3.4.5 Business-Ebene

Dieser Abschnitt behandelt die Eingliederung des MHS in den betriebswirtschaftlichen Kontext, in dem es genutzt wird. Aus diesem Grund wird im nächsten Abschnitt zunächst ein kleiner Exkurs in die Aufgabenstellung Enterprisemanagement unternommen, um anschließend den Bezug zum Anwendungsmanagement herzustellen und ein Modell für die betriebswirtschaftlichen Randbedingungen, die an eine Applikation gestellt werden, vorzustellen.

3.4.5.1 Grundlagen des Enterprisemanagements

Es gibt zwei Hauptgesichtspunkte, die die Struktur eines Unternehmens beschreiben:

- Die Aufbauorganisation beschreibt, wie sich das Unternehmen gliedert, welche Unternehmensbereiche existieren und welche Abteilungen diesen zugeordnet sind. Diese Struktur wird üblicherweise in einem Organigramm dargestellt.

- Die Ablauforganisation beschreibt, welche Vorgänge innerhalb des Unternehmens ablaufen und wie sie gegliedert und strukturiert sind. Im folgenden wird ein Modell für eine solche Beschreibung vorgestellt.

Um die Abläufe innerhalb eines Unternehmen erfassen und insbesondere elektronisch bearbeiten und überwachen zu können, wurden in den letzten Jahren sogenannte Workflowmanagementsysteme entwickelt.

Das Modell, von diese Produkte ausgehen, beschreibt alle betriebswirtschaftlich relevanten Vorgänge durch sogenannte Workflows. Dabei wird zwischen elementaren Workflows — einfachen Aufgaben, die in einem Arbeitsschritt ausgeführt werden können — und komplexen Workflows, die sich aus mehreren untergeordneten elementaren oder komplexen Workflows zusammensetzen, unterschieden. Mit diesem Instrumentarium können somit alle Abläufe innerhalb des Unternehmens in ein hierarchisch strukturiertes Modell abgebildet werden.

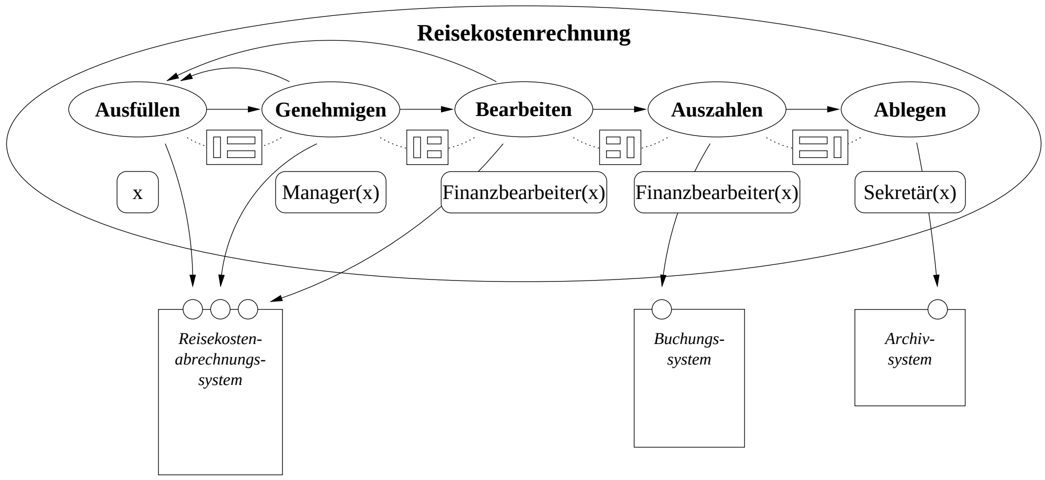


Abbildung 3.5: Beispiel eines Workflows (aus [Jab95])

In Abbildung 3.5 ist ein Beispiel aus [Jab95] dargestellt, das den komplexen Workflow „Reisekostenabrechnung“ in fünf elementare Workflows zerlegt.

In diesem Modell sind eine Reihe von Daten beschrieben:

- Workflows

  Ein komplexer Workflow gliedert sich in mehrere elementare oder komplexe Workflows. Das Modell beschreibt, welche zeitlichen und kausalen Abhängigkeiten zwischen diesen Workflows bestehen. Im Beispiel laufen die einzelnen Teilschritte sequentiell mit Verzweigungsmöglichkeit ab. Denkbar sind auch Workflows, deren Teilschritte parallel ablaufen, oder Kombinationen aus beiden Konzepten.

- Aktoren

  Jedem Arbeitsschritte wird eine Rolle zugeordnet. Die Person, die diesen Arbeitsschritt ausführt, muß dieser Rolle entsprechen. Im Beispiel kann nur ein Manager eine Reisekostenabrechnung genehmigen, während der Arbeitsschritt „Ausfüllen“ keine besonderen Kompetenzen voraussetzt.

  Da allerdings nicht jede Person im Unternehmen, die die Rolle Manager bekleidet, jede Reisekostenabrechnung genehmigen kann, ist im Modell eine Einschränkung definiert. Wird der Antrag von Mitarbeiter x ausgefüllt, kann er nur von einem Manager genehmigt werden, der für Mitarbeiter x zuständig ist, was durch die Klausel *Manager(x)* — auch komplexere Definitionen sind denkbar — zum Ausdruck kommt.

Diese Zuordnung wird aus der Aufbauorganisation des Unternehmens abgeleitet.

- Daten

  Daten werden zwischen Workflows ausgetauscht, um den Kontext, in dem ein Workflow ausgeführt wird, zu umschreiben. Beispielsweise ist der Name des Antragstellers in Abbildung 3.5 ein Datum, das von jedem Teilschritt benötigt wird. Auch die Entscheidung, welcher Weg an einer Verzweigung eingeschlagen wird, hängt von diesen Daten ab.

- Applikationen

  Im allgemeinen setzt ein elementarer Workflow in seinem Ablauf die Verwendung eines oder mehrerer Werkzeuge voraus. Im Beispiel sind das etwa die Applikationen „Reisekostenabrechnungssystem", „Buchungssystem" und „Archivsystem". Die Person, die den jeweiligen Workflow bearbeitet, benötigt dazu die Dienste der jeweiligen Anwendung.

3.4.5.2 Beziehung zum Anwendungsmanagement

Wie im letzten Abschnitt bereits anklingt, sind Applikationen ebenso wie Systeme und Netze in betriebliche Abläufe eingebunden, aus denen sich ableitet, welche Aufgaben jede einzelne Ressource zu erfüllen hat und welche Dienste sie anbieten muß. Daher ergibt sich aus diesem Zusammenhängen erst die eigentliche Definition derjenigen Dienste, die die verteilten Anwendung erbringen muß und die daher auf Service-Ebene überwacht werden müssen.

Wie das Reisekostenabrechnungssystem ist auch das MHS eine Applikation, die innerhalb eines Workflow Anwendung finden kann. Beispiel einer Aktion könnte der Versand einer Auftragsbestätigung an einen Kunden sein. Zudem werden die Dienste des MHS auch vom Workflow Management System selbst benutzt, um etwa die Daten zwischen den einzelnen Schritten zu übertragen oder einen Aktor über einen anstehenden Arbeitsschritt zu informieren.

Aus den Unternehmensabläufen läßt sich auch ableiten, ob ein Serviceobjekt für das gesamte MHS ausreicht, oder ob für spezielle Absender-Empfänger-Kombinationen zusätzliche Objekte angelegt werden müssen.

3.4.5.3 Policies

Inhalt der Business-Ebene sind die Randbedingungen für den Betrieb der verteilten Anwendung, die sich aus dem betriebswirtschaftlichen Umfeld ableiten. Im folgenden werden einige Beispiele für solche Policies beschrieben.

Quality of Service Aus dem Kontext der Arbeitsabläufe ergeben sich verschiedene Randbedingungen, die die angebotenen Dienste erfüllen müssen. Im allgemeinen handelt es sich hier um Vorgaben bzgl. Leistung und Verfügbarkeit eines Dienstes. Tritt eine Verletzung einer solchen Vorgabe auf, ist die Managementstation zu benachrichtigen. Gegebenenfalls können in diesem Fall automatische Korrekturabläufe angestoßen werden.

Zwei Strategien sind denkbar:

- Die Qualitätsvorgabe wird die Dienstehierarchie hinab zergliedert, so daß schließlich Anforderungen für die einzelnen Komponenten abgeleitet werden können. Diese werden mit Hilfe von Schwellwertobjekten, wie sie in Abschnitt 3.4.2.7 beschrieben sind, überwacht.
- Die QoS-Vorgaben werden an Hand der in den Serviceobjekten bereitgestellten Performancedaten überwacht.

In der Anwendung MHS darf insbesondere die Verzögerung zwischen Erstellung der Nachricht beim Absender und Verfügbarkeit beim Empfänger einen vorzugebenden Rahmen nicht verlassen. Als Korrektur ist beispielsweise eine Neuberechnung der Routingtabellen mit angepaßten Gewichtungen (Abschnitt 3.4.3.3) oder das Anlegen zusätzlicher Links denkbar.

Accounting Innerhalb einer Accounting-Policy wird definiert, welche Dienste abgerechnet werden, welches Abrechnungsverfahren verwendet wird, und so fort.

Das Policy-Objekt bestimmt aus dieser Definition die relevanten Daten, die protokolliert werden müssen und konfiguriert und aktiviert in den einzelnen Komponenten die entsprechenden Accounting-Logobjekte.

Im MHS kann der Übermittlungsdienst abgerechnet werden, wobei verschiedene Policies denkbar sein. Aspekte sind

- Abrechung im Gesamtnetz oder nur an Übergängen in andere Management Domains?
- Kostenaufstellung nach Nachrichtenanzahl oder -volumen?
- Welchen Einfluß hat die Absender- und Empfängeradresse auf die Preisbildung?

Priorität Die verschiedenen Workflows sind innerhalb des Unternehmens unterschiedlich wichtig. So hat eine Verzögerung in einem Geschäftsablauf größere Auswirkungen als in der beschriebenen Reisekostenabrechnung. Aus diesen Vorgaben läßt sich auch die Wichtigkeit von Diensten, Applikationen und Komponenten ableiten, was in einer Priorisierung der Managementaufgaben resultieren kann.

3.4.6 Implementierungsüberlegungen

In diesem Abschnitt soll betrachtet werden, wo innerhalb des Managementszenarios welche Mangementinformation und -funktionalität implementiert wird.

Die Objekte der Application-Entity-Ebene beziehen sich ausschließlich auf einzelne Softwarekomponenten, die jeweils vollständig einem Rechensystem zugeordnet werden können. Somit kann hier das klassische Manager-Agent-Szenario verwendet werden, in dem die Managementinformation im Agenten abgelegt ist.

Ein X.400-MHS stellt insofern eine Ausnahme von dieser Regel dar, da Konfigurationsinformation zum Teil bereits in einem anderem Informationssystem, nämlich dem X.500-Directory (siehe auch Abschnitt 2.5), verfügbar ist. Es liegt daher nahe, von der Managementstation direkt auf das Verzeichnissystem zuzugreifen, wobei diese Vorgehensweise für die Managementanwendungen transparent bleiben sollte.

Die Objekte der oberen drei Ebenen sind in zunehmendem Maße abstrakter und lassen sich nicht mehr einem einzelnen Agentensystem zuordnen. Diese Objekte und ihre Funktionalität werden daher in der Middleware der Managementplattform implementiert. Auch hier ist es denkbar, das Directory als Objektspeicher einzubeziehen.

Schließlich sollte die Managementplattform auch einen Übergang zum Workflow Management System besitzen, um einerseits Vorgabewerte für die verschiedenen Policies zu erhalten und andererseits Status-, Fehler- und Leistungsinformation zu übermitteln. Das Workflow Management System könnte auf Basis dieser Daten etwa Rückschlüsse auf die Effizienz der Unternehmensabläufe ziehen.

Abbildung 3.6 zeigt einen Überblick über das beschriebene Szenario.

Anmerkung: Zur Zeit werden Konzepte erarbeitet, die eine weitergehende Integration eines Directorys in ein Managementsystem erlauben. Für eine detaillierte Beschreibung dieser Problematik sei auf [MTW93] und [HBB93] verwiesen.

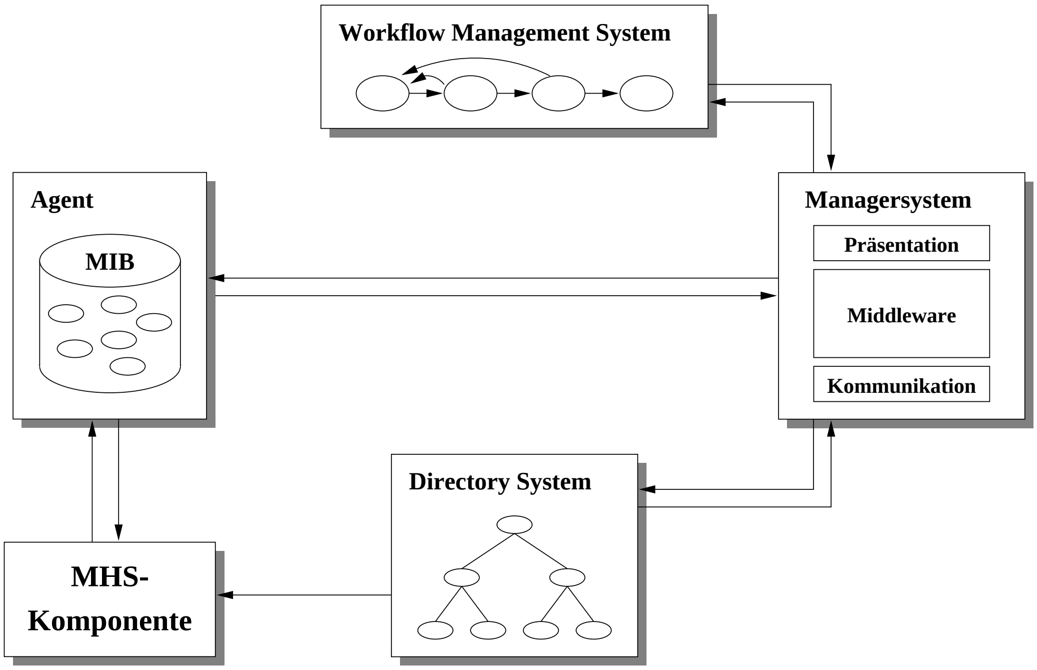


Abbildung 3.6: Parteien im Managementszenario

Kapitel 4

Bestehende Managementansätze

In diesem Kapitel werden bestehende Managementansätze untersucht, wobei sowohl proprietäre Lösungen wie auch herstellerübergreifende Konzepte berücksichtigt werden.

4.1 Herstellerspezifische Ansätze

Um den Kunden die Möglichkeit zu geben, ihre X.400-Komponenten einfach zu konfigurieren und adäquat zu überwachen, befinden sich im Lieferumfang der meisten Produkte bereits eine Reihe von proprietären Managementtools, die im allgemeinen die Bereiche Konfiguration und Überwachung abdecken.

Diese Werkzeuge können grob in zwei Klassen gegliedert werden:

- Tools, die lokal auf dem Rechner, auf dem die X.400-Komponente installiert ist, zu verwenden sind.
- Tools, die Zugriff auch über Systemgrenzen hinweg erlauben und damit eine Fernüberwachung und -verwaltung möglich machen.

Die Informationen der folgenden Abschnitte stammen größtenteils aus den Antworten auf eine Anfrage, die im März 1995 an verschiedene Hersteller von X.400-Produkten versandt wurde. Nexors XT-PP wird im folgenden nicht berücksichtigt, dieser MTA wird in Abschnitt 6.1 beschrieben.

4.1.1 Werkzeuge für lokales Management

Üblicherweise wird die Funktion eines MTA durch eine größere Anzahl von Konfigurationsdateien bestimmt. Viele Hersteller liefern daher Tools aus, die

die Erstellung dieser Dateien erleichtern. So ist im Lieferumfang des Message Transfer Agent EAN, der von der GMD vertrieben wird, ein Werkzeug zur Konfiguration von Partner-MTAs, Queues, lokalen Usern und Routing-Tabellen enthalten. Auch für den AlisaMail Information Switch, der auf einem relationalen Datenbanksystem aufsetzt, existieren entsprechende Werkzeuge.

Ein weiteres Einsatzgebiet lokaler Managementwerkzeuge ist die Überwachung. Der MTA NAR400 der Universität Madrid realisiert diese Funktion durch das Einbinden spezieller Management-Funktionsgruppen.

4.1.2 Werkzeuge für entferntes Management

Gerade für den Betrieb eines größeren MHS ist es notwendig, daß die verteilten Komponenten des Systems von einer zentralen Stelle überwacht und gegebenenfalls konfiguriert werden können.

Für den ISOPLEX 4 von ISOCOR existiert ein Management Center, das die Überwachung mehrerer dieser MTAs zuläßt. Desweiteren ist es möglich, Änderungen an der Konfiguration vorzunehmen, etwa Adressen ein- und auszutragen. Diese Managementsoftware wird auf einem Windows-PC installiert und kommuniziert mit den einzelnen MTAs mittels eines proprietären Protokolls, das auf TCP/IP oder X.25 aufsetzt.

Einen Schritt weiter geht Control Datas Advanced Operation Manager für das Produkt Mail*Hub. Hier wird als Kommunikationsprotokoll SNMP verwendet, wobei neben einer privaten MIB auch die in den Abschnitten 4.3.1 und 4.3.2 beschriebenen Standard-MIBs unterstützt werden. Mit dem Operation Manager können zum einen eine Reihe von Statistikdaten wie Queuefüllung und Durchsatz dargestellt werden, auch das Anzeigen der von Mail*Hub geführten Logfiles, etwa um den Weg zu verfolgen, den eine Nachricht zurückgelegt hat, ist möglich. Zum anderen kann der Manager auch direkt in den MTA eingreifen, um neue Queues anzulegen oder umzukonfigurieren oder verschiedene Tests durchzuführen. Um die Netzlast gering zu halten, können im Agenten Ereignisse definiert werden, bei deren Auftreten der Manager durch einen Alarm benachrichtigt wird. Bemerkenswert ist, daß der Operation Manager auch direkten Zugriff auf das X.500-Directory erlaubt, in dem die Routing- und Adreßtabellen des Mail*Hub abgelegt sind.

4.2 Ein Beispiel für einen herstellerübergreifenden Ansatz

In diesem Abschnitt soll ein Ansatz vorgestellt werden, der zur Zeit an der Universität Genf verfolgt wird ([HPBS94]). Ziel dieser Arbeit ist es, die Managementproblematik von verteilten Anwendungen am Beispiel von EMail-Systemen zu untersuchen und daraus Strategien und Technologien für eine allgemeine Applikationsmanagementlösung zu finden.

Ein zentralistischer Ansatz mit einer Managementstation, die unmittelbar alle Anwendungskomponenten überwacht, wurde verworfen, da zu viel Netzlast erzeugt wird, das Konzept nicht skalierbar ist und die Managementstation die verschiedensten Komponenten unmittelbar unterstützen muß.

Statt dessen wurde ein verteilter Ansatz gewählt, der skalierbar ist, eine Delegation der Managementaufgaben ermöglicht und toleranter auf Ausfälle einzelner Komponenten reagiert.

Als Struktur wurde ein hierarchisches Netz von Managementknoten gewählt, der „Distributed Management Tree" (vgl. Abbildung 4.1).

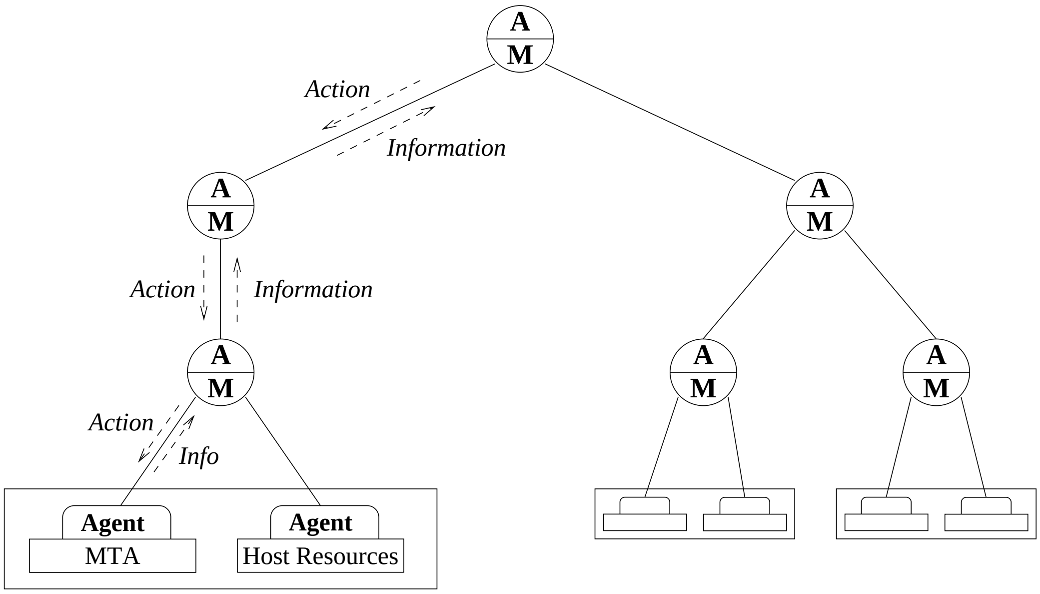


Abbildung 4.1: Der Distributed Management Tree (aus [HPBS94])

Jeder innere Knoten im Baum agiert seinem Vater gegenüber als Agent, seinen Söhnen gegenüber dagegen als Manager. Informationen, die von den Blättern den Baum nach oben durchlaufen, werden in den einzelnen Knoten gesammelt und verdichtet, während Aktionsaufforderungen, die im Baum von oben nach unten weitergereicht werden, für die einzelnen Söhne expandiert werden. Die Kriterien, nach denen dies geschieht, unterscheiden sich von Ebene zu Ebene:

- Für jedes System, auf dem ein MTA installiert ist, existieren zwei Blätter im Baum, zum einen der zum MTA gehörige Agent, zum anderen ein Agent, der die Systemressourcen überwacht. Diese Agenten können einem gängigen Standard entsprechen, etwa den in den Abschnitten 4.3.1 und 4.3.2 beschriebenen MIBs bzw. der Host Ressources MIB ([RFC1514]), es ist aber auch möglich, daß ein solcher Agent ein proprietäres Managementverfahren implementiert.

- Auf der nächsthöheren Ebene werden die MTA- und System-Daten erfaßt und in ein einheitliches Format gebracht, wobei unwesentliche Systeminformation ausgefiltert wird. Die Managerkomponente eines solchen Knotens muß die Managementverfahren unterstützen, die die beiden Sohnknoten verwenden.

- Auf Ebene 3 werden die MTA und Systemdaten in Beziehung zueinander gesetzt und weiter komprimiert. Nach oben bietet ein Knoten dieser Ebene nur noch die wichtigsten Informationen bzgl. des MTA an, etwa Status und Auslastung.

- Auf Ebene 4 werden die Informationen mehrerer MTAs zusammengefaßt. Diese Ebene bildet zusammen mit allen höheren Ebenen, sofern solche existieren, ein hierarchisches Verwaltungskonzept.

Dieses Instrumentarium ermöglicht dem Verwalter, eine Momentaufnahme des EMail-Verbundes zu erhalten. Während das für die Verwaltung von Koppelelementen bereits ausreicht, sind im Anwendungsmanagement detailliertere Informationen gefragt. Beispielsweise ist nicht nur relevant, wieviele Nachrichten verlorengegangen sind, sondern auch, welche betroffen sind.

Um diesem Problem zu begegnen, protokolliert jeder MTA die wichtigsten Aktionen mit. Dazu gehört das Entgegennehmen und der Weiterversand von Nachrichten und deren Konvertierung ebenso wie aufgetretene Probleme und Fehler, etwa Verzögerung beim Transport oder Verlust einer Nachricht.

Damit existiert im Verbund eine verteilte Datenbank, die alle Aktionen des gesamten Systems enthält. Durch die Möglichkeit, von der Managementstation aus auf diese Daten zuzugreifen, entsteht virtuell eine globale Datenbasis, die eine Vielzahl von Anwendungen findet:

- Verfolgen des Weges einer Nachricht
- Erkennen von Nachrichtenverlust
- Benutzerbezogene Abrechnung
- Erfassung statistischer Daten, z.B. Auslastung
- Überwachung von QoS-Vorgaben

4.3 Ein normierter Ansatz

4.3.1 Network Services Monitoring MIB

Es existiert eine größere Anzahl von verteilten Anwendungen, für die es sinnvoll und notwendig ist, sie in ein integriertes Management einzugliedern. Daher wurde in [RFC1565] der Versuch unternommen, eine MIB zu entwerfen, die einerseits die wichtigsten Daten jeder auf einem System laufenden netznahen Anwendung bereitstellt, andererseits aber auch möglichst allgemein gehalten ist, um auf eine große Anzahl solcher Applikationen anwendbar zu sein.

Weiterhin soll diese MIB als Basisbaustein für applikationsspezifische Erweiterungen dienen, von denen eine, nämlich die Mail Monitoring MIB, in Abschnitt 4.3.2 beschrieben wird. Weitere Beispiele sind die DSA Monitoring MIB ([RFC1567]) und die Relational Database Management System MIB ([RFC1697]).

Jeder überwachten Applikation wird eine Indexnummer zugeordnet, mit der sie sowohl in der Network Services MIB als auch in allen applikationsspezifischen MIBs eindeutig identifiziert werden kann.

Die MIB setzt sich aus zwei Tabellen zusammen, wobei die erste verschiedene Status-Daten jeder einzelnen Applikation aufführt, während die zweite deren Verbindungen zu Partnerapplikationen angibt. Die logische Struktur der MIB ist in Abbildung 4.2 dargestellt.

applTable (indiziert mit applIndex)

> Diese Tabelle stellt Objekte für alle möglichen netznahen verteilten Anwendung bereit, wobei jeweils Daten wie der Name, der aktuelle Status oder verschiedene Verbindungsstatistiken zur Verfügung gestellt werden.

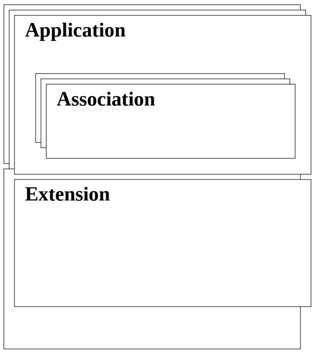


Abbildung 4.2: Logische Struktur der Network Services Monitoring MIB

Jede Zeile dieser Tabelle repräsentiert eine solche Applikation. Der in dieser Tabelle eingeführt Index ist in allen weiteren Tabellen Index oder zumindestens erste Komponente eines mehrdimensionalen Indexes.

applIndex
: Index, der die jeweilige Applikation eindeutig identifiziert

applName
: Name der Applikation

applDirectoryName
: Name des Verzeichniseintrags, in dem statische Informationen dieser Applikation abgelegt sind

applVersion
: Version der Applikation

applUptime
: Wert von sysUpTime, als die Applikation zuletzt initialisiert wurde

applOperStatus
: Zustand der Applikation, mögliche Werte sind „up" (funktionsfähig und verfügbar), „down" (nicht verfügbar), „halted" (funktionsfähig, aber nicht verfügbar), „congested" (funktionsfähig, aber überlastet) und „restarting" (demnächst verfügbar)

applLastChange
: Wert von sysUpTime, als applOperStatus sich zuletzt geändert hat

applInboundAssociations
: Aktuelle Anzahl der akzeptierten Verbindungen

applOutboundAssociations
Aktuelle Anzahl der initiierten Verbindungen

applAccumulatedInboundAssociations
Gesamtzahl der akzeptierten Verbindungen

applAccumulatedOutboundAssociations
Gesamtzahl der initiierten Verbindungen

applLastInboundActivity
Wert von sysUpTime, als die letzte akzeptierte Verbindung bestand

applLastOutboundActivity
Wert von sysUpTime, als die letzte initiierte Verbindung bestand

applRejectedInboundAssociations
Anzahl der abgewiesenen Verbindungen

applFailedOutboundAssociations
Anzahl der fehlgeschlagenen initiierten Verbindungen

assocTable (indiziert mit applIndex und assocIndex)

Zu jeder in der applTable beschriebenen verteilten Anwendung werden in dieser Tabelle die einzelnen aktuell bestehenden Verbindungen aufgeführt, wobei Informationen etwa zu Verbindungspartner, -protokoll und -dauer bereitgestellt werden.

assocIndex
Index, der die jeweilige Verbindung für die Applikation eindeutig identifiziert

assocRemoteApplication
Name des entfernten Systems

assocApplicationProtocol
Bezeichnung des verwendeten Protokolls

assocApplicationType
gibt an, ob die entfernte Applikation ein untergeordneter Client oder ein gleichgestellter Server ist und von welcher Seite die Verbidung initiiert wurde

assocDuration
Wert von sysUpTime, als die Verbindung aufgebaut wurde

4.3.2 Mail Monitoring MIB

Basierend auf der in Abschnitt 4.3.1 vorgestellten Network Services Monitoring MIB wird in [RFC1566] eine Erweiterung für die Überwachung eines Message Transfer Agent vorgestellt.

Diese MIB basiert auf folgendem, vier Schritte umfassenden Nachrichtenfluß durch den MTA:

1. Die Nachricht wird von einem benachbarten UA, MTA, MS oder AU empfangen.

2. Die nächste Station auf dem Weg zum Ziel wird bestimmt. Falls verschiedene Empfänger der Nachricht über verschiedene Stationen erreicht werden, wird die Nachricht entsprechend vervielfältigt.

3. Gegebenenfalls muß die Nachricht wird in ein Format, das von der nächsten Station akzeptiert wird, konvertiert werden.

4. Die Nachricht wird an die ermittelte Station weitergereicht.

Zwischenspeicherung ist zwischen jedem Paar von aufeinanderfolgenden Schritten denkbar.

Üblicherweise gruppiert der MTA verschiedene Funktionen, etwa den Empfang von Nachrichten über bestimmte Verbindungen, die Queueverwaltung oder den Versand von Nachrichten an bestimmte Nachbarn, in Funktionsgruppen, wobei die Granularität dieser Gruppen zwischen verschiedenen Herstellern variieren kann. Solche Gruppen können sich auch überschneiden, allerdings sollte jeder Aspekt des MTAs mindestens einer Gruppe zugeordnet werden.

In der Mail Monitoring MIB sind nur MTA-spezifische Daten untergebracht, Status- und Verbindungs-Daten können aus der Network Services Monitoring MIB (Abschnitt 4.3.1) entnommen werden.

Die MIB gliedert sich in drei Tabellen, wobei in der ersten Tabelle jeder auf dem System laufende MTA durch eine Zeile repräsentiert wird. In der zweiten Tabelle sind alle Gruppen aller MTAs dargestellt, während die dritte Tabelle den Zusammenhang zwischen diesen Gruppen und den vom MTA unterhaltenen Verbindungen herstellt. Die logische Struktur der MIB ist in Abbildung 4.3 dargestellt.

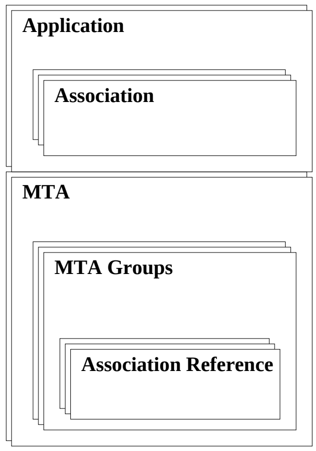


Abbildung 4.3: Logische Struktur von Network Services und Mail Monitoring MIB

mtaTable (indiziert mit applIndex)

In dieser Tabelle wird jeder auf dem System laufende MTA durch genau einen Eintrag repräsentiert, wobei der Index derselbe ist, den der jeweilige MTA in der Network Services Monitoring MIB belegt. Die Einträge in dieser Tabelle geben verschiedene MTA-globale Traffic-Statistiken wieder.

mtaReceivedMessages
Anzahl der empfangenen Nachrichten seit Initialisierung

mtaStoredMessages
Anzahl der zur Zeit zwischengespeicherten Nachrichten

mtaTransmittedMessages
Anzahl der versandten Nachrichten seit Initialisierung

mtaReceivedVolume
Volumen der empfangenen Nachrichten seit Initialisierung

mtaStoredVolume
Volumen der zur Zeit zwischengespeicherten Nachrichten

mtaTransmittedVolume
Volumen der versandten Nachrichten seit Initialisierung

mtaReceivedRecipients
Gesamtzahl der Empfänger aller empfangenen Nachrichten seit Initialisierung

mtaStoredRecipients
Gesamtzahl der Empfänger aller zur Zeit zwischengespeicherten Nachrichten

mtaTransmittedRecipients
Gesamtzahl der Empfänger aller versandten Nachrichten seit Initialisierung

mtaGroupTable (indiziert mit applIndex und mtaGroupIndex)

In der mtaGroupTable wird der Verkehr innerhalb des MTAs nach Gruppen aufgeschlüsselt. Desweiteren werden die Verbindungsstatistiken der Network Services Monitoring MIB nach einzelnen Gruppen zergliedert, wobei zusätzlich einige Attribute zur Fehlerbehandlung ergänzt werden.

mtaGroupIndex
Index, der die jeweilige Gruppe innerhalb eines MTA eindeutig identifiziert

mtaGroupReceivedMessages
Anzahl der Nachrichten, die seit Initialisierung von dieser Gruppe empfangen wurden

mtaGroupRejectedMessages
Anzahl der Nachrichten, die seit Initialisierung von dieser Gruppe zurückgewiesen wurden

mtaGroupStoredMessages
Anzahl der Nachrichten, die zur Zeit von dieser Gruppe zwischengespeichert werden

mtaGroupTransmittedMessages
Anzahl der Nachrichten, die seit Initialisierung von dieser Gruppe versandt wurden

mtaGroupReceivedVolume
Volumen der Nachrichten, die seit Initialisierung von dieser Gruppe empfangen wurden

mtaGroupStoredVolume
Volumen der Nachrichten, die zur Zeit von dieser Gruppe zwischengespeichert werden

mtaGroupTransmittedVolume
Volumen der Nachrichten, die seit Initialisierung von dieser Gruppe versandt wurden

mtaGroupReceivedRecipients
Gesamtzahl der Empfänger der Nachrichten, die seit Initialisierung von dieser Gruppe empfangen wurden

mtaGroupStoredRecipients
Gesamtzahl der Empfänger der Nachrichten, die zur Zeit von dieser Gruppe zwischengespeichert werden

mtaGroupTransmittedRecipients
Gesamtzahl der Empfänger der Nachrichten, die seit Initialisierung von dieser Gruppe versandt wurden

mtaGroupOldestMessageStored
Alter der ältesten Nachricht innerhalb der Gruppe

mtaGroupInboundAssociations
Aktuelle Anzahl der akzeptierten Verbindungen

mtaGroupOutboundAssociations
Aktuelle Anzahl der initiierten Verbindungen

mtaGroupAccumulatedInboundAssociations
Gesamtzahl der akzeptierten Verbindungen

mtaGroupAccumulatedOutboundAssociations
Gesamtzahl der initiierten Verbindungen

mtaGroupLastInboundActivity
Zeit seit der letzten akzeptierten Verbindung

mtaGroupLastOutboundActivity
Zeit seit der letzten initiierten Verbindung

mtaGroupRejectedInboundAssociations
Anzahl der abgewiesenen Verbindungen

mtaGroupFailedOutboundAssociations
Anzahl der fehlgeschlagenen initiierten Verbindungen

mtaGroupInboundRejectionReason
Grund, warum die letzte eingehende Verbindung abgewiesen wurde

mtaGroupOutboundConnectFailureReason
Grund, warum die letzte ausgehende Verbindung fehlgeschlagen ist

mtaGroupScheduledRetry
Zeitspanne, die noch verstreichen muß, ehe der nächste Verbindungsaufbau initiiert werden wird

mtaGroupMailProtocol
Bezeichnung des verwendeten Protokolls

mtaGroupName
Beschreibender Name einer Gruppe

mtaGroupAssociationTable (indiziert mit applIndex, mtaGroupIndex und mtaGroupAssociationIndex)

In dieser Tabelle werden die einzelnen Verbindungen aus der assocTable der Network Services Monitoring MIB den verschiedenen Gruppen des MTAs zugeordnet.

mtaGroupAssociationIndex
Index einer Verbindung in der assocTable, die dieser Gruppe zugeordnet ist.

4.3.3 Bewertung

Mit der Network Services Monitoring MIB ([RFC1565]) liegt ein Internetstandard für die Überwachung von netznahen verteilten Anwendungen vor. Diese MIB enthält nur Informationen, die allen solchen Applikationen gemeinsam sind, sie ist also der Basisbaustein einer Managementlösung. Applikationsspezifische Informationen werden in zusätzlichen Bausteinen abgelegt. Für MTA-, DSA- sowie Datenbank-Management wurden bereits solche Zusatzbausteine spezifiziert. Dieses offene und erweiterungsfähige Konzept läßt hoffen, daß in naher Zukunft auch für andere verteile Applikationen Agenten auf Basis dieser MIB entwickelt werden.

Die MIB enthält u.a. den Status (up, down,...) der jeweiligen Applikation, was eine Zustandsüberwachung möglich macht. Weiterhin wird die Anzahl der seit Systemstart initiierten, akzeptierten und fehlgeschlagen sowie eine Liste der aktuell bestehenden Verbindungen angeboten, woraus man mit Zusatzwissen eventuell gewisse Rückschlüsse auf Auslastung und Sicherheitsprobleme ziehen könnte.

Desweiteren existiert bereits eine Variable, die einen Verweis auf den Directory-Eintrag der jeweiligen Applikation enthält, was in Zukunft (sobald entsprechende Übergange zwischen Managementplattformen und Directory Systems verfügbar sind) von Vorteil sein könnte.

Wie bereits beschrieben, existiert speziell für die Verwaltung von MTAs ein weiterer MIB-Baustein, die in [RFC1566] spezifizierte Mail Monitoring MIB.

Diese MIB enthält zum einen Daten, die den gesamten MTA betreffen, bietet andererseits aber auch die Möglichkeit einer Untergliederung des MTA in verschiedene Funktionsgruppen. Leider wird nicht näher spezifiziert, nach welchen Kriterien diese Gliederung erfolgt, so daß diese Einteilung bei verschiedenen Produkten unterschiedlich ausfallen wird und somit die Informationen dieses MIB-Abschnittes von unterschiedlichem Nutzen sein dürften. Andererseits wird durch diese Vorgehensweise den Herstellern eine große Freiheit in der Definition dieser Gruppen gewährt, wodurch die Akzeptanz dieser Lösung erhöht wird.

Der MTA-globale Abschnitt dieser MIB bietet Zähler für die Anzahl und das Volumen der empfangenen, versandten und gespeicherten Nachrichten an. An Hand der ersten beiden Kriterien kann auf die Auslastung des MTA geschlossen werden, die Menge der gespeicherten Nachrichten läßt eventuell Rückschlüsse auf Fehlersituationen innerhalb bzw. im Umfeld des MTA zu.

Im gruppenspezifischen Abschnitt der MIB werden u.a. dieselben Daten mit entsprechend feinerer Granularität zur Verfügung gestellt, was eine Eingrenzung von Performanceproblemen oder Fehlern ermöglicht. Weitere Variablen wie das Alter der ältesten Nachricht innerhalb dieser Gruppe sowie Statistiken über die Anzahl der aufgebauten und fehlgeschlagenen Verbindungen, der Aktivität auf diesen Verbindungen sowie eine Beschreibung des letzten Aufbaufehlers ergänzen diese Überwachungsmöglichkeiten.

Generell gilt, daß beide MIBs im wesentlichen Informationen zur Erkennung von Fehlersituationen und Performanceproblemen sowie sehr eingeschränkt Konfigurationsdaten beinhalten, jedoch nicht auf die Sicherheits- und Accountingproblematik eingehen.

Sämtliche Variablen der MIBs sind read-only, es ist also auf diesem Weg keine Einflußnahme auf das Verhalten des MTA möglich.

Als weiterer Kritikpunkt kann angeführt werden, daß beide MIBs keinerlei Traps enthalten. Die Intelligenz zur Erkennung von Fehlersituationen und Ausfällen muß also vollständig auf der Seite der Managementplattform zur Verfügung gestellt werden. Auch ist die Netzbelastung durch ständiges Pollen der Statusinformation höher.

Innerhalb ihres Haupteinsatzgebiets ist die Kombination aus Network Services Monitoring MIB und Mail Monitoring MIB dennoch gut geeignet, die Funktion eines MTAs zu überwachen und Hinweise auf Probleme zu liefern. Für eine genaue Eingrenzung reicht allerdings die angebotene Information, die sich zum großen Teil nur aus statistischen Daten zusammensetzt, nicht aus. Hier wird man in den meisten Fällen auf die proprietären Managementwerkzeuge der Hersteller zurückgreifen müssen.

Kapitel 5

Einsatzszenarien

Im folgenden werden zwei existierende Einsatzszenarien vorgestellt. Zuerst wird das firmeninterne Kommunikationsnetz der BASF AG beschrieben, anschließend wird der X.400-Mailverbund des DFN-Vereins vorgestellt, der Hochschulen und Forschungseinrichtungen in Deutschland verbindet.

5.1 Das MHS-Szenario bei der BASF AG

Die BASF AG betreibt ein firmenprivates, weltweit ausgedehntes Kommunikationsnetz. Ziel der realisierten Architektur war,

- die Kommunikation innerhalb des Unternehmens und mit Geschäftspartnern zu vereinfachen.
- möglichst wenig Belastung des privaten Corporate Network zu erzeugen.
- eine hierarchische Managementstruktur aufzubauen, um möglichst kurze Verwaltungswege zu erhalten.

Bereits vor der Einführung dieses unternehmensübergreifenden Netzes existierten in den einzelnen Abteilungen verschiedene proprietäre Nachrichtenübermittlungssysteme. Diese Liste umfaßt unter anderem

- hostbasierte Systeme: Office Vision, Memo, Profs, Wangoffice, CEO
- verteilte Systeme: MS-Mail, Lotus Notes, DEC Mail Bus, VMS Mail

Eine homogene Kommunikationsinfrastruktur schied wegen der dafür notwendigen Migration auf ein einheitliches System, die u.a. ein Umlernen bei den Mitarbeitern vorausgesetzt und erhebliche Ersatzinvestitionen erfordert hätte, aus. Aus diesem Grund wurde ein weltweites X.400/SMTP-Backbone-Netz aufgebaut, an das all diese proprietären Systeme mittels Gateways angebunden sind.

Die MTAs dieses Backbones — größtenteils wird XT-PP von Nexor (siehe Abschnitt 6.1) eingesetzt — unterstützen mindestens die beiden Protokolle X.400 und SMTP und können bereits selbst zwischen beiden Nachrichtenformaten konvertieren.

5.1.1 Weltweite Struktur

In jedem Land, in dem eine größere Niederlassung der BASF AG existiert, betreibt der Konzern eine PRMD mit Namen *basf*. Zwischen diesen PRMDs existiert ein voll vermaschter Backbone, auf dem mittels der Protokolle der X.400-Welt sowie SMTP Nachrichten zwischen den einzelnen Niederlassungen ausgetauscht werden. Für die Anbindung der einzelnen PRMDs an die öffentliche X.400-Infrastruktur existiert jeweils ein Übergang zu einer nationalen X.400-ADMD.

Eine Ausnahme in diesem Szenario bildet die deutsche Niederlassung. Da dort bereits vor Einführung des Kommunikations-Backbones die PRMD *basf-ag* vorhanden und eingeführt war, wurde diese zusätzlich beibehalten. In Deutschland verfügt die BASF AG damit also über zwei PRMD-Namen, *basf* und *basf-ag*.

Zusätzlich existieren in Deutschland und den USA Übergänge ins Internet, über die der gesamte SMTP-Verkehr von und nach außerhalb des Konzerns abgewickelt wird.

Diese Struktur ist in Abbildung 5.1 beispielhaft für die Niederlassungen in Deutschland, der Schweiz, Belgien sowie den USA aufgezeigt.

5.1.2 Struktur innerhalb eines Landes

Innerhalb einer PRMD existiert ein zentraler MTA (das sog. Postamt), der alle Anbindungen der PRMD zur Außenwelt realisiert. Dieser MTA unterhält Links zu der jeweiligen nationalen ADMD sowie zu sämtlichen Postämtern aller anderen zur BASF gehörenden PRMDs. Alle anderen in der jeweilgen Domain gelegenen MTAs unterhalten ausschließlich Links innerhalb ihrer PRMD. Das Postamt ist somit der zentrale Punkt innerhalb der PRMD, den

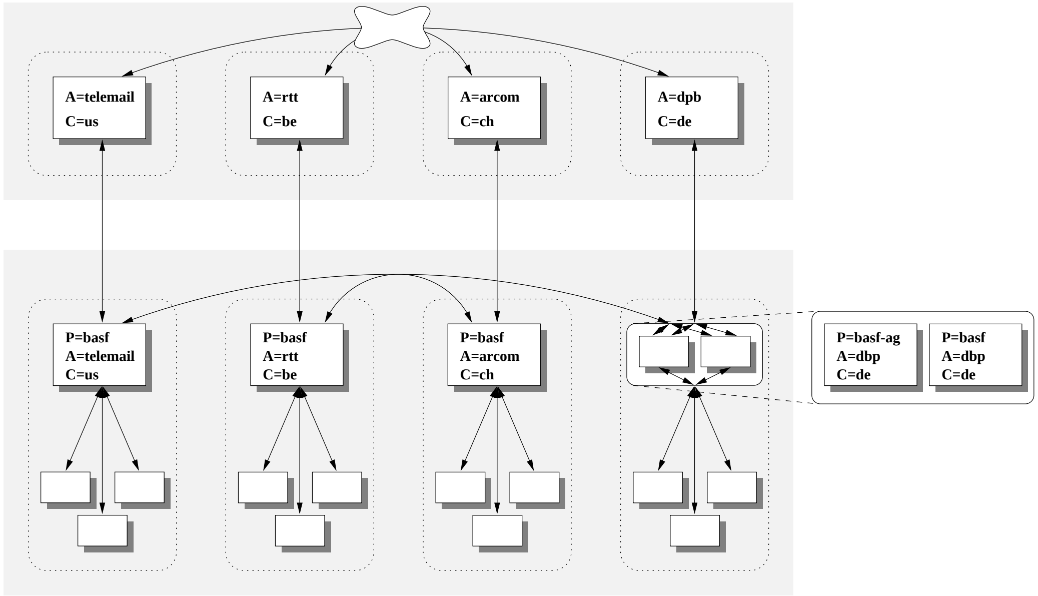


Abbildung 5.1: PRMDs und ADMD-Anschlüsse der BASF AG

jede Nachricht aus der bzw. in die PRMD durchlaufen muß (vgl. Abbildung 5.2).

Eine Ausnahme von dieser Regel existiert in Deutschland. Hier übernehmen zwei MTAs die Aufgaben des Postamts, wobei einer den Internetanschluß und die PRMD basf und der andere die PRMD basf-ag und den Übergang zu den HOST-Gateways realisiert.

Nachrichten, die an Benutzer der verschiedenen hostbasierten Mailsysteme wie etwa Office Vision gerichtet sind, werden bereits vom Postamt im X.400-Format an die entsprechenden Gateways weitergeleitet.

Alle anderen Nachrichten werden an einen exponierten MTA innerhalb der Abteilung des jeweiligen Empfängers weitergereicht, wobei hier als Protokolle sowohl SMTP als auch X.400 zum Einsatz kommen.

5.1.3 Abteilungsstruktur

Analog zum Postamt existiert auch innerhalb jeder Abteilung eine zentrale Instanz, über die nahezu sämtlicher Nachrichtenverkehr in die und aus der Abteilung läuft. Eine Ausnahme von dieser Regel bilden die Mitarbeiter, die eines der hostbasierten Nachrichtensysteme benutzen und die ihre Nachrichten mittels Loginsession auf dem Host bearbeiten.

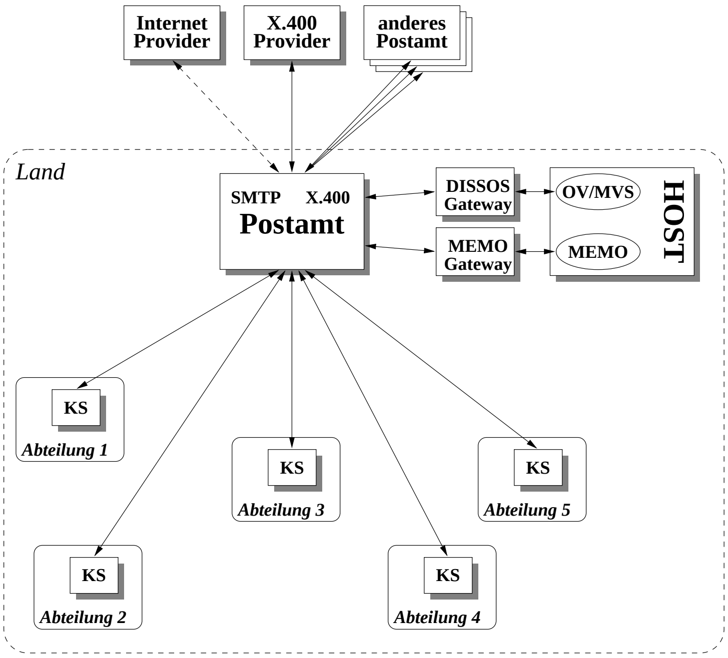


Abbildung 5.2: Das Postamt als Landeszentrale

Der Kommunikationsserver tauscht mit dem Postamt Nachrichten im RFC- und X.400-Format aus. Anwender, die ihre Mail via SMTP empfangen und verschicken, werden direkt vom Kommunikationsserver bedient. Für alle proprietären EMail-Systeme, die in der Abteilung eingesetzt werden, existieren Gateways, die mit dem Kommunikationsserver mittels der X.400-Protokolle und mit den User Agents der Anwender mittels des jeweiligen proprietären Protokolls kommunizieren (siehe auch Abbildung 5.3).

5.1.4 Adressierung

Die Gatewaystruktur schlägt sich allerdings nicht in den Adressen der Angestellten nieder, für die Adressierung werden nur organisatorische Daten sowie der Name verwendet:

country-name	*Land*
administration-domain-name	*Provider-ADMD*
private-domain-name	BASF
organization-name	*Firma*
organizational-unit-name	*Abteilung*
sur-name	*Nachname*
given-name	*Vorname*

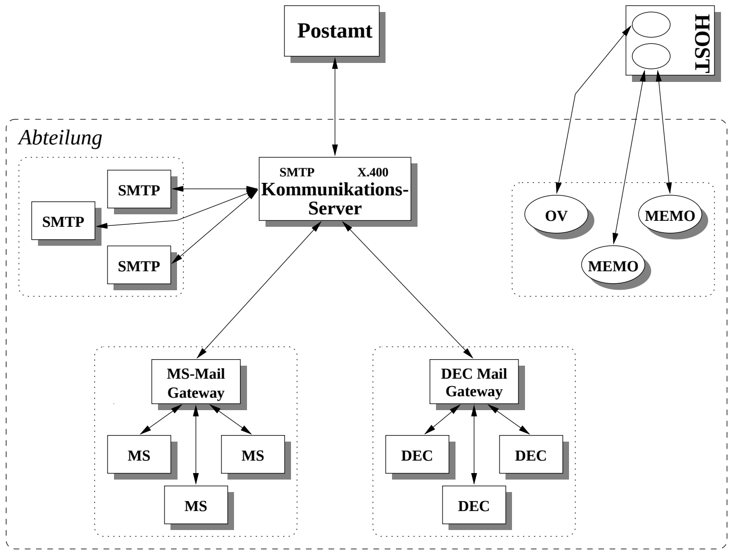


Abbildung 5.3: MHS-Struktur innerhalb einer Abteilung

Um dieses Konzept technisch zu realisieren, existiert eine zentrale Benutzerdatenbank, in der neben verschiedenen anderen Informationen auch verschiedene Daten für den Nachrichtenübermittlungsdienst enthalten sind, etwa das verwendete Mailsystem.

Diese Daten werden periodisch aus den Verwaltungsdaten der verschiedenen proprietären Nachrichtenübermittlungssysteme gewonnen, was zum Teil automatisch, zum Teil von Hand geschieht.

Aus dieser Zentraldatenbank wird eine MHS-spezifische Routingdatenbank erzeugt, die vom Postamt-MTA verwendet wird, um eingehende Nachrichten je nach Adressat an das entsprechende HOST-Gateway bzw. an den Kommunikationsserver der jeweiligen Abteilung weiterzuleiten.

5.1.5 Managementaspekte

Bei der BASF AG wurde eine hierarchische Verwaltungsstruktur verwirklicht, die auch Aufgabenteilung erlaubt.

- Die Verwaltung der abteilungslokalen proprietären EMail-Systeme liegt im Verantwortungsbereich von Mitarbeitern dieser Abteilungen. Insbesondere übernehmen diese auch das Ein- und Austragen der lokalen Benutzer.

- Auch die Kommunikationsserver in den Abteilungen sollen, soweit das möglich ist, bereits vor Ort verwaltet werden.
- Innerhalb der zentralen Informatikabteilung der BASF AG gibt es eine Gruppe von Mitarbeitern, die für den Betrieb des X.400/SMTP-Backbones zuständig ist. Diese EMail-Gruppe ist für die Verwaltung der Postämter zuständig, kann aber auch in die Funktion der Kommunikationsserver, die ja ebenfalls zum Backbone zählen, eingreifen, um den Betrieb aufrecht zu erhalten.

Zur Zeit sind die Managementabläufe dergestalt, daß jeder MTA von der EMail-Gruppe in periodischen Abständen (Größenordnung mehrere Stunden) bzgl. Verfügbarkeit und Auslastung überprüft wird. Mit diesem Verfahren kann zwar die Funktion des MHS überwacht werden, schnelles und zielgenaues Reagieren auf Ausnahmesituationen ist allerdings nicht möglich.

An eine Managementlösung stellt das Verwaltungspersonal daher verschiedene Anforderungen:

- Die Lösung soll in die von den Netz- und Systemmanagementgruppen benutzten Managementprodukte Spectrum (siehe auch Abschnitt 6.2) und MaestroVision integriert werden. Weiterhin sollen nur genormte Schnittstellen (siehe Abschnitt 4.3) für die Kommunikation mit den MTAs verwendet werden, um Wartung und Portierung auf neue Systeme zu erleichtern.
- In jedem Fall müssen der Funktionsstatus des MTAs und der Zustand der Nachrichtenqueue abrufbar sein, detailliertere Auskünfte über die MTA-Funktionsbereiche wären wünschenwert. Weiterhin soll das Managementpersonal benachrichtigt werden, wenn ein MTA ausfällt bzw. wenn der Füllstand seiner Queue einen vorgegebenen Grenzwert überschreitet.
- Wünschenswert wäre weitergehende Funktionalität, wie Accounting an Übergängen zu anderen Betreiber und Meldung von unbefugter Nutzung an die Managementstation.

5.2 Der X.400-Verbund des deutschen Forschungsnetzes

Das Deutsche Forschungsnetz ist das Kommunikationssystem für die Wissenschaft in Deutschland. Es wird vom DFN-Verein verwaltet.

Neben dem Zugang zum Wissenschaftnetz WIN bietet der DFN-Verein auch Mehrwertdienste an, die neben dem Zugang zum Internet auch die Anbindung an den DFN-X.400-Mailverbund beinhalten.

Der DFN-Verein selbst betreibt zwei ADMDs *d400* und *d400-gw*. Desweiteren werden Übergänge zu anderen europäischen ADMDs im Bereich der Wissenschaft und Forschung sowie zur ADMD *dbp*, die von der Telekom AG im Rahmen des Telebox-Projekts betrieben wird und über die eine größere Anzahl von Firmen (auch die BASF AG, siehe Abschnitt 5.1) angeschlossen sind, betrieben. Der DFN-Verein stellt auf Wunsch auch Übergänge zu anderen ADMDs zur Verfügung.

5.2.1 Die ADMD d400

Dieser ADMD sind die PRMDs aller am X.400-Mail-Verbund beteiligten Organisationen, vor allem Universitäten und Forschungsinstitute zugeordnet.

Die Vernetzungsstruktur ist sternförmig. In ihrem Zentrum sind die MTAs des DFN-Relay angeordnet. Jede PRMD muß einen MTA, den sogenannten Entry-Point, ausweisen, der die Verbindung zum DFN-Relay realisiert. Links zwischen PRMDs sind möglich, müssen allerdings von den beteiligten Organisationen in Eigenverantwortung verwaltet werden. Die Default-Route muß allerdings immer auf das DFN-Relay zeigen.

Durch diese Gliederung sind die X.400-Adreßattribute country-name (DE), administration-domain-name (D400) und private-domain-name (Name der jeweiligen Organisation) bereits festgelegt. Das Attribut organization-name wurde im DFN bei Universitäten und anderen Hochschulen nie, bei Forschungseinrichtungen nur selten verwendet. Die Gliederung der PRMDs wird vielmehr durch das Attribut organizational-unit-name vorgenommen, gängige Bezeichnungen sind bei Hochschulen etwa informatik, e-technik und rz (Rechenzentrum).

Bezüglich des inneren Aufbaus einer PRMD existieren von Seiten des DFN-Vereins keine Vorgaben. Das Leibniz-Rechenzentrum in München betreibt beispielsweise nur einen MTA, der für die PRMDs *lrz-muenchen*, *tu-muenchen*, *uni-muenchen* und *fh-muenchen* als Entry-Point dient. Dieser MTA fungiert ausschließlich als Gateway zwischen dem DFN-X.400-Verbund und den im Münchner Hochschulnetz verwendeten MTAs, die größtenteils SMTP verwenden. Eingehende Nachrichten werden also einer Protokoll-Konvertierung unterzogen und anschließend mittels SMTP weitergeleitet.

5.2.2 Die ADMD d400-gw

Innerhalb der ADMD d400-gw sind die Übergänge eingeordnet, die der X.400-Mail-Verbund des DFN zu Nachrichtenübermittlungssystemen anderer Technologie unterhält. Im wesentlichen handelt es sich hier um je ein Gateway zum Internet und zum Bitnet.

Ziel der Einführung dieser ADMD war es, die teilweise völlig anders aufgebauten Adreßstrukturen in Fremdnetzen auf das in X.400 verwendete Format abzubilden und so den den Anwendern die Möglichkeit zu geben, die gewohnte Attributschreibweise zu verwenden.

Für diese Abbildung existieren eine Reihe von Tabellen, die das Domainformat, das im Internet verwendet wird, auf die Attribute private-domain-name, organization-name und organizational-unit-name (eventuell mehrfach) abbilden. So wird beispielsweise `hans@mystery.muc.de` auf `C=de;A=d400-gw;P=muc;O=mystery;S=hans` abgebildet.

Ebenso wird für Bitnet-Adressen die Pseudo-PRMD *bitnet* verwendet, beispielsweise wird `INF12FCC@DM0TUI1S` auf `C=de;A=d400-gw;P=bitnet;O=DM0TUI1S;S=INF12FCC` abgebildet.

Kapitel 6

Entwicklungsumgebung für den Prototypen

In diesem Kapitel werden die beiden Softwarepakete, die für die Implementierung des Prototypen ausgewählt wurden, vorgestellt.

Als MTA wird XT-PP von Nexor verwendet. Zum einen unterstützt dieses Produkt die in Abschnitt 4.3 beschriebenen MIBs, zum anderen kann die Implementierung unmittelbar im Kommunikationsnetz der BASF AG (siehe Abschnitt 5.1) getestet werden.

Als Managementplattform wurde Spectrum von Cabletron ausgewählt, da der objektorientierte Managementansatz dieses Systems den in Kapitel 3 beschriebenen Konzepten entgegenkommt und die mächtigen Entwicklungswerkzeuge die Implementierung stark vereinfachen.

6.1 Der MTA XT-PP von Nexor

6.1.1 Funktionsumfang

XT-PP ist ein Mail Transfer Agent (MTA), der eine Reihe von Protokollen unterstützt und auch über den reinen Nachrichtentransport hinausreichende Fähigkeiten besitzt.

XT-PP ist eine kommerzielle Weiterführung des PP[1]-Projekts, das 1985 am University College London gestartet wurde und dessen Zielsetzung war, ein frei verfügbares System zu entwickeln, das X.400 ([X400]) unterstützen und

[1] „PP is not an acronym. There is no truth in the rumour that PP stands for “Postman Pat,” a famous British postman.“ ([Kil91], Abschnitt 31.4)

eine hohe Flexibilität, Stabilität und Performanz aufweisen sollte. Eine ausführliche Beschreibung zu PP und seiner Geschichte findet sich in [Kil91]. Der Aufbau und Funktionsumfang von PP wurde in XT-PP beibehalten (siehe Abschnitt 6.1.2), zusätzlich wurden an vielen Stellen Erweiterungen vorgenommen.

XT-PP basiert auf dem ISODE (ISO Development Environment), das die oberen Schichten des OSI-Modells ([ISO 7498]) implementiert. Dieses Softwarepaket enthält die Dienste des Transport, Session und Presentation Layer, die Application Service Elements ACSE, ROSE und RTSE, die Applikationen FTAM und VTP sowie verschiedene Werkzeuge für den Umgang mit ASN.1.

XT-PP unterstützt die X.400-Protokollfamilie, wobei sowohl das P1-Protokoll der Recommendation von 1988 ([X482]) als auch aus Kompatibilitätsgründen das P1-Protokoll des entsprechenden Dokuments von 1984 implementiert wurden. Die verwendeten ISODE-Module setzen die Installation der unteren OSI-Schichten, namentlich X.25, CONS oder CLNS auf Schicht 1 bis 3 voraus. Alternativ kann auch der TCP/IP-Protokollstack mit dem in [RFC1006] beschriebenen Abbildungsverfahren benutzt werden.

Aufgrund der hohen Verbreitung der RFC-basierten Mailprotokolle (etwa im Internet) wurde auch diese Welt berücksichtigt. XT-PP verarbeitet daher auch nach [RFC 822] formatierte Nachrichten und unterstützt auch die Multipurpose Internet Mail Extension (MIME, [RFC1521, RFC1522]). Als Transportprotokolle werden hier UUCP und SMTP ([RFC 821]) verwendet, wobei eine entsprechende UUCP- bzw. TCP/IP-Installation vorausgesetzt wird.

Weiterhin kann der MTA als Gateway zwischen X.400- und RFC822-Netzen verwendet werden. Header und Nachrichteninhalte werden nach [RFC1327] konvertiert.

Als dritte Protokollwelt unterstützt XT-PP das FAX-Format. Der MTA steuert ein Modem mit der Fähigkeit, Faxe zu empfangen und zu versenden. Auch hier kann eine Konvertierung von und in die anderen Formate erfolgen.

Die Funktion des MTA wird durch eine Vielzahl von Konfigurationsdateien und -tabellen gesteuert, die größtenteils als lokale Dateien vorliegen. Die Tabellen für Routing sowie die Expansion von Verteilerlisten können alternativ als Einträge in einem X.500-Directory vorliegen. Falls SMTP verwendet wird, wird für Routingzwecke zusätzlich der MX-Record des Domain Name Systems ([RFC 974]) herangezogen.

6.1.2 Architektur

XT-PP hält alle Nachrichten in einer einzigen Message-Queue. Jede Nachricht wird in einer eigenen Verzeichnisstruktur gespeichert, wobei die verschiedenen Header- und Body-Anteile ggf. in einzelnen Dateien abgelegt werden.

Herz von XT-PP sind die beiden Verwaltungsprozesse submit und qmgr (queue manager). Um diese herum sind verschiedene Arbeitsprozesse, die sogenannten Channels, angeordnet, die alle weiteren Funktionen des MTAs implementieren (Abbildung 6.1).

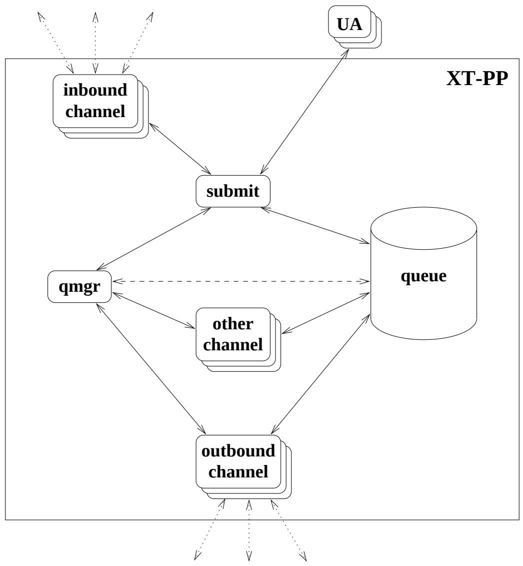


Abbildung 6.1: XT-PPs Prozeßstruktur (nach [NexPPb])

submit nimmt lokal oder entfernt erstellte Nachrichten entgegen, prüft ihre syntaktische Korrektheit, legt sie in der Message-Queue ab und benachrichtigt qmgr.

qmgr hält einen Index über den Inhalt der Queue im Hauptspeicher und steuert die Channels, die die Nachrichten ausliefern oder bearbeiten. qmgr selbst nimmt keinerlei Veränderungen an der Queue vor.

Die Channels von XT-PP lassen sich in verschiedene Typen gliedern, die sich ihrerseits grob in die vier Gruppen Mandatory, Protocol, Local Delivery und Formatting Channels einordnen lassen.

Mandatory Channels

Diese Channels stellen hauptsächlich Funktionen zur Queueverwaltung bereit. Im einzelnen sind dies:

qmgr-load aktualisiert die im Hauptspeicher gehaltene Nachrichtenliste anhand der in der Queue enthaltenen Nachrichten.

msg-clean entfernt alle Dateien einer bereits bearbeiteten Nachricht aus dem Spoolbereich.

timeout entfernt Nachrichten, die innerhalb einer vorgegebenen Zeit nicht ausgeliefert werden konnten, und verschickt einen Non-Delivery-Report an den Absender. Dieser Channel wird auch verwendet, um von der MTAconsole bzw. der lconsole (siehe Abschnitt 6.1.3) kommende timeout- oder delete-Aufforderungen zu erfüllen.

dr2rfc konvertiert das interne Format eines Delivery-Reports in eine RFC822-Nachricht.

warning verschickt an den Absender einer Nachricht einen Hinweis, daß sich die Auslieferung verzögert.

debris entfernt Nachrichten aus dem Spoolbereich, die dort aus irgendeinem Grund liegengeblieben, aber für den MTA nicht mehr von Interesse sind.

delay stellt eine Nachricht, die aufgrund externer Ursachen (etwa Verzögerungen beim Directoryzugriff) nicht sofort ausgeliefert werden kann, zurück.

Protocol Channels

Generell gibt es zu jedem Protokoll zwei Channeltypen, einen Inbound Channel, der Nachrichten von außen entgegen nimmt und an submit weiterreicht, und einen Outbound Channel, der Nachrichten an benachbarte MTAs ausliefert.

XT-PP stellt Channels für die Protokolle X.400-P1(88), X.400-P1(84), SMTP, UUCP und FAX bereit.

Local Delivery Channels

Diese Channels sind für die Auslieferung von Nachrichten an lokale Benutzer zuständig:

822-local liefert standardmäßig Nachrichten in ein Mailboxfile aus, kann zusätzlich vom Benutzer mittels einer .mailfilter-Datei konfiguriert werden.

shell startet in Abhängigkeit vom Absender Verarbeitungsprozesse.

list implementiert dateibasierte Verteilerlisten.

dirlist implementiert directorybasierte Verteilerlisten.

Formatting Channels

Diese Channels sind für die Formatierung und Konvertierung von Nachrichtenkopf bzw. -inhalt verantwortlich:

p2explode/p2flatten zerlegt eine komplexe Interpersonal Message in der Queue in ihre Bodyparts bzw. setzt diese Komponenten zu einer Nachricht zusammen.

mimeexplode/mimeflatten bietet jeweils dieselbe Funktionalität wie der korrespondierende p2-Channel, allerdings für MIME-Nachrichten.

P2toRFC/RFCtoP2 konvertiert X.400-Header in RFC822-Header bzw. umgekehrt.

rfc934 setzt eine mehrteilige RFC822-Nachricht nach der in [RFC 934] beschriebenen Vorgehensweise zu einer einzigen Nachricht zusammen. Die Funktionalität entspricht also dem mimeflatten-Channel, wobei allerdings ein anderes Zielformat verwendet wird.

fcontrol bildet die Schnittstelle zwischen XT-PP und einfachen Konverterroutinen, die verschiedene Typen von Nachrichteninhalten ineinander überführen, etwa zwischen ASN und IA5, zwischen X.400- und MIME-Bodyparts oder von Headerformaten zu Faxdeckblättern. Typischerweise wird pro Konverter mindestens ein Channel dieses Typs angelegt.

bpcontrol besitzt die gleiche Funktionalität wie fcontrol, erlaubt allerdings auch die Verarbeitung komplexerer (ggf. proprietärer) Quell- und Zielinhaltstypen.

6.1.3 Proprietäres Managementsystem

Im Lieferumfang von XT-PP sind einige Management-Tools enthalten. Neben den einfacheren Werkzeugen, die den Queue-Status ausgeben, die Konfigurationsfiles auf syntaktische Korrektheit prüfen oder eine gegebene X.400-Adresse anhand der Konfiguration verifizieren bzw. eine Probe an diese Adresse absenden, existieren zwei komplexere Werkzeuge, die MTAconsole und die lconsole.

Beide Tools haben in etwa den gleichen Funktionsumfang, die MTAconsole besitzt allerdings eine X11-Oberfläche, während die lconsole ein shellbasiertes Werkzeug ist und gut in Scripten verwendet werden kann.

Beim Start baut das jeweilige Verwaltungsprogramm eine Verbindung zu einem vom Benutzer vorgegebenen qmgr auf. Dabei wird zwischen autorisierten Verbindungen (mit Passwort) und nicht autorisierten Verbindungen unterschieden, wobei letztere nur eingeschränkten, lesenden Zugriff erhalten.

Sobald eine entsprechende Verbindung besteht, ist es möglich, den Status der einzelnen Channels, der bestehenden Verbindungen zu entfernten MTAs sowie jeder einzelnen Nachricht abzurufen. Ist die Verbindung autorisiert, können diese Verwaltungsobjekte auch ein- und ausgeschaltet sowie eine bestehende Auslieferungsverzögerung verändert werden. Einzelne Nachrichten können mit oder ohne Non-Delivery-Report an den Absender gelöscht werden.

6.1.4 SNMP-Agent

In der aktuellen Version von XT-PP ist im qmgr ein SNMPv2-Agent, der auf dem ISODE-SNMPv2-Paket basiert, enthalten. Der Agent stellt folgende MIBs bereit:

- system-Group der MIB-2 (Name, Kontakt, Standort, ...)
- snmp-Group der MIB-2 (Statistik-Information bzgl. SNMP-Requests)
- Network Services Monitoring MIB (siehe Abschnitt 4.3.1)
- Mail Monitoring MIB (siehe Abschnitt 4.3.2)

Um Koordinationsprobleme zwischen den Agenten verschiedener Anwendungen, die auf einen System laufen, zu vermeiden, schlägt Nexor ein Konzept

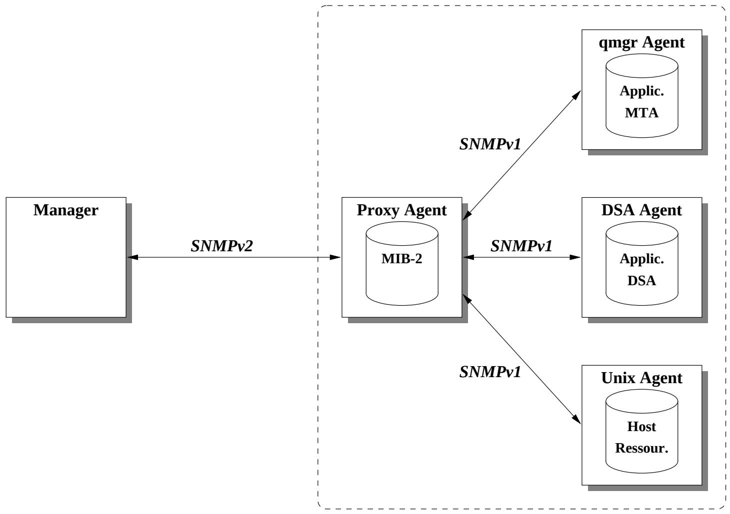


Abbildung 6.2: Vorgesehenes Manager-Agent-Szenario

mit einem Hauptagenten und mehreren Subagenten vor, wie es in Abbildung 6.2 dargestellt ist. Die genaue Konfiguration ist in Anhang A aufgeführt.

Nach diesem Konzept kommuniziert der Manager ausschließlich mit dem Proxy Agent, wobei SNMPv2 als Kommunikationsprotokoll zum Einsatz kommt. Der Proxy Agent stellt somit einen Single Point of Control im Agentensystem dar. Der Manager spezifiziert in seiner Anfrage, ob der Agent die Anfrage aus seiner lokalen Datenbasis beantworten oder die gewünschte Information von einem bestimmten Subagenten einholen soll.

Im zweiten Fall tritt der Proxy Agent den Subagenten gegenüber als Manager auf. Er schickt eine SNMPv1-Anfrage an den jeweiligen Subagenten und leitet dessen Antwort an den Manager weiter.

Als Proxy Agent wird der SNMPv2-Agent der ISODE eingesetzt, der diese Proxy-Funktionalität vollständig implementiert. Dieser Agent kann auch so konfiguriert werden, daß er SNMPv1-Anfragen vom Manager akzeptiert, wobei eine Abbildung des Community-String auf die in SNMPv2 verwendeten Zugriffsmechnismen erfolgt.

Leider hat sich herausgestellt, daß dieser Mechanismus nur mit agentenlokalen Variablen funktioniert, dagegen nicht mit Informationen, die von einem Subagenten geholt werden müssen. Aus diesem Grund wurde für die Entwicklung des Prototypen das in Abbildung 6.3 abgebildete Szenario verwendet, bei dem der Manager direkt mit dem qmgr-Subagenten kommuniziert.

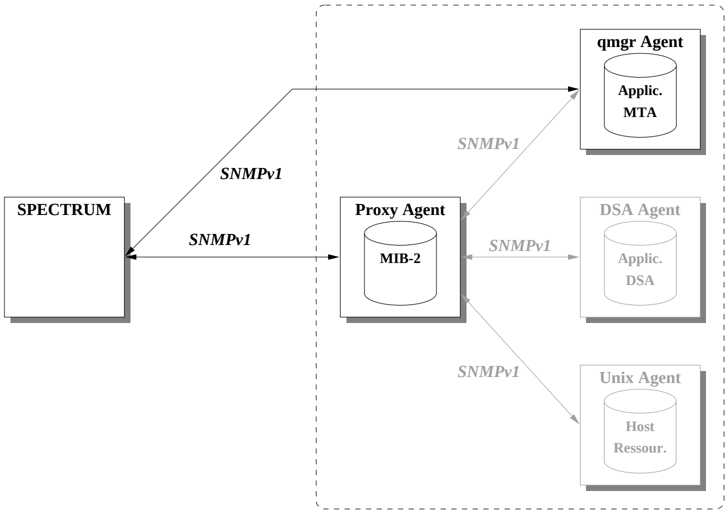


Abbildung 6.3: Verwendetes Manager-Agent-Szenario

Auch der qmgr-Agent ist nicht fehlerfrei:

- Objekte eines zusammengesetzten Datentyps, deren Länge 127 Byte überschreitet, werden mit Länge *unbestimmt* codiert, was in SNMPv1 nicht erlaubt ist:

  Objekte werden in ASN.1 durch ein Tripel (Typkennung, Länge, Inhalt) codiert. Bei zusammengesetzten Datentypen, etwa Strukturen oder Listen, ist es möglich, als Länge den Wert *unbestimmt* zu spezifizieren und das Objekt durch eine Endekennung abzuschließen. In SNMPv1-Paketen ist diese Vorgehensweise allerdings explizit verboten ([Ros91]).

  Der qmgr-Agent verwendet ungeachtet dessen diese Codierung, wenn die Länge des Objektinhalts 127 Byte überschreitet. Da ein SNMP-Paket selbst ein solches zusammengesetztes ASN.1-Objekt ist, werden alle SNMP-Pakete, die länger als 129 Byte sind — was bei Paketen, die mehrere Variablen enthalten, oft vorkommt —, inkorrekt codiert.

- get-next-Requests auf zweifach indizierte Tabellen werden falsch beantwortet, wenn der erste Index vorgegeben wird:

  Als Beispiel wird die assocTable (Abschnitt 4.3.1) verwendet, die folgendermaßen belegt ist (nur die ersten beiden Spalten):

Index	assocIndex	assocRemoteApplication
(238,3)	3	responder
(238,7)	7	smtp

Ein get-next-Request für assocIndex liefert erwartungsgemäß OID und Inhalt von assocIndex.238.3, ebenso wird ein entsprechender Request auf assocIndex.238.3 korrekt mit assocIndex.238.7 beantwortet.

get-next auf assocIndex.238 sollte OID und Inhalt von assocIndex.238.3 (also des ersten Eintrags, dessen erster Index 238 ist) liefern, tatsächlich antwortet der Agent mit assocRemoteApplication.238.3, er überspringt in diesem Fall also die erste Spalte vollständig.

6.2 Die Managementplattform Spectrum von Cabletron

6.2.1 Wissensbasis

Die konzeptionelle Grundlage von Spectrum ist die sogenannte Inductive Modeling Technology (IMT), auf deren Basis in der Managementplattform ein Abbild der realen Netzwelt modelliert wird. Im folgenden werden die Konzepte, die hinter IMT stehen, erläutert.

6.2.1.1 Informationsmodell

Die Wissensbasis von Spectrum basiert auf einem objektorientierten Modell. Die beiden wesentlichen Konzepte sind Klassen und Beziehungen (Relations) zwischen Klassen.

Klassen

Diese Klassen, im Spectrum-Jargon Model Types genannt, beinhalten zwei Arten von Wissen (vgl. Abbildung 6.4):

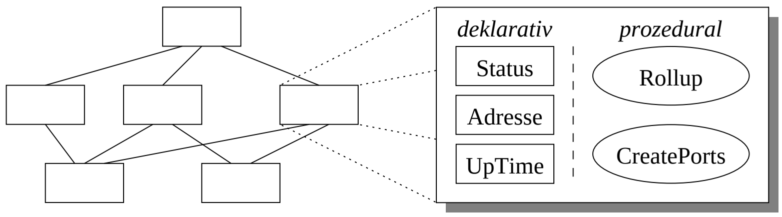


Abbildung 6.4: Klassenmodell von Spectrum

- Deklaratives Wissen
  beschreibt, welche Informationen die Klasse bereitstellt.

  Beispielsweise enthält ein Objekt einer Hubklasse unter anderem Informationen über die Anzahl und Art seiner Boards und Ports.

  Die Informationen sind durch eine Liste von Attributen definiert. Für jedes Attribut sind verschiedene Eigenschaften festgelegt, unter anderem:

  - Name und AttributeID
  - Flags (lesbar, schreibbar, extern, in einer Liste enthalten, ...)
  - Object Identifier, falls es sich um ein externes Attribut handelt
  - Defaultwert

- Prozedurales Wissen
  beschreibt, wie sich Objekte dieser Klasse verhalten.

  Unter anderem überwacht das oben erwähnte Hubobjekt den Zustand aller seiner Portobjekte und leitet daraus den Zustand des gesamten Geräts sowie die Erreichbarkeit der jeweils angeschlossenen Netzabschnitte ab.

  Prozedurales Wissen wird in Spectrum durch sogenannte Inference Handler implementiert, die in Abschnitt 6.2.1.2 beschrieben sind.

Spectrum unterstützt Mehrfachvererbung, einer Klasse können also mehrere Superklassen zugeordnet werden. Dabei ist zu berücksichtigen, daß

- Attribute, die von mehreren Klassen ererbt werden, nur einfach, nicht mehrfach in die aktuellen Klasse übernommen werden.

- die Inference Handler in einem umgekehrten Tiefensuchealgorithmus von unten nach oben (im Vererbungsgraph) aktiviert werden, wobei nur garantiert wird, daß die Inference Handler der Klasse selbst als erste instantiiert werden.

Eine Instanz eines Model Types wird in Spectrum als Model bezeichnet.

Relationen

Das zweite Konzept, auf dem Spectrums Wissensbasis aufbaut, sind Relationen, die die Abhängigkeiten zwischen je genau zwei Objekten beschreiben.

Mit solchen Relationen wird etwa modelliert, welche Geräte welchem Subnetz zugeordnet sind, welche Räume in welchen Gebäuden liegen, welche Ports zu einem Hub gehören und so fort.

Ein Instanz einer Relation zwischen zwei Objekten heißt Association. Da sich nicht aus allen Objekten und allen Relationen sinnvolle Associations ergeben, gibt es in Spectrum Regeln (Rules), die definieren, zwischen Objekten welcher Klassen (bzw. deren Subklassen) welche Relationen bestehen können.

6.2.1.2 Inference Handler

Das prozedurale Wissen eines Model Types wird in sogenannten Inference Handlern beschrieben, die asynchron auf Änderungen in Spectrums Wissensbasis reagieren.

Ein Inference Handler kann durch verschiedene Ereignisse ausgelöst werden, etwa das Anlegen und Löschen von Objekten der zugeordneten Klasse, Änderungen von Attributwerten, Anlegen und Löschen von Associations zwischen Objekten und Auftreten von Events.

In seinem Ablauf kann ein Inference Handler Models anlegen und löschen, Attribute lesen und schreiben, Events auslösen, Alarme erzeugen, modifizieren und löschen und Associations erzeugen und entfernen.

Untereinander kommunizieren Inference Handler mittels zweier Mechanismen:

Actions Actions kann man in etwa mit Methodenaufrufen von objektorientierten Programmiermodellen vergleichen. Ein solcher Action-Request enthält einen Integerwert zur Kennzeichnung der gewünschten Aktion sowie optional eine Reihe von Parametern. Wie der Inference Handler diese Daten auswertet, wird von Spectrum nicht vorgegeben, liegt also vollständig im Ermessen des Entwicklers.

Results Ein Inference Handler kann nach Abschluß seiner Arbeit ein Funktionsergebnis zurückgeben, das andere Inference Handler weiterverwenden können.

Programmtechnisch werden Inference Handler durch C++-Klassen implementiert. C++ bietet als objektorientierte Programmiersprache ein Vererbungskonzept an, das von Cabletron ausgiebig genutzt wurde.

So enthält das Inference Handler API bereits eine Hierarchie vordefinierter Klassen. An deren Spitze steht eine Basisklasse, die Methoden enthält, mit denen sich der Inference Handler für bestimmte Ereignisse registrieren kann. Daneben sind Klassen verfügbar, die zusätzliche Funktionalität bieten, etwa Summen- und Durchschnittsfunktionen sowie Methoden, die den konkurrierenden Zugriff mehrerer Inference Handler auf dasselbe Attribut koordinieren.

6.2.1.3 Modularisierung

Das Informationsmodell von Spectrum erlaubt es, überwachte Komponenten nicht nur durch ein, sondern durch mehrere Objekte in der Managementplattform darzustellen, die jeweils einen Aspekt des Geräts repräsentieren und untereinander durch Relationen verbunden sind.

Dieses Konzept wird im „Generic SNMP Device", einer Menge von Klassen, die als Ausgangspunkt zur Modellierung beliebiger Netzkomponenten verwendet werden kann, ausgiebig verwendet.

Zwei Beispiele für diesen Ansatz sind:

- Boards und Ports

  Die verwalteten Geräte werden ihrem Hardwareaufbau entsprechend in einzelne Objekte gegliedert. So erzeugt ein Inference Handler für einen Hub in der Managementplattform neben dem Hub-Model selbst für jedes Board und jeden Port je ein Model. Das Hub-Model ist mit allen Boards sowie jedes Board mit den zugehörigen Ports durch die HAS_PART-Relation verbunden.

- MIB-Teilbäume

  Der gesamte Internet-Baum wird in einzelne Gruppen, die sogenannten Major Applications, unterteilt. Jede dieser Major Applications kann wiederum in Minor Applications gegliedert werden. Wird nun ein Model zur Überwachung einer Komponente angelegt, prüft ein Inference Handler, welche dieser Gruppen in den von dem Gerät unterstützten MIB-Teilbäumen enthalten sind, und kreiert selbständig die zugehörigen Application-Objekte. Major Applications sind mit dem Geräte-Model durch die Manages- und mit ihrem Minor Applications durch die Provides-Relation verbunden.

  Die in Abbildung 6.5 dargestellte Gliederung eines Routers weist sechs Major Applications auf:

  - Token Ring und Ethernet für die MIB-Gruppen, die die Ports des Routers beschreiben. Für jeden Port wird jeweils eine Minor Application erzeugt.

– Data Link Switch, Generic Bridge und Routing, um die Funktion des Gerätes zu beschreiben. Der Bridge-Bereich wird in Minor Applications für die verschiedenen Algorithmen zur Wegewahl gegliedert, während bei Routing die unterstützten Protokolle unterschieden werden.
– MIB-II beschreibt einige Gruppen, die Systemdaten und Verkehrsstatistiken enthalten.

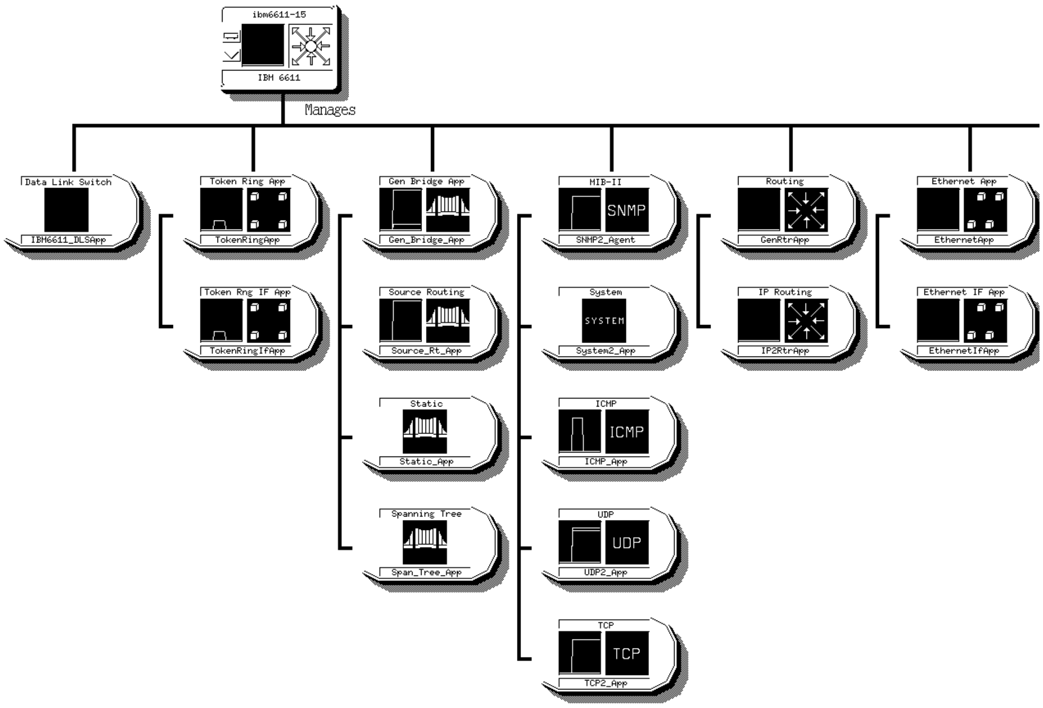


Abbildung 6.5: MIB-Gruppen eines Routers (aus [DM94])

Der Vorteil gegenüber der monolithischen Integration eines Gerätes ist, daß eine spezieller Board- oder Porttyp oder eine bestimmte MIB-Gruppe nur ein einziges Mal modelliert werden muß. Für jede Komponente, die das jeweilige Merkmal besitzt, kann der jeweilige Model Type wiederverwendet werden. Cabletron bietet bereits entsprechende Model Types für den größten Teil der MIB-II an.

Da das Generic SNMP Device zur Laufzeit automatisch die Fähigkeiten des überwachten Gerätes auswertet und entsprechende Models instantiiert, beschränkt sich der Entwicklungsaufwand für die Integration einer neuen Komponente auf die Modellierung derjenigen Merkmale, die noch nicht in der

Plattform verfügbar sind. In der Wissenbasis von Spectrum sind entsprechende Klassen enthalten, von denen eigene Major bzw. Minor Applications abgeleitet werden können.

6.2.2 Architektur

Um eine möglichst flexible Nutzung von Spectrum zu erlauben, wurde die Funktionalität der Managementplattform auf verschiedene Programme verteilt, die nach dem Client-Server-Prinzip zusammenarbeiten. Der Baustein für die Kommunikation mit den Agenten ist im SpectroSERVER realisiert, während die Bedienoberfläche auf verschiedene Clients ausgelagert wurde, wobei SpectroGRAPH das am meisten verwendete Tool ist. Die eigentliche Funktionalität der Managementplattform verteilt sich auf Server und Clients.

Der Vorteil dieser Architektur liegt nicht zuletzt darin, daß mehrere Benutzer mit ihren Clients denselben SpectroSERVER benutzen (können), so daß einerseits weniger Netzlast von und zu den überwachten Komponenten erzeugt wird und andererseits eine unnötige Replikation der Managementinformation im Netz verhindert wird.

Abbildung 6.6 gibt einen Überblick über die Architektur der Managementplattform Spectrum.

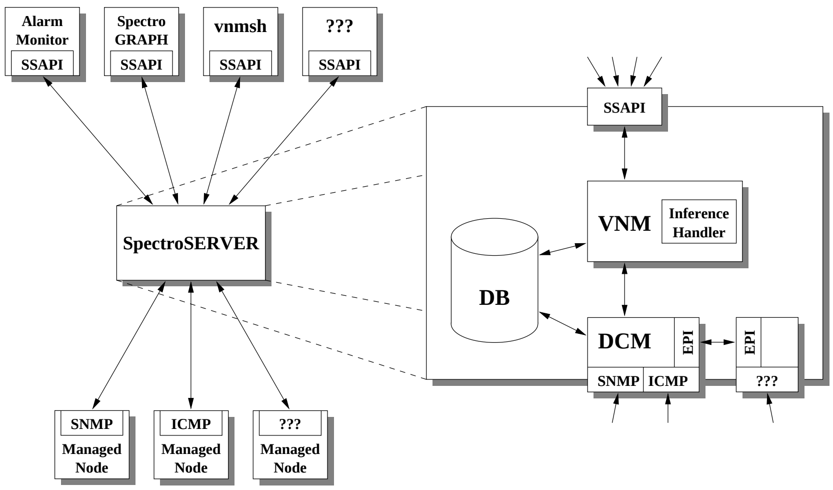


Abbildung 6.6: Architektur von Spectrum (nach [SpecCon])

Der SpectroSERVER setzt sich aus mehreren Modulen zusammen.

- Das Herzstück des Servers ist die Virtual Network Machine (VNM). Alle verwalteten Objekte werden in der VNM durch ein oder mehrere

Models repräsentiert. Anfragen nach Attributwerten beantwortet die VNM soweit möglich aus ihren eigenen Speicherbereichen, ansonsten wird die Anfrage transparent für die höheren Schichten an die Datenbank oder den DCM weitergeleitet. Mit den verschiedenen Clients kommuniziert die VNM über das SpectroSERVER API (SSAPI).

Außerdem enthält die VNM die verschiedenen Inference Handler, die die Funktion und Interaktion der verschiedenen Models implementieren. Mit diesen Programmodulen tauscht die VNM über das Inference Handler API (IHAPI) Daten aus. Für das Scheduling der einzelnen Inference Handler existiert innerhalb der VNM eine Taskkontrollinstanz namens Modular Object Oriented Threads (MOOT), die kooperatives Multitasking der Inference Handler innerhalb des SpectroSERVER-Prozesses erlaubt. Es wird garantiert, daß ein Inference Handler nicht unterbrochen wird, es sei denn, er gibt die Kontrolle ab (was implizit auch beim Zugriff auf externe Attribute, deren Wert vom DCM ermittelt werden muß, geschieht).

- Die Datenbank enthält alle statischen Informationen wie Model Types, Relationships und Rules. Weiterhin sind Informationen über die aktuell instanziierten Models und Associations sowie der Inhalt verschiedener Attribute, soweit das vom Entwickler vorgesehen ist, enthalten.
- Der Device Communication Manager ist für die Kommunikation des SpectroSERVER mit den Komponenten des Netzes zuständig. Seine Aufgabe ist es, die Anfragen der VNM in das von der jeweiligen Managed Node verwendete Protokoll zu übersetzen. Der DCM selbst unterstützt die Protokolle SNMP und ICMP, für die Einbindung weiterer Protokolle existiert das External Protocol Interface, über das neue Kommunikationsmodule (Management Station Access Provider, MSAP) angebunden werden.

Für die Interaktion mit dem Benutzer existieren verschiedene Clients:

- SpectroGRAPH

  SpectroGRAPH ist eine auf Motif basierende Benutzeroberfläche für Spectrum. Die verschiedenen Objekte werden durch graphische Objekte, sogenannte Icons, dargestellt. Jedes Icon besitzt einige sensitive Bereiche für Mausklicks und ein Menü, über die verschiedene objektspezifische Views erreicht und Aktionen ausgelöst werden können. Associations wie connects_to und Manages werden durch Verbindungslinien dargestellt, Associations wie contains und HAS_PART durch Viewhierarchien.

Beispielsweise wird innerhalb eines größeren Netzes ein Ethernet-Subnetz als Icon dargestellt. Durch Anklicken eines Feldes dieses Icons wird ein neuer View geöffnet, in dem alle Komponenten innerhalb dieses Subnetzes angezeigt werden, wobei die Verkabelungsstruktur durch Linien zwischen diesen Geräten visualisiert wird.

Eine Reihe von Views sind fest im SpectroGRAPH implementiert. Zu diesen gehört etwa der Attribute Walk zum Anzeigen aller Attribute eines Models, die Views zur Überwachung von Alarmen und Events, Device View und Application View (siehe Abschnitt 6.2.1.1).

Alle anderen (sogenannten generischen) Views sind durch eine einfache Viewbeschreibungssprache, die SpectroGRAPH interpretiert, beschrieben. Für die Darstellung dieser Views bietet SpectroGRAPH eine große Anzahl von Bausteinen, deren Spektrum von einfachen Integer- und Textfeldern über Enumerations und Tabellen bis zu Tortendiagrammen und Multi-Attribute-Graphs reicht.

- vnmsh

  Hier handelt es sich um eine Reihe von shell-basierten Programmen, die sich zum Einsatz in Skripten eignen. Es existieren Kommandos für den Auf- und Abbau der Verbindung zum SpectroSERVER, zum Kreieren, Löschen, Anzeigen und Modifizieren von Models, Associations und Alarmen sowie zum Erzeugen von Events.

- Alarm Monitor

  Mit diesem Tool kann der Anwender automatisch mittels eines Shellscripts Aktionen ausführen lassen, wenn in der VNM ein Alarm erzeugt, modifiziert oder entfernt wird.

- Weitere Tools

  Da Cabletron das SpectroSERVER API zur Verfügung stellt, können auch weitere (fremdentwickelte) Clients die Dienste des SpectroSERVERs nutzen.

6.2.3 Entwicklungswerkzeuge

Die Werkzeuge, die in der Entwicklungsumgebung von Spectrum enthalten sind, gliedern sich in zwei Gruppen:

Level-I-Tools erlauben es, Managementmodule für Spectrum zu erstellen, ohne C++-Code erstellen zu müssen.

Im einzelnen handelt es sich um

- den MTE für die Modellierung der Wissensbasis. Mit diesem Werkzeug können Klassenhierarchie, Attributdefinitionen, Relationen und Regeln modifiziert und erweitert werden.
- den IIB-Editor für die Erstellung von Icons und ihre Zuordnung zu den verschiedenen Views.
- den GIB-Editor für die Erstellung von generischen Views.
- den CSI-Editor, ein Malprogramm, mit dem die Bilder, die in Icons und Views verwendet werden, erstellt werden.
- das Extension Integration Toolkit, einige Scripts, die das Erstellen des installationsfertigen Managementmoduls erleichtern.

Desweiteren wird ein Texteditor benötigt, mit dem Event- und Alarmkonfiguration und -texte erstellt werden.

Für eine detaillierte Beschreibung des Level-I-Toolkits sei auf [Dem95] verwiesen.

Level-II-Tools umfassen die verschiedenen Programmierschnittstellen, die Spectrum anbietet.

- MSAP-API, SSAPI und IHAPI wurden bereits in Abschnitt 6.2.2 beschrieben.
- Das View API ist eine Schnittstelle, die vom SpectroGRAPH angeboten wird und mit der es möglich ist, neue Views, für die die Fähigkeiten der generischen Views nicht ausreichen, in Spectrum einzubinden.

Dieses Toolkit umfaßt Headerdateien, Libraries und Beispielcode für jedes der genannten APIs. Eine ausführlichere Beschreibung ist in [Kel93] zu finden.

Kapitel 7

Spezifikation des Prototypen

In diesem Kapitel wird zunächst ein allgemeines Modell für eine Managementlösung vorgestellt, das anschließend für das gegebene Entwicklungsumfeld verfeinert wird. Desweiteren werden Ansätze für ein Ausbau des Prototypen und die damit verbundenen Probleme aufgezeigt.

7.1 Modellierung

7.1.1 Klassen- und Objektmodell

Mit der Network Services Monitoring MIB (Abschnitt 4.3.1) wurde ein Baukastensystem zur Modellierung verteilter Anwendungen eingeführt. Diese MIB selbst stellt den Basisbaustein dar, der die grundlegenden Informationen, die alle verteilten Anwendungen gemeinsam haben, bereitstellt. Durch Hinzunahme weiterer MIB-Bausteine kann eine Erweiterung in Richtung applikations- und herstellerspezifischer Informationen erfolgen. Beispielsweise bietet die in Abschnitt 4.3.2 vorgestellte Mail Monitoring MIB zusätzliche MTA-spezifische Informationen, und es wäre denkbar, daß ein Hersteller eines solchen Produkts diese um weitere produktspezifische MIB-Bausteine ergänzt.

Es bietet sich an, innerhalb einer objektorientierten Plattform einen ähnlichen, generischen Ansatz zu verfolgen, da Parallelen zu dem dort vorhandenen Vererbungskonzept exisitieren.

Es wird eine generische Klasse für Applikationen eingeführt. Sie repräsentiert die Basisinformation und -funktionalität, die allen verteilten Anwendungen gemeinsam ist. Diese Basisklasse ist Wurzel einer Vererbungshierarchie, wobei jede Subklasse einer Klasse zusätzliche applikations- oder herstellerspezifische Information und/oder Funktionalität beinhaltet.

Abbildung 7.1 zeigt eine solche Hierarchie. Wurzel ist die Basisklasse Application. Von dieser wird die Klasse MTA abgeleitet, die MTA-spezifische Information und Funktionalität bereitstellt und Superklasse der produktspezifischen Klasse XT-PP ist. Entsprechende Teilbäume existieren für Directory System Agents und Datenbankserver.

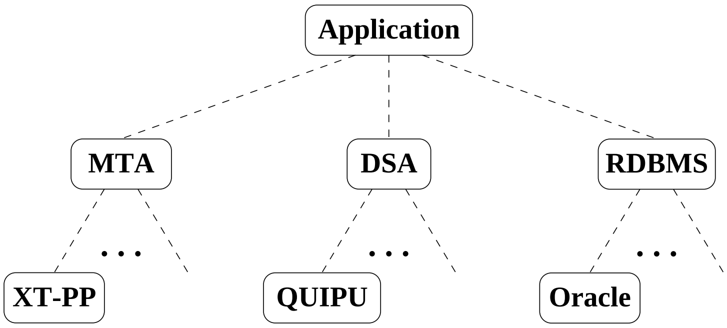


Abbildung 7.1: Beispiel einer Klassenhierarchie

7.1.2 Kritierien zur Zuordnung von Applikationen zu Objekten und Klassen

Da für die Indizierung der einzelnen Applikationen innerhalb der Network Services Monitoring MIB keine Regeln existieren, kann an Hand dieser Indizes nicht ermittelt werden, welcher Klasse eine auf dem Agentensystem ablaufende Applikation angehört. Es gilt also, Kriterien zu finden, mit denen eine Applikation bzw. ihr Index in der MIB einer Klasse zugeordnet werden kann.

Im folgenden werden drei Kriterien verwendet, um die zur Applikation passende Klasse zu ermitteln.

- Für jede Applikations-Klasse innerhalb der Managementplattform wird ein Attribut angegeben, dessen Existenz im Agentensystem notwendig ist für die Zuordnung zu dieser Klasse. Existiert also dieses Attribut (erweitert um den Index der Applikation) auf dem Agentensystem oder ist kein solches Attribut innerhalb der Klasse angegeben, ist dieses Kriterium erfüllt.

- Für jede Applikations-Klasse wird eine Zeichenkette angegeben. Ist diese Zeichenkette Substring des Namens der Applikation (applName) oder ist diese Zeichenkette leer, ist dieses Kriterium erfüllt.
- Da durch Anwendung der beiden ersten Kriterien eventuell die passende Klasse nicht eindeutig bestimmt werden kann, wird zusätzlich eine Priorität eingeführt. Existieren also mehrere passende Klassen, wird diejenige mit der höchsten Priorität ausgewählt. Sollten mehrere Klassen mit höchster Priorität existieren, wird eine davon nichtdeterministisch ausgewählt.

7.1.3 Klassifizierung von Applikationen und Objekten

Eine weitere Klassifizierung der zu verwaltenden Applikationen ist die Einordnung in permanent verfügbare und temporär verfügbare Applikationen.

Temporär verfügbare Applikationen werden von Anwender gestartet und terminieren, sobald sie ihre Aufgabe erfüllt haben. Ein Datenbank-Client ist ein Beispiel für diese Klasse. Innerhalb der Managementplattform wird beim Start der Applikation dynamisch ein entsprechendes temporäres Objekt erzeugt, das beim Beenden der Applikation wieder entfernt wird.

Permanent verfügbare Applikationen sind solche, die im fehlerfreien Fall ohne Unterbrechung auf dem jeweiligen Rechner zur Verfügung stehen sollen. In diese Gruppe fällt beispielsweise ein MTA. Existiert auf einem Rechner, für den ein MTA konfiguriert ist, keine entsprechende Anwendung, liegt ein Fehlerzustand vor. Permanent verfügbare Applikationen werden innerhalb der Managementplattform durch permante Objekte repräsentiert, diese werden also im Gegensatz zu temporären Objekten nicht entfernt, wenn die entsprechende Applikation nicht mehr zur Verfügung steht.

Permanente Objekte können weiter untergliedert werden. Ein permantes Objekt heißt gebunden, wenn die zugehörige Applikation (d.h. die zugehörige Zeile in der MIB) verfügbar ist. Freie permanente Objekte besitzen keine Entsprechung in einer laufenden Applikation auf dem Agentensystem, beispielsweise, weil diese aufgrund einer Fehlersituation nicht mehr zur Verfügung steht.

Da (wie bereits erwähnt) applIndex nicht fest einer bestimmten Applikation zugeordnet ist, ergibt sich allerdings ein Problem in der dauerhaften Zuordnung zwischen applIndex und dem zugehörigen permanenten Objekt in der Managementplattform. Würde beispielsweise eine Applikation angehalten und zu einem späteren Zeitpunkt neu gestartet, kann nicht garantiert werden, daß sie dieselbe Indexnummer erhält.

7.1.4 Abbildung der Applikationen auf Objekte

Im wesentlichen ist zu klären, welche Schritte innerhalb der Managementplattform stattfinden, wenn ein Applikation gestartet wird oder terminiert, also wenn innerhalb der Network Services Monitoring MIB ein neue Zeile erzeugt bzw. eine bestehende entfernt wird.

Wird auf dem Agentensystem eine neue Applikation gestartet, wird eine neue Zeile innerhalb der Network Services Monitoring MIB angelegt. Als Reaktion darauf kann die Plattform entweder ein freies permanentes Objekt an diese Applikation binden oder ein neues Objekt einer Applikations-Klasse kreieren. Daher wird eine Liste all dieser Objekte und Klassen erzeugt. Aus dieser wird an Hand der in Abschnitt 7.1.2 vorgestellten Kriterien das passende Objekt bzw. die passende Klasse ausgewählt, wobei die dritte Regel dahingehend modifiziert wird, daß bei gleicher Priorität einem freien permanenten Objekt der Vorzug gegenüber einem neu zu erzeugenden Objekt gegeben wird.

Wurde ein vorhandenes freies permanentes Objekt ausgewählt, wird es an den neuen Applikationsindex gebunden, ansonsten wird ein neues Objekt der in diesem Fall ausgewählten Klasse erzeugt und ebenfalls an den Index gebunden.

Terminiert eine Applikation auf dem Anwendungssystem, verschwindet also der entsprechende Eintrag aus der MIB, wird das an diesen Index gebundene Objekt gelöscht, sofern es sich um ein temporäres Objekt handelte, andernfalls wird nur die Bindung aufgehoben, das gebundene permanente Objekt wird somit zum freien Objekt.

7.1.5 Funktionalität einer generischen Klasse für Applikationen

Ein Objekt dieser Klasse ermittelt aus dem Bindungsstatus, dem Antwortverhalten des Agenten und dem Inhalt der MIB-Variable applOperStatus den aktuellen Funktionsstatus der Applikation, stellt diesen dar und benachrichtigt bei Zustandsänderungen ggf. den Verwalter.

Außerdem wird ein Log geführt, in dem immer dann ein Eintrag vorgenommen wird, wenn sich die Anzahl der abgewiesenen Verbindungswünsche erhöht.

7.1.6 Funktionalität einer MTA-Klasse

Da ein MTA eine permanent verfügbare Applikation ist, wird innerhalb der Managementplattform ein entsprechendes permanentes Objekt kreiert.

Geht dieses Objekt vom gebundenen Zustand in den freien Zustand über, d.h. steht der MTA auf dem Agentensystem nicht mehr zur Verfügung, wird ein Alarm ausgelöst und so das Betriebspersonal benachrichtigt. Wird zu einem späteren Zeitpunkt das Objekt wieder an einen MTA gebunden, wird der Alarmzustand aufgehoben.

Für jeden Channel des MTAs werden drei Variablen mtaGroupStoredMessages, mtaGroupStoredVolume und mtaGroupOldestMessageStored überwacht. An Hand von vom Benutzer konfigurierbaren Schwell- und Rücksetzwerten werden Überlastungs- und Fehlersituationen erkannt und gemeldet.

Speziell für XT-PP ist eine Oberflächenintegration der Nexor-Tools in eine Managementplattform vorgesehen. Es soll möglich sein, aus dieser heraus das X11-basierte Managementtool MTAconsole aufzurufen.

7.2 Implementierung in Spectrum

Da die Managementplattform Spectrum bereits ein objektorientiertes Konzept anbietet, kann das in den letzten Abschnitten entwickelte Modell ohne große Modifikationen auf den Prototyp übertragen werden. Aus der Klassenhierarchie werden entsprechend der Aufgabenstellung nur die Klassen auf dem Pfad von der generischen Klasse für verteilte Anwendungen zur produktspezifischen Klasse für Nexors XT-PP implementiert.

Eine zentrale Frage für die Konzeption des Prototypen ist, welche Instanz innerhalb der Managementplattform für das automatische Erzeugen und Entfernen der Objekte, die die verteilten Anwendungen auf dem Agentensystem repräsentieren, zuständig ist.

Die technisch sinnvollste Lösung wäre, diejenige Basisklasse um diese Funktionalität zu erweitern, von der die Klassen für die verschiedenen Komponenten im Netz, speziell die Workstationklassen, abgeleitet sind. Allerdings läßt sich diese Vorgehensweise im Rahmen der Erstellung des Prototypen nicht verwirklichen, da ein entsprechender Zugriff auf diese Klassen Cabletron vorbehalten ist.

Daher wird die neue Klasse Application Master Model Type eingeführt, die im wesentlichen die beschriebene Funktionalität implementiert. Damit diese Funktion auf jedem Rechensystem bereitgestellt werden kann, muß dafür

Sorge getragen werden, daß für jedes Rechnerobjekt automatisch ein entsprechendes Application-Master-Objekt erzeugt wird. Zu diesem Zweck wird der Application Master Model Type als Major Application in Spectrums Generic SNMP Device eingebunden.

Neben diesen instantiierbaren Klassen werden noch eine Reihe von internen, nicht instantiierbaren Klassen definiert, um die Erweiterung der Managementlösung zu vereinfachen und den Modellierungsrichtlinien von Cabletron zu entsprechen.

Einen Überblick über die Klassenhierarchie gibt Abbildung 7.2. Klassen, die bereits von Spectrum vorgegeben sind, sind grau dargestellt, Klassen, die instantiiert werden können, sind fett umrandet.

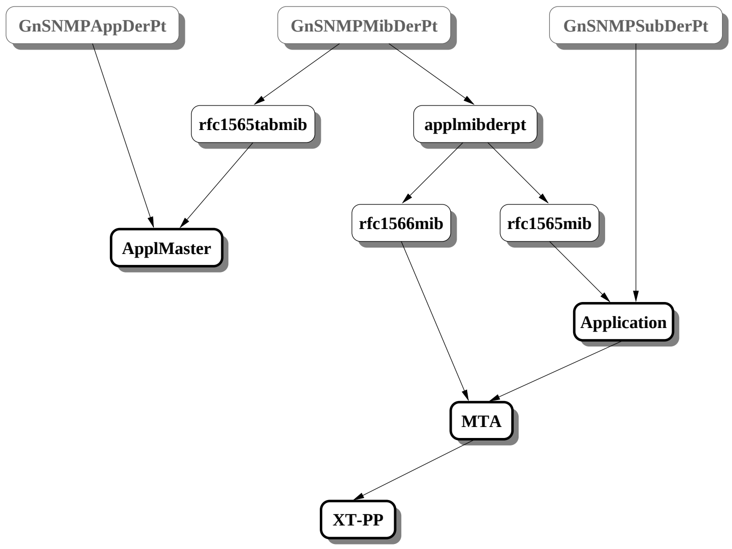


Abbildung 7.2: Klassen des Prototypen im Überblick

Abbildung 7.3 zeigt ein Beispiel, wie sich ein MTA-Objekt in das Application-Konzept des Generic SNMP Device einordnet (vgl. auch Abbildung 6.5).

7.2.1 Model Types

Eine Reihe von Attributen (etwa Developer und Revision) werden in allen Model Types gleichermaßen gesetzt. Die zugehörigen Werte ergeben sich aus dem Entwicklungsumfeld, weshalb auf diese Attribute im weiteren nicht näher eingegangen wird.

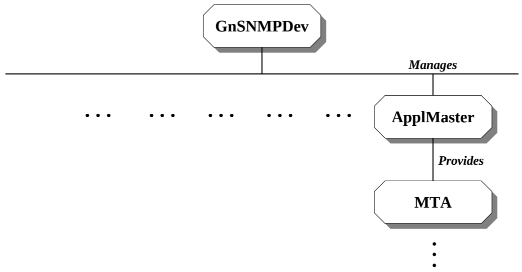


Abbildung 7.3: Einordnung der Objekte in das Generic SNMP Device

7.2.1.1 rfc1565tabmib

Beschreibung:
Diese Klasse dient als Container für die Network Services Monitoring MIB und dient als Basisklasse für den ApplMaster Model Type.

Basisklassen:
GnSNMPMibDerPt (Ableitungspunkt für MIB-Container)

Attribute:
Sämtliche Variablen der Network Services Monitoring MIB werden eingelesen.

Inference Handler:
keine

7.2.1.2 ApplMaster

Beschreibung:
Für jeden Rechner, auf dem ein Agent installiert ist, der die Network Services Monitoring MIB unterstützt, wird ein Objekt dieser Klasse erzeugt. Dieses Objekt liefert einen Überblick über alle auf diesem Rechner ablaufenden Applikationen und ist für das Erzeugen, Löschen und Zuweisen von applikationsspezifischen Objekten zuständig.

Diese Klasse stellt innerhalb des Generic SNMP Device eine Major Application dar.

Basisklassen:
rfc1565tabmib, GnSNMPDevAppDerPt (Ableitungspunkt für Major Applications)

Attribute:

Attribute, deren Defaultwert geändert wird:

Name	Vorgabewert
default_attr	ID von applIndex

Durch diese Belegung von default_attr wird vom Generic SNMP Device automatisch eine Instanz dieser Klasse erzeugt, wenn das Agentensystem die Network Service Monitoring MIB unterstützt.

Inference Handler:

Außerdem wird ein Inference Handler installiert:

CsIHGenAppl

Dieser Inference Handler liest periodisch die Tabelleneinträge in der Spalte applIndex aus. Er vergleicht dabei die ermittelten, auf dem Agentensystem vorhandenen Indizes mit den Indizes, die den von ihm verwalteten (mittels der Provides-Relation angebundenen) Models zugewiesen sind.

Wird in der MIB ein neuer Eintrag entdeckt, wird an Hand des in Abschnitt 7.1.4 beschriebenen Algorithmus bestimmt, ob ein passendes Model existiert oder ob ein solches erst noch kreiert werden muß. Im ersten Fall wird in diesem Model die Indexnummer in Appl_Index eingetragen, im zweiten Fall wird ein Model der passenden Klasse kreiert, der Indexeintrag vorgenommen und die Association Provides vom Master sowie PossPrimApp vom Gerätemodel zu diesem Model eingetragen.

Ist andererseits eine Applikation, an die ein Model der Plattform gebunden ist, auf dem Agentensystem nicht mehr verfügbar, wird abhängig vom Attribut Appl_Permanent entweder das Model gelöscht oder die Indexnummer entfernt.

Zusätzlich wird dieser Inference Handler von Veränderungen innerhalb der Provides-Relation getriggert, die etwa auftreten, wenn der Benutzer ein Model von Hand löscht.

7.2.1.3 applmibderpt

Beschreibung:

In dem Teilbaum, der unterhalb dieser Klasse liegt, werden Klassen für verschiedene Typen von Applikationen definiert. Jedes Objekt einer solchen Klasse repräsentiert genau eine Applikation auf dem Agentensystem. Da die einzelnen Applikationen in der Network Services Monitoring MIB in einer Tabelle abgelegt sind, ist ein Attribut notwendig, in dem der applIndex der zugeordneten Applikation abgelegt wird und das bei allen Zugriffen auf den Agenten als (Teil-)Index verwendet wird.

Aufgabe von applmibderpt ist es, neben den Eigenschaften des generischen MIB-Ableitungspunkts dieses Attribut zur Verfügung zu stellen.

Basisklassen:
GnSNMPMibDerPt

Attribute:
Neue Attribute:

Name	Typ	Vorgabewert
Appl_Index	Object Identifier	

Inference Handler:
keine

7.2.1.4 rfc1565mib

Beschreibung:
Diese Klasse dient als Container für die Network Services Monitoring MIB und als Basisklasse für den Application Model Type.

Im Unterschied zu rfc1565tabmib (Abschnitt 7.2.1.1) werden alle externen Attribute durch Appl_Index referenziert.

Basisklassen:
applmibderpt

Attribute:
Sämtliche Variablen der Network Services Monitoring MIB werden eingelesen. Bei all diesen Attributen wird als Indexreferenz ein Verweis auf Appl_Index eingetragen. Bei den Attributen der Tabellen, die nur durch applIndex indiziert sind, wird zudem das List-Flag (Attribut ist Eintrag einer Liste) entfernt.

Inference Handler:
keine

7.2.1.5 Application

Beschreibung:
Application ist die Basisklasse für alle Klassen, deren Objekte jeweils

einzelne Anwendungen repräsentieren, wobei bereits einige Basisfunktionen bereitgestellt werden. Objekte dieser Klasse werden für die Anwendungen generiert, für die keine spezifische Klasse existiert.

Application (und alle Subklassen) stellen Minor Applications innerhalb des GnSNMPDev dar.

Basisklassen:

rfc1565mib, GnSNMPSubDerPt (Ableitungspunkt für Minor Applications)

Attribute:

Neue Attribute:

Name	Typ	Vorgabewert
Appl_Attr	Attribute ID	
Appl_Name	String	
Appl_Prio	Integer	0
Appl_Permanent	Boolean	FALSE

Die ersten drei Attribute werden für den in Abschnitt 7.1.4 beschriebenen Zuordnungsalgorithmus benötigt, das vierte Attribut zeigt an, ob die Applikation permanent ist.

Inference Handler:

Der Klasse Application werden zwei Inference Handler zugeordnet.

Status

Dieser Inference Handler ist für das Erstellen und Löschen von Alarmen zuständig, wenn sich applOperStatus oder der Bindungsstatus (also der Inhalt von Appl_Index) ändert.

Audit

Erhöht sich die Anzahl der von außen initiierten, aber abgewiesenen Verbindungen, wird ein entsprechender Event erzeugt.

7.2.1.6 rfc1566mib

Beschreibung:

Diese Klasse dient als Container für die Mail Monitoring MIB und als Basisklasse für den MTA Model Type.

Basisklassen:

applmibderpt

Attribute:

Sämtliche Variablen der Mail Monitoring MIB werden eingelesen. Bei all diesen Attributen wird als Indexreferenz ein Verweis auf Appl_Index eingetragen. Bei den Attributen der Tabellen, die nur durch applIndex indiziert sind, wird zudem das List-Flag (Attribut ist Eintrag einer Liste) entfernt.

Inference Handler:

keine

7.2.1.7 MTA

Beschreibung:

Die MTA-Klasse erbt die Funktionalität von Application und berücksichtigt zudem die MTA-spezifischen Erweiterungen der Mail Monitoring MIB.

Basisklassen:

rfc1566mib, Application

Attribute:

Neue Attribute:

Name	Typ	Vorgabewert
Number_Thres	String	
Number_Reset	String	
Volume_Thres	String	
Volume_Reset	String	
Age_Thres	String	
Age_Reset	String	

In diesen Variablen werden die Schwellwerte auf Anzahl und Volumen der Nachrichten und das Alter der ältesten Nachricht in der Queue für das Setzen und Rücksetzen von Alarmen definiert. Da Spectrum keine internen Tabellen unterstützt, werden die Daten als durch Kommas getrennte Folge von (Substring,Wert)-Paaren abgelegt.

Attribute, deren Defaultwert verändert wird:

Name	Vorgabewert
Appl_Attr	ID von mtaReceivedMessages
Appl_Prio	1
Appl_Permanent	TRUE

Inference Handler:

Der Model Type MTA enthält einen Inference Handler:

CsIHPerformance

Dieser Inference-Handler vergleicht periodisch mtaGroupStored-Messages, mtaGroupStoredVolume und mtaGroupOldestMessage-Stored jedes Channels mit den Threshold- und Reset-Werten und setzt bzw. löscht entsprechende Performance-Alarme.

Um die zu einem Channel gehörenden Threshold- bzw. Reset-Werte zu bestimmen, wird die Liste der (Substring,Wert)-Paare der jeweiligen Variablen durchlaufen, bis ein Substring gefunden wird, der in mtaGroupName dieses Channels enthalten ist.

7.2.1.8 XT-PP

Beschreibung:

Diese Klasse dient ausschließlich zur Oberflächenintegration der verschiedenen proprietären Tools von Nexor.

Basisklassen:

MTA

Attribute:

Neue Attribute:

Name	Typ	Vorgabewert
password	String	

password enthält das Passwort für den console-Zugriff auf diesen MTA.

Attribute, deren Defaultwert geändert wird:

Name	Vorgabewert
Appl_Name	qmgr
Appl_Prio	2

Inference Handler:

keine

7.2.2 Rules

Die Regel „ApplMaster Provides Application“ (One-to-Many) wird eingetragen. Die Manages-Regel zwischen GnSNMPDev und ApplMaster muß nicht explizit definiert werden, sie wird vom GnSNMPDevAppDerPt geerbt.

7.2.3 Icons

Für die vier instantiierbaren Klassen ApplMaster, Application, MTA und XT-PP muß jeweils ein Icon erstellt werden. Hierfür wird das Standard-Icon für Applications verwendet, das in Abbildung 7.4 dargestellt ist.

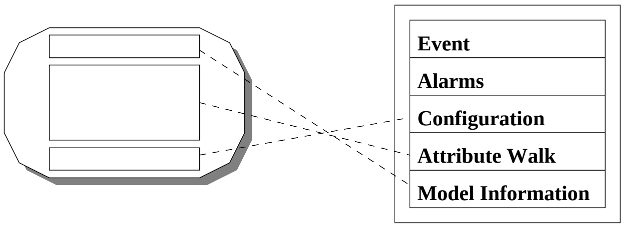


Abbildung 7.4: Icon mit Menü

Über das Menü erreicht man die internen Views Event View, Alarm View und Attribute Walk View, über den auf alle Attribute des Models lesend zugegriffen werden kann. Außerdem können die beiden generischen Views Model Information View und Configuration View, ein klassenspezifischer View, erreicht werden.

Die drei zuletzt genannten Views können auch über die sensitiven Felder des Icons erreicht werden. Das mittlere Feld wird je nach Zustand des Models unterschiedlich eingefärbt, wobei die Farbpalette von grün (korrekte Funktion) über gelb und orange bis rot (Totalausfall) reicht.

Im Menü des XT-PP-Icons wird zusätzlich zu den beschriebenen Einträgen ein Menüpunkt „Console“ eingefügt. Durch Anwählen dieses Menüpunkts wird ein Shellscript gestartet, dem der Name des Rechners sowie der Wert des password-Attributs übergeben wird und das die MTAconsole bzw. die lconsole (vgl. Abschnitt 6.1.3) startet.

7.2.4 Views

Für jede der vier instantiierbaren Klassen müssen die generischen Views beschrieben werden.

Der „Model Information View“ wird in allen Fällen von GnSNMPDev übernommen. Über diesen View sind Verwaltungsinformationen wie Modelname, Agentenadresse, Standort, SysUpTime, Kommunikationsprotokoll und Pollingstatus erreichbar.

Sämtliche im folgenden spezifizierten generischen Views werden mit einem sogenannten Banner ausgestattet, wie er in Abbildung 7.5 dargestellt ist.

Hier werden System- und Anwendungsdaten wie Name und Laufzeit sowie Kurzbeschreibung und verantwortliche Person abgelegt. Der Rahmen um diesen Informationsbereich wechselt seine Farbe je nach Erreichbarkeit des Agenten auf grün oder rot. In den Abbildungen der Views wird der Banner jeweils durch ein dickumrandetes Rechteck dargestellt.

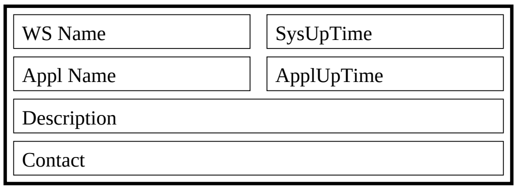


Abbildung 7.5: Banner-Information

7.2.4.1 ApplMaster

ApplMaster besitzt nur einen generischen View, den Applications View (vgl. Abbildung 7.6), der eine Tabelle aller von diesem Model verwalteten Anwendungen enthält. Da keine einzelne Applikation angesprochen wird, werden aus dem Banner dieses Views die Einträge ApplName und ApplUpTime entfernt.

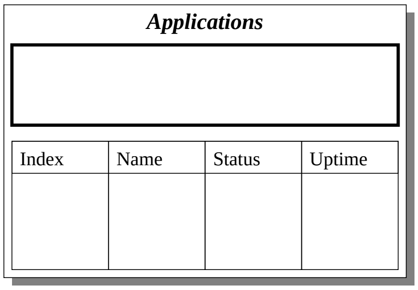


Abbildung 7.6: Der Applications View

7.2.4.2 Application

Für den Model Type Application werden drei zusätzliche generische Views angelegt (vgl. Abbildung 7.7).

- Der Application View, der vom Icon aus erreicht werden kann, enthält Name, Version und Statusdaten der Anwendung. Durch zwei Buttons können die anderen beiden Views erreicht werden.

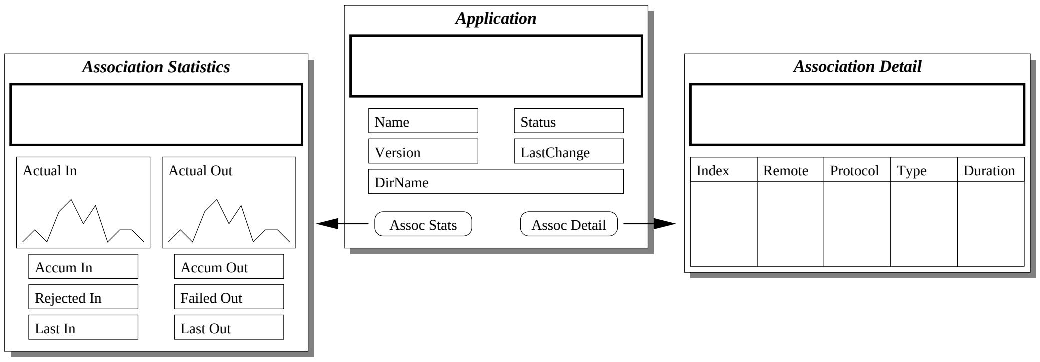


Abbildung 7.7: Generische Views eines Application Models

- Der Association Statistics View enthält verschiedene Zähler für ein- und ausgehende Verbindungen, wobei die aktuellen Verbindungszahlen als Graph dargestellt werden.
- Im Association Detail View sind alle z.Zt. bestehenden Verbindungen in einer Tabelle enthalten, in der Daten wie Verbindungspartner und Verbindungsdauer angezeigt werden.

7.2.4.3 MTA

Dem Model Type MTA sind die sieben in Abbildung 7.8 dargestellten generischen Views zugeordnet.

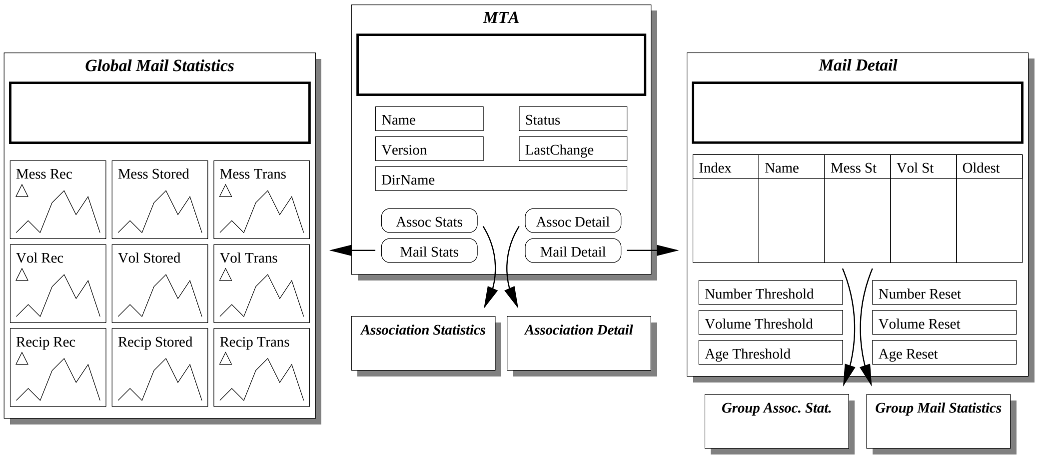


Abbildung 7.8: Generische Views eines MTA Models

- Beim MTA View handelt es sich im wesentlichen um den in Abschnitt 7.2.4.2 vorgestellten Application View, der um zwei Buttons zum Erreichen weiterer MTA-spezifischer Views erweitert wurde.

- Association Statistics View und Association Detail View werden unverändert vom Model Type Application übernommen.

- Der Global Mail Statistics View zeigt Anzahl, Volumen und Empfängerzahl der empfangenen, versandten und zwischengespeicherten Nachrichten an. Für die Visualisierung der beiden ersten Kategorien wird jeweils ein Delta-Graph, der die Änderung eines Zähler aufzeichnet, verwendet, während die Belegung der Queue in absoluten Werten dargestellt wird.

- Im Mail Detail View wird eine Übersicht über die einzelnen Gruppen, aus denen der MTA besteht, gegeben. Neben dem Namen der Gruppe sind die in Abschnitt 7.1.6 angegebenen, speziell zu überwachenden Variablen aufgeführt. Durch Anklicken eines Tabellenfeldes wird der zur jeweiligen Gruppe gehörende Group Association bzw. Group Mail Statistics View geöffnet. Außerdem können in diesem View die in Abschnitt 7.2.1.7 definierten Threshold- und Reset-Werte modifiziert werden.

- Der Group Mail Statistics View ist vom Aufbau her identisch zum Global Mail Statistics View, wobei der Banner um Gruppenname und -index ergänzt wird. Allerdings werden hier die Werte der jeweiligen Gruppe, nicht die des gesamten MTAs, dargestellt.

- In gleicher Weise wird der Group Association Statistics View aus dem Association Statistics View konstruiert, wobei hier zusätzlich die letzte Fehlerursache sowie das Intervall bis zum nächsten Aufbauversuch enthalten sind.

7.2.4.4 XT-PP

Dieser Model Type übernimmt sämtliche Views größtenteils unverändert von MTA, einzig der MTA View wird in XT-PP View umbenannt sowie um ein Feld zur Eingabe des Passworts für die proprietären Managementtools ergänzt.

7.2.5 Events und Alarme

Es werden die in Tabelle 7.1 aufgeführten Events und Alarme definiert.

Model Type	Art	Grund
Application	Event & Alarm	Application is congested
Application	Event & Alarm	Application is down
Application	Event	Number of failed inbound associations has raised
MTA	Event & Alarm	Group has exceeded the number threshold
MTA	Event & Alarm	Group has exceeded the volume threshold
MTA	Event & Alarm	Group has exceeded the age threshold

Tabelle 7.1: Events und Alarme

7.3 Möglichkeiten zur Weiterentwicklung des Prototypen

Der beschriebene Prototyp implementiert nur einen kleinen Ausschnitt der in Abschnitt 3.4 angedachten Funktionalität. Die Implementierung bewegt sich ausschließlich auf Application-Entity-Ebene, wo hauptsächlich Belange des Leistungsmanagements sowie rudimentär des Fehler- und Sicherheitsmanagements realisiert wurden.

Die nächsten Schritte in der Implementierung des in Kapitel 3 vorgestellten Konzepts sind

- der Ausbau der Application-Entity-Ebene, wobei insbesondere die Konfigurations- und Sicherheitsproblematik berücksichtigt werden muß.

- die Integration der Application-Ebene. Zunächst gilt es, die Organisationsstruktur und die Topologie des MHS in der Managementplattform zu modellieren.

Im folgenden soll kurz aufgezeigt werden, wie diese Ziele erreicht werden können und wo Probleme auftreten.

7.3.1 Verfügbare Managementinformation

Das größte Problem ist die Beschaffung der Managementinformation, die für die Weiterentwicklung benötigt würde. Die Daten, die Network Services Monitoring MIB und Mail Monitoring MIB liefern, reichen für ein komplexeres Managementmodul nicht aus, es müssen also zusätzliche Quellen gefunden werden.

- XT-PP bietet in der aktuellen Version die Möglichkeit, Teile seiner Konfigurationsdaten aus einem X.500-Directory auszulesen. Um Spectrum den Zugriff auf diese Daten zu ermöglichen, ist es notwendig, den DCM (siehe Abschnitt 6.2.2) um ein Modul zu erweitern, das die Requests der VNM in Directoryabfragen übersetzt, bzw. ein solches Modul über das EPI anzubinden.
- Der SNMP-Agent von XT-PP müßte erweitert werden, so daß er mehr und detailliertere Daten anbietet.
- XT-PP bietet eine weitere Managementschnittstelle an, die von MTA-console und lconsole verwendet wird. An diesem Interface werden zum Teil detailliertere Daten als an der SNMP-Schnittstelle angeboten. Um Zugriff auf diese Daten zu erhalten, müßte ebenfalls eine Erweiterung des DCM vorgenommen werden.
- Als letzte Möglichkeit könnten auch mit modifizierten Systemmanagementtools auf dem Agentensystem die Log- und Konfigurationsdateien der Komponente ausgelesen und in letzterem Fall modifiziert werden.

All diese Vorgehensweisen bringen allerdings den großen Nachteil mit sich, daß hier auf proprietäre Methoden bzw. Datenstrukturen zurückgegriffen werden müßte, was der Managementlösung die breite Anwendungsbasis entziehen würde.

7.3.2 Modellierung

Durch sein Konzept von Klassen und Relationen stellt Spectrum sehr mächtige Modellierungsmöglichkeiten bereit. Einerseits können die verschiedenen in Abschnitt 3.4 vorgestellten Klassen unmittelbar auf Model Types, andererseits die Verweise zwischen Objekten auf Relationen abgebildet werden.

Zum Teil stellt Spectrum schon verwendbare Strukturen bereit.

- Beispielsweise existiert eine auf den organizes- und org_owns-Relations basierende Hierarchie, die Klassen wie Enterprise, Division, Subsidiary und Department miteinander verbindet und so den Aufbau eines Unternehmens beschreibt. Es wäre zu untersuchen, ob die Adreßraumorganisation des MHS auf diese Struktur abgebildet werden kann.

- Um die Topologie des MHS zu modellieren, könnte die Relation connects_to verwendet werden, die ansonsten die Topologie des Netzes beschreibt.

Die Funktionalität der verschiedenen Klassen und Objekte des Modells kann durch Inference Handler implementiert werden, wobei allerdings zu beachten ist, daß ein erheblicher Entwicklungsaufwand in Kauf genommen werden muß. Daher muß untersucht werden, inwieweit die in der Plattform bereits verfügbare Funktionalität an diese Aufgabenstellung angepaßt und wiederverwendet werden kann.

Die generischen Methoden von Spectrum, die in Abschnitt 6.2.1.3 beschrieben sind, sind im wesentlichen auf eine statische Konfiguration ausgelegt. So wird die Hardwarekonfiguration eines Geräts ausschließlich bei der Kreation des zugehörigen Objekts ausgewertet. Auch die Instantiierung der Major und Minor Applications geschieht zu diesem Zeitpunkt und wird ausschließlich auf explizite Anforderung der Anwenders erneut durchgeführt.

Daher sollte in diesem Zusammenhang MaestroVision, eine Spectrum-Erweiterung für Systemmanagement, näher betrachtet werden.

MaestroVision verwaltet auch dynamische Daten, etwa die Prozeßtabelle eines Unixsystems, wobei hier auch eine Überwachung von kurzlebigen Objekten und eine dynamische Zuordnung von Objekten zu Ressourcen unterstützt wird. Ob die Funktionalität von MaestroVision so generisch gehalten ist, daß sie auch für andere Zwecke Verwendung finden kann, wäre noch zu untersuchen.

Für eine detailliertere Beschreibung von MaestroVision sei auf [Dem95] verwiesen.

7.3.3 Darstellung

Ein großes Problem ist die graphische Darstellung neuer Hierarchien mit dem SpectroGRAPH, da Spectrum in seiner Bedienoberfläche noch nicht genügend parametrisierbare Funktionen für die Visualisierung komplexerer Zusammenhänge bereitstellt.

Beispielsweise stellt der SpectroGRAPH bereits eine Viewhierarchie für Darstellung der organisatorischen und topologischen Struktur des Netzes zur Verfügung. Die Gliederung des Netzes wird auf eine Viewhierarchie abgebildet, wobei Subnetze im übergeordneten View je durch ein Icon dargestellt werden, über das der entsprechende Subnetz-View erreichbar ist. Die Verkabelungsstruktur des Netzes wird durch Verbindunglinien zwischen den einzelnen Geräte- bzw. Subnetzicons dargestellt.

Diese Viewhierarchie würde sich in dieser Form auch sehr gut für die Visualisierung der organisatorischen und topologischen Struktur des MHS eignen, allerdings ist es nicht möglich, diese Viewstruktur für diese Aufgabe mit geänderten Bezugsrelationen wiederzuverwenden.

Kapitel 8

Ausblick

Im allgemeinen werden zur Zeit bei der Erstellung einer Managementlösung für eine verteilte Anwendung die folgenden Schritten durchlaufen:

1. Der Bedarf für ein Managementkonzept für eine bestehende verteilte Anwendung wird erkannt.
2. Die Anwendung wird auf relevante Managementinformation hin analysiert.
3. Ein Objektmodell wird entwickelt, das alle managementrelevanten Aspekte der Anwendung beschreibt.
4. Die Anwendung wird um einen (für die gesamte Anwendung) oder mehrere (etwa für jeden Baustein) Managementagenten erweitert, die diese Managed Objects bereitstellen.
5. Eine Managementanwendung wird erstellt.

Bei der Entwicklung der Applikation wird also die Managementproblematik zumeist nicht berücksichtigt. Vielmehr behandeln die Managementkonzepte für verteilte Anwendungen im allgemeinen ausschließlich den laufenden Betrieb, nicht aber die Planungs-, Spezifikations-, Entwicklungs- und Installationsphase.

Die zunehmende Komplexität der neu entwickelten Applikationen wird ein nachträgliches „Aufpfropfen“ eines Managementkonzepts zunehmend schwieriger gestalten. Langfristiges Ziel muß daher sein, bestehende Methoden des Softwareengineerings und des Anwendungsmanagements zu verschmelzen, um zu einer umfassenden Lösung zu gelangen.

Ein Modell, das diesen holistischen Ansatz verfolgt, ist Open Distributed Processing (ODP), ein Rahmenwerk für die Entwicklung von verteilten Anwendungen, das die ISO und andere Organisationen zur Zeit entwerfen.

ODP gliedert eine Applikation in fünf Viewpoints (vgl. Abbildung 8.1), von denen Enterprise und Technology Viewpoint eine Verbindung zu Enterprise- bzw. Netz- und Systemmanagement bilden, während Information, Computation und Engineering Viewpoint den Entwicklungsprozeß der Anwendung beschreiben.

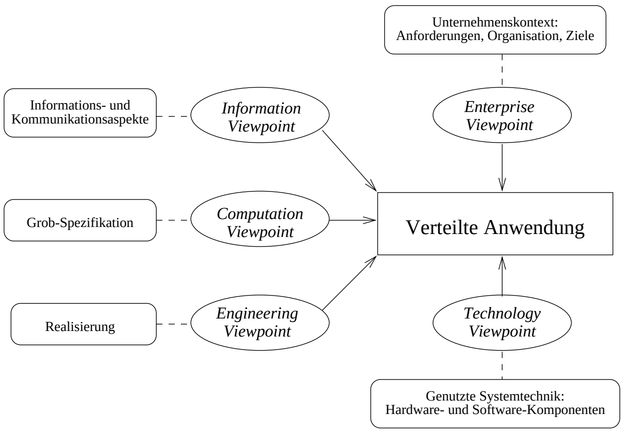


Abbildung 8.1: Viewpoints von ODP (aus [HA93])

ODP ist in seiner Entwicklung noch nicht so weit fortgeschritten, um bereits fertige Lösungen anbieten zu können (Ansätze finden sich in [SZ92]), das Modell dürfte in Zukunft eine (zumindestens konzeptionell) nicht unerhebliche Rolle im Problemkreis Anwendungsmanagement spielen.

Während sich ODP sehr stark auf konzeptioneller Ebene bewegt, ist die Object Management Architecture (OMA) der Object Management Group (OMG), die im folgenden beschrieben wird, im technischen Bereich anzusiedeln.

Abbildung 8.2 gibt einen Überblick über OMA. Eine Applikation besteht aus einer Menge zusammenarbeitenender Objekte, die untereinander ausschließlich mit Hilfe des Object Request Broker (ORB) kommunizieren. Dadurch ist es möglich, die Objekte transparent auf verschiedene Rechensysteme zu verteilen. OMA unterscheidet zwischen Object Services, die Basisfunktionen für die Verwaltung der Objekte bereitstellen, Common Facilities, die Basisdienste anbieten, und Application Objects, die die spezifische Funktionalität der jeweiligen Anwendung realisieren.

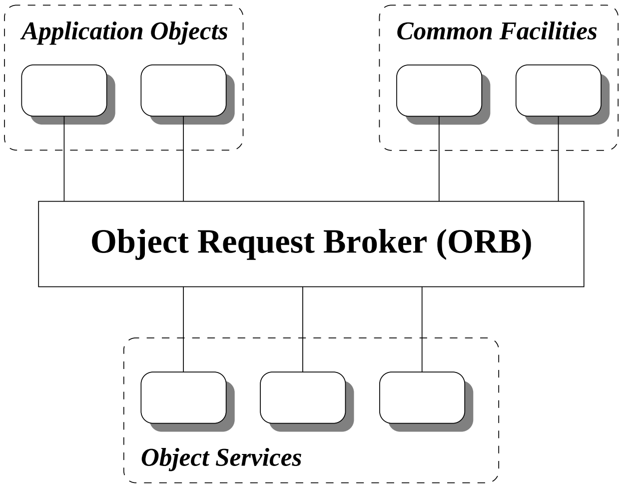


Abbildung 8.2: Überblick über OMA (aus [OMG92])

Die Objekte, aus denen sich die Anwendung zusammensetzt, werden auch innerhalb der Managementarchitektur als Managed Objects betrachtet. Ansätze, das OMA- und das OSI-Informationsmodell zu verschmelzen, werden in [QP91] beschrieben.

Auf die bisher bestehende Trennung zwischen Softwarearchitektur und Managementmodell kann dann verzichtet werden. Desweiteren ist eine dedizierte Agentenkomponente innerhalb der Anwendung nicht mehr notwendig, vielmehr kommuniziert die Managementanwendung unmittelbar mit den Softwareobjekten, aus denen sich die Anwendung zusammensetzt. So wird eine vollständige Integration von Funktions- und Managementaspekten einer verteilten Anwendung möglich.

Anhang A

Konfiguration des Proxy-Agenten

Während SNMPv1 als einziges Kriterium für den lesenden bzw. schreibenden Zugriff auf eine Managementinformation die Kenntnis eines Passworts, des Community Strings, voraussetzt, existieren bei SNMPv2 eine größere Menge von Regulativen für diese Aufgabe:

- die Partei, die den Request gestellt hat
- die Partei, an die der Request gerichtet wurde
- der Context, der dem Request zugeordnet ist
- die Art des Requests (Lesen, Schreiben, Antworten, Benachrichtigen)
- die Information, auf die sich der Request bezieht

Im folgenden wird die Konfiguration des Proxyagenten im Managementszenario, wie es in Abschnitt 6.1.4 dargestellt ist, beschrieben.

In Tabelle A.1 sind die Parteien aufgeführt, die an Managementabläufen teilnehmen. Dem Manager und dem Proxyagenten sind je drei Parteien zugeordnet, wobei unterschiedliche Authentifizierungs- und Verschlüsselungsalgorithmen zum Einsatz kommen. Den Subagenten ist je eine Party zugeordnet, hier findet die Kommunikation mit dem Hauptagenten mit den SNMPv1-Protokollen statt (vergleiche auch Abbildung 6.2 in Abschnitt 6.1.4).

Auf den Variablen, die der Agent selbst anbietet, sind zwei Views definiert, von denen einer den Zugriff auf vier Teilbereiche einschränkt, während der andere vollen Zugriff bietet (vgl. Tabelle A.2).

Für diese beiden Views ist jeweils ein Kontext definiert, außerdem existieren Kontexte für den Zugriff auf die drei Subagenten (siehe Tabelle A.3).

Nr	Party	Domain	Auth.	Priv.	Host	Port
1	Proxy Agent	snmpUDP	keine	keine	local	161
2	Manager	snmpUDP	keine	keine	*	*
3	Proxy Agent	snmpUDP	MD5	keine	local	161
4	Manager	snmpUDP	MD5	keine	*	*
5	Proxy Agent	snmpUDP	MD5	DES	local	161
6	Manager	snmpUDP	MD5	DES	*	*
7	Unix Agent	rfc1157	rfc1157	keine	local	25761
8	qmgr Agent	rfc1157	rfc1157	keine	local	26017
9	DSA Agent	rfc1157	rfc1157	keine	local	26273

Tabelle A.1: Parties im Proxy-Setup

Nr	Teilbäume
1	system
	snmpStats
	snmpParties
	snmpTraps
2	internet

Tabelle A.2: MIB-Views im Proxy-Setup

Nr	Context	Zugriff
1	system	Lokaler Zugriff mit View 1
2	internet	Lokaler Zugriff mit View 2
3	unix	Proxy-Zugriff auf Unix Agent
4	qmgr	Proxy-Zugriff auf qmgr Agent
5	quipu	Proxy-Zugriff auf DSA Agent

Tabelle A.3: Contexte im Proxy-Setup

Auf Basis dieser Definitionen werden in Tabelle A.4 die Zugriffsrechte festgelegt. Jeder Manager kann auf die in View 1 definierten Managementinformationen sowie auf sämtliche Subagenten lesend zugreifen. Um auf die Variablen schreiben zu können, muß er sich gegenüber dem Agenten zumindestens authentifizieren. Selbiges gilt für den Lese- und Schreibzugriff auf die Variablen des Proxyagenten, die nicht in View 1 enthalten sind.

Dst	Src	Context	Rechte
1	1	1	notify
1	2	1, 3, 4, 5	get get-next get-bulk
2	1	1, 3, 4, 5	response
3	4	2, 3, 4, 5	get get-next set get-bulk
4	3	2, 3, 4, 5	response
5	6	2	get get-next set get-bulk
6	5	2	response

Tabelle A.4: Zugriffsrechte im Proxy-Setup

Anhang B

Beispiel für einen Inference Handler

Der hier implementierte Inference Handler wird durch Änderungen des Attributs applOperStatus (Application) getriggert. Abhängig vom neuen Attributwert wird ein congested- oder down-Alarm gesetzt bzw. gelöscht.

Makefile

```
######################################################################
##
##  Makefile for Inference Handler Demo
##  Author:  Hans Maurer
##
##  based on Makefile by
##
##    Cabletron Systems Incorporated
##    Post Office Box 5005
##    Rochester, NH 03867-5005
##    Copyright (c) 1991, All Rights Reserved
##
######################################################################


TOP = /Development/Spectrum3.0.1/IHAPI
SS_DIR = $(TOP)/../SS
MAKEDEPEND = makedepend

include $(TOP)/make.defs

depend_here :
	$(MAKEDEPEND) -- $(INCLUDES) -- *.cc

CFLAGS = -DCS_DEBUG
```

```
INCLUDES = \
        -I$(IHAPI_INC) \
        -I$(GLOBL_INC) \
        -I$(VPAPI_INC) \
        $(ENDOFLIST)

LIBS = \
        $(MALLOCO) \
        $(ENDOFLIST)

.PRECIOUS: IcsX_Appl.o

OBJS = \
        IcsX_IH_Application_Status.o \
        IcsX_MI_Application.o \
        $(ENDOFLIST)

init_here :

link_here : ApplSS

ApplSS : IcsX_Appl.o
        SS_DIR="$(SS_DIR)" LN="$(LN)" /bin/sh $(TOP)/demo/DemoLink $@ \
            IcsX_Appl.o $(LIBS)

local : IcsX_Appl.o

IcsX_Appl.o : $(OBJS)
        $(LD) -r $(LDFLAGS) $(OBJS) -o $@

# DO NOT DELETE THIS LINE -- make depend depends on it.
```

IcsX_MT_Application.h

```
#ifndef __ICSX_MT_APPLICATION__H__
#define __ICSX_MT_APPLICATION__H__

class IcsX_MT_Application
{
    public:

        enum
        {
            MT_HANDLE           =  0x013c0005
        };
        enum
        {
            AI_CONDITION        =  0x0001000a,
            AI_APPLOPERSTATUS   =  0x013c0023
        };
        enum
        {
            PC_STATUS_DOWN      =  0x013c0001,
            PC_STATUS_CONGESTED =  0x013c0002
        };
};

#endif
```

IcsX_IH_Application_Status.h

```
#ifndef __ICSX_IH_APPLICATION_STATUS_H__
#define __ICSX_IH_APPLICATION_STATUS_H__

#ifndef __CSIHASCOND_H__
#include "CsIHAsCond.h"
#endif

#ifndef __CSCHANGEN_H__
#include "CsChangeN.h"
#endif

#ifndef __ICSX_MT_APPLICATION_H__
#include "IcsX_MT_Application.h"
#endif

class IcsX_IH_Application_Status : public CsIHAsCond
{
    public:

        IcsX_IH_Application_Status( CsMTypeHandle& mth);

        void trig_attr_change( const CsModelHandle & , const CsChangeNode * );

};

#endif
```

IcsX_MI_Application.cc

```
#ifndef  __CSMINODE_H__
#include "CsMINode.h"
#endif

#ifndef __ICSX_MT_APPLICATION_H__
#include "IcsX_MT_Application.h"
#endif

#ifndef __ICSX_IH_APPLICATION_STATUS_H__
#include "IcsX_IH_Application_Status.h"
#endif

static void
IcsX_MI_Application_Status( CsMTypeHandle& m_t_handle )
{
    IcsX_IH_Application_Status *ih_appl_status =
        new IcsX_IH_Application_Status( m_t_handle );

    if ( ih_appl_status->get_error() == CsError::FAILURE )
    {
        CsDelete( ih_appl_status );
    }
}

static CsMINode mi_appl_status ( IcsX_MT_Application::MT_HANDLE,
                                 IcsX_MI_Application_Status );
```

IcsX_IH_Application_Status.cc

```
#ifndef __ICSX_IH_APPLICATION_STATUS_H__
#include "IcsX_IH_Application_Status.h"
#endif

#ifndef __CSERROR_H__
#include "CsError.h"
#endif

#ifndef __CSATTRVALDEF_H__
#include "CsAttrValDef.h"
#endif

#ifndef __CSCHANGEN_H__
#include "CsChangeN.h"
#endif

#ifndef __CSDEBUG_H__
#include "CsDebug.h"
#endif

IcsX_IH_Application_Status::IcsX_IH_Application_Status(
        CsMTypeHandle& in_m_t_handle ) : CsIHAsCond( in_m_t_handle )
{
    tout( "IcsX_IH_Application_Status::IcsX_IH_Application_Status() m_t_handle=("
        << hex << in_m_t_handle << dec << ") starting." << endl );

    CsVnmMTypeHandle vmth = in_m_t_handle;

    // Verify that the model type contains the attribute Condition
    if ( ! vmth.has_attr ( IcsX_MT_Application::AI_CONDITION ))
    {
        wout( "No such attribute: Condition." << endl );
        set_error( CsError::FAILURE );
        return;
    }

    // Verify that the model type contains the attribute applOperStatus
    if ( ! vmth.has_attr ( IcsX_MT_Application::AI_APPLOPERSTATUS ))
    {
        wout( "No such attribute: applOperStatus." << endl );
        set_error( CsError::FAILURE );
        return;
    }

    // Register trigger on changes of applOperStatus
    if ( !reg_attr_change ( IcsX_MT_Application::AI_APPLOPERSTATUS ) )
    {
        wout( "IcsX_IH_Application_Status() Couldn't register for changes in "
              "OperStatus." << endl );
        set_error( CsError::FAILURE );
        return;
    }
```

```
    {
        iout( "IcsX_IH_Application_Status() Registered for OperStatus Attr "
              "Change !" << endl);
    }

    tout ("IcsX_IH_Application_Status::IcsX_IH_Application_Status() ending."
          << endl);
}

void
IcsX_IH_Application_Status::trig_attr_change(
        const CsModelHandle& mh, const CsChangeNode * change )
{
    tout( "IcsX_IH_Application_Status::trig_attr_change() mh=("
          << hex << mh << dec << ") starting." << endl );

    // Changed attribute == applOperStatus?
    if ( change->get_attr_id() == IcsX_MT_Application::AI_APPLOPERSTATUS )
    {
        CsTulong operstatus = * (CsTulong *)(change->get_cur_value());

        iout( "IcsX_MT_Application::trig_attr_change() mh=(" << hex << mh
               << dec << ") OperStatus = " << operstatus << endl);

        // Create/delete alarms
        switch(operstatus)
        {
            case 1:    // up
                disassert_c( mh );
                break;
            case 2:    // down
            case 3:    // halted
            case 5:    // restarting
                assert_c( mh, RED_CONDITION,
                          IcsX_MT_Application::PC_STATUS_DOWN );
                break;
            case 4:    // congested
                assert_c( mh, ORANGE_CONDITION,
                              IcsX_MT_Application::PC_STATUS_CONGESTED );
                break;
            default:   // unknown status
                eout( "IcsX_MT_Application::trig_attr_change() mh=(" << hex << mh
                      << dec << ") Unknown OperStatus = " << operstatus << endl);
                break;
        }
    }
    else
    {
        eout( "Model with mh=(" << hex << mh << dec << ") is only registered "
              "for OperStatus." << endl );
    }
}
```

Abkürzungsverzeichnis

ACSE	Association Control Service Element
ADMD	Administration Management Domain
AG	Aktiengesellschaft
API	Application Programming Interface
ASE	Application Service Element
ASN.1	Abstract Syntax Notation One
AU	Access Unit
BASF	Badische Anilin- und Sodafabriken
CLNS	Connectionless Network Service
CMIP	Common Management Information Protocol
CONS	Connection Oriented Network Service
DCM	Device Communication Manager
DES	Data Encryption Standard
DFN	Deutsches Forschungsnetz
DIB	Directory Information Base
DIT	Directory Information Tree
DL	Distribution List
DN	Distinguished Name
DS	Directory System
DSA	Directory System Agent
EDI	Electronic Data Interchange
EPI	External Protocol Interface
FTAM	File Transfer, Access and Management
GIB	Generic Information Block

GMD	Gesellschaft für Mathematik und Datenverarbeitung
IAB	Internet Activities Board
ICMP	Internet Control Message Protocol
IHAPI	Inference Handler Application Programming Interface
IIB	Icon Information Block
IMT	Inductive Modeling Technoloy
IP	Internet Protocol
IPM	Interpersonal Message
ISO	International Standardization Organization
ISODE	ISO Development Environment
ITU	Internation Telecommunication Union
MASE	Message Administration Service Element
MD	Management Domain
MDSE	Message Delivery Service Element
MHE	Message Handling Environment
MHS	Message Handling System
MIB	Management Information Base
MIME	Multipurpose Internet Mail Extension
MOOT	Modular Object Oriented Threads
MRSE	Message Retrieval Service Element
MS	Message Store
MS	Microsoft
MSAP	Management Station Access Provider
MSSE	Message Submission Service Element
MTA	Message Transfer Agent
MTE	Model Type Editor
MTL	Message Transfer Layer
MTS	Message Transfer Service
MTS	Message Transfer System
MTSE	Message Transfer Service Element
MX	Mail Exchange
O/R	Originator/Recipient
ODP	Open Distributed Processing
OID	Object Identifier
OMA	Object Management Architecture
OMG	Object Management Group
ORB	Object Request Broker

OSI	Open Systems Interconnection
OU	Organizational Unit
PC	Personal Computer
PDAU	Physical Delivery Access Unit
PRMD	Private Management Domain
QoS	Quality of Service
RFC	Request for Comments
ROSE	Remote Operations Service Element
RPC	Remote Procedure Call
RTSE	Reliable Transfer Service Element
SMTP	Simple Mail Transport Protocol
SNMP	Simple Network Management Protocol
SSAPI	SpectroSERVER Application Programming Interface
TCP	Transmission Control Protocol
TMN	Telecommunications Management Network
UA	User Agent
UAL	User Agent Layer
UUCP	Unix to Unix Copy
VNM	Virtual Network Machine
VMS	Virtual Memory System
VTP	Virtual Terminal Protocol
WIN	Wissenschaftsnetz

Literaturverzeichnis

[Abe94a] Sebastian Abeck, “Integriertes Anwendungsmanagement am Beispiel einer unternehmensspezifischen, kommunikationsnahen Anwendung”, *Praxis der Informationsverarbeitung und Kommunikation*, (4), Oktober 1994.

[Abe94b] Sebastian Abeck, “Integriertes Management von Informationsverarbeitungs-Ressourcen am Beispiel eines großen Netzbetreibers”, Foliensatz für einen Vortrag an der Universität Koblenz-Landau, Juni 1994.

[Abe95] Sebastian Abeck, “Integrationstechniken im Netzmanagement”, In *Kommunikation in verteilten Systemen*, GI/ITG–Fachtagung, Februar 1995.

[Alv95a] Harald T. Alvestrand, “Frequently Asked Questions on comp.protocols.iso.X400”, Monatliche Veröffentlichung in `news:comp.protocols.iso.x400`, Januar 1995, archiviert auf `ftp://aun.uninett.no/pub/mail/x400faq/FAQ-mhsnews.text`.

[Alv95b] Harald T. Alvestrand, “FAQ/Product list — Products implementing the X.400 standards”, Monatliche Veröffentlichung in `news:comp.protocols.iso.x400`, April 1995, archiviert auf `ftp://aun.uninett.no/pub/mail/x400faq/FAQ-products.text`.

[Bal89] Alexander Balling, “Entwicklung eines in das OSI-Management integrierten Fehlerbehandlungskonzepts für die Anwendung MHS”, Diplomarbeit, Technische Universität München, Institut für Informatik, Mai 1989.

[Dem95] Stefan Deml, “Modellierung der verteilten Anwendung SAP R/3 für ein integriertes Management”, Diplomarbeit, Technische Universität München, Institut für Informatik, Mai 1995.

[DHM93] James B. Duff, James D. Hunter und David C. Matthews, "Process Management: The Future of Integrated Management Systems", In Heinz-Gerd Hegering und Yechiam Yemini, Herausgeber, *Integrated Network Management III*, Seiten 451–462, International Federation of Information Processing, North-Holland, 1993.

[DM94] Stefan Deml und Johann Maurer, "Integration des IBM 6611 Network Processors in SPECTRUM", Fortgeschrittenenpraktikum, Technische Universität München, Institut für Informatik, Mai 1994.

[Güc94] Angelika Güc, "Auf dem Vormarsch — X.400: Weltweiter Standard für elektronischen Datenaustausch", *Gateway*, Seiten 34–38, April 1994.

[HA93] Heinz-Gerd Hegering und Sebastian Abeck, *Integriertes Netz- und Systemmanagement*, Addison-Wesley, Bonn, 1993.

[Has94] Harald Hassenmüller, "Türöffner — Gateways für EMail-Systeme", *Gateway*, Seiten 24–28, April 1994.

[HBB93] James W. Hong, Michael A. Bauer und J. Michael Bennett, "Integration of the Directory Service in the Network Management Framework", In Heinz-Gerd Hegering und Yechiam Yemini, Herausgeber, *Integrated Network Management III*, Seiten 149–160, International Federation of Information Processing, North-Holland, 1993.

[HNG94] Heinz-Gerd Hegering, Bernhard Neumair und Markus Gutschmidt, "Cooperative Computing and Integrated System Management — A Critical Comparision of Architectural Approaches", Bericht 9403, Ludwig-Maximilians-Universität München, Institut für Informatik, Februar 1994.

[HNG95] Heinz-Gerd Hegering, Bernhard Neumair und Markus Gutschmidt, "Cooperative Computing und integriertes Systemmanagement", *Informatik-Spektrum*, 1995.

[HNW95] Heinz-Gerd Hegering, Bernhard Neumair und Rene Wies, "Integriertes Management verteilter Systeme — Ein Überblick über den State-of-the Art", Bericht 9503, Ludwig-Maximilians-Universität München, Institut für Informatik, Januar 1995.

[HPBS94] Jürgen Harms, Jean-Francois Paccini, Vito Baggiolini und Eduardo Solana, "Management of Distributed Applications: Management of Electronic-Mail Systems", Folien zu einem Vortrag auf dem HP Labs Forum, Juni 1994.

[ISO 7498] "Information Processing Systems — Open Systems Interconnection — Basic Reference Model, Part 1–4", IS 7498-X, International Standardization Organization, 1989–1994.

[ISO 7498-4] "Information Processing Systems — Open Systems Interconnection — Basic Reference Model Part 4: Management Framework", IS 7498-4, International Standardization Organization, 1989.

[ISO 9596] "Information technology — Open Systems Interconnection — Common management information protocol, Part 1–2", IS 9596-X, International Standardization Organization, 1991–1993.

[ISO10040] "Information technology — Open Systems Interconnection — Systems management overview", IS 10040, International Standardization Organization, 1992.

[ISO10164] "Information technology — Open Systems Interconnection — Systems Management (Functions), Part 1–18", IS/DIS 10164-X, International Standardization Organization, 1992–1994.

[ISO10165] "Information technology — Open Systems Interconnection — Management Information Services — Structure of management information, Part 1–7", IS/DIS 10165-X, International Standardization Organization, 1992–1994.

[ISO10746] "Information technology — Open Systems Interconnection — Data Management and Open Distributed Processing — Basic reference model of open distributed processing, Part 2–3", DIS 10746-X, International Standardization Organization.

[Jab95] S. Jablonski, "Workflow-Management-Systeme: Motivation, Modellierung, Architektur", *Informatik-Spektrum*, 18(1):13–24, Februar 1995.

[Kel93] Alexander Keller, "Entwurf und Implementierung von Managementszenarien zu bestehenden Kommunikationsanwendungen", Diplomarbeit, Technische Universität München, Institut für Informatik, Mai 1993.

[Kil91] Stephen E. Kille, *Implementing X.400 and X.500: The PP and QUIPU Systems*, Artech House, Inc., 685 Canton Street, Norwood, MA 02062, 1991.

[Kri93] Iyengar Krishnan, "X.400 Message Handling Systems Management Functional Requirements", In Heinz-Gerd Hegering und Yechiam Yemini, Herausgeber, *Integrated Network Management III*, Seiten 435–449, International Federation of Information Processing, North-Holland, 1993.

[Kru93] B. Krupczak, "UNIX Systems Management via SNMP", In Heinz-Gerd Hegering und Yechiam Yemini, Herausgeber, *Integrated Network Management III*, Seiten 289–299, International Federation of Information Processing, North-Holland, 1993.

[LLO93] Jerry N. Luftman, Paul R. Lewis und Scott H. Oldach, "Transforming the Enterprise: The Alignment of Business and Information Technology Strategie", *IBM Systems Journal*, 32(1):198–221, Januar 1993.

[Loh92] Jürgen Lohrmann, "Management der NWP-Knoten auf SNMP-Basis", Fachkonzept, ICS GmbH, August 1992.

[MTW93] Ursula Meyer zu Natrup, Michael Tschichholz und Susanne Waßerroth, "A Service for Administering Management Information — VPN Inter-Domain Management Support using X.500 and X.700", In Heinz-Gerd Hegering und Yechiam Yemini, Herausgeber, *Integrated Network Management III*, Seiten 137–148, International Federation of Information Processing, North-Holland, 1993.

[MVRN] Calypso Software Systems, Inc., 134 South Mast Road, Goffstown, NH 03045, *MaestroVision 2.0 beta 1 Release Notes*, Oktober 1993.

[MVTech] Calypso Software Systems, Inc., 134 South Mast Road, Goffstown, NH 03045, *MaestroVision 2.0 Technical Summary*, August 1993.

[MVUser] Cabletron Systems, Inc., Rochester, NH 03867-5005, *MaestroVision User's Guide*, März 1995, Order Number: 9031224E2.

[NexDUa] NEXOR Limited, *XT-DUA Administration Guide*, Version 1.1.

[NexDUb] NEXOR Limited, *XT-DUA User Guide*, Version 1.1.

[NexMUA] NEXOR Limited, *XT-MUA User Guide*, Version 1.1.

[NexPPa] NEXOR Limited, *XT-PP Reference Manual*, Version 7.1.

[NexPPb] NEXOR Limited, *XT-PP Tutorial*, 1993.

[NexQUa] NEXOR Limited, *XT-QUIPU Reference Manual*, Version 7.1.

[NexQUb] NEXOR Limited, *XT-QUIPU Tutorial*, 1993.

[OMG92] Object Management Group, Inc., *Object Management Architecture Guide*, September 1992.

[PLL+93] B. Plattner, G. Lanz, H. Lubich, M. Müller und T. Walter, *X.400, elektronische Post und Datenkommunikation*, Addison-Wesley, Bonn, 3. erweiterte Auflage, 1993.

[PM94] Dinesh Pai und Dave Milham, "An Approach to Business-driven Definition of Managed Objects", kommentierter Foliensatz, Februar 1994.

[QP91] Peggy Quinn und George Preoteasa, "Reconceiling Object Models for Systems and Networks Management", Technischer Bericht, UNIX Systems Laboratories, Inc., 1991.

[RFC 821] Jonathan B. Postel, "Simple Mail Transfer Protocol", RFC 821, Internet Activities Board, August 1982.

[RFC 822] David H. Crocker, "Standard for the Format of ARPA Internet Text Messages", RFC 822, Internet Activities Board, August 1982.

[RFC 934] Marshall T. Rose und Einar A. Stefferud, "Proposed Standard for Message Encapsulation", RFC 934, Internet Activities Board, Januar 1985.

[RFC 974] Craig Partridge, "Mail Routing and the Domain System", RFC 974, Internet Activities Board, Januar 1986.

[RFC1006] Marshall T. Rose und Dwight E. Cass, "ISO Transport Services on top of the TCP Version: 3", RFC 1006, Internet Activities Board, Mai 1987.

[RFC1327] Steve Hardcastle-Kille, "Mapping between X.400(1988) / ISO 10021 and RFC 822", RFC 1327, Internet Activities Board, Mai 1992.

[RFC1441] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose und Steven Waldbusser, "Introduction to version 2 of the Internet-standard Network Management Framework", RFC 1441, Internet Activities Board, April 1993.

[RFC1442] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose und Steven Waldbusser, "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1442, Internet Activities Board, April 1993.

[RFC1443] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose und Steven Waldbusser, "Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1443, Internet Activities Board, April 1993.

[RFC1444] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose und Steven Waldbusser, "Conformance Statements for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1444, Internet Activities Board, April 1993.

[RFC1445] James M. Galvin und Keith McCloghrie, "Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1445, Internet Activities Board, April 1993.

[RFC1446] James M. Galvin und Keith McCloghrie, "Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1446, Internet Activities Board, April 1993.

[RFC1447] Keith McCloghrie und James M. Galvin, "Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1447, Internet Activities Board, April 1993.

[RFC1448] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose und Steven Waldbusser, "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1448, Internet Activities Board, April 1993.

[RFC1449] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose und Steven Waldbusser, "Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1449, Internet Activities Board, April 1993.

[RFC1450] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose und Steven Waldbusser, "Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1450, Internet Activities Board, April 1993.

[RFC1451] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose und Steven Waldbusser, "Manager-to-Manager Management Information Base", RFC 1451, Internet Activities Board, April 1993.

[RFC1452] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose und Steven Waldbusser, "Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework", RFC 1452, Internet Activities Board, April 1993.

[RFC1514] Pete Grillo und Steven Waldbusser, "Host Resources MIB", RFC 1514, Internet Activities Board, September 1993.

[RFC1521] Nathaniel S. Borenstein und Ned Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, Internet Activities Board, September 1993.

[RFC1522] Keith Moore, "MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text", RFC 1522, Internet Activities Board, September 1993.

[RFC1565] Steve Kille und Ned Freed, "Network Services Monitoring MIB", RFC 1565, Internet Activities Board, Januar 1994.

[RFC1566] Steve Kille und Ned Freed, "Mail Monitoring MIB", RFC 1566, Internet Activities Board, Januar 1994.

[RFC1567] Glenn Mansfield und Steve E. Kille, "X.500 Directory Monitoring MIB", RFC 1567, Internet Activities Board, Januar 1994.

[RFC1697] David Brower, Bob Purvy, Anthony Daniel, Marc Sinykin und Jay Smith, "Relational Database Management System (RDBMS) Management Information Base (MIB) using SMIv2", RFC 1697, Internet Activities Board, August 1994.

[Ros91] Marshall T. Rose, *The Simple Book: An Introduction to Management of TCP/IP-based internets*, Prentice Hall, 1991.

[Sch89] Stefanie Schnappinger, “Analyse der MAIL.2000-Fehlerbehandlung unter dem Aspekt der Abbildung auf OSI-Fehlermanagementkonzepte”, Diplomarbeit, Technische Universität München, Institut für Informatik, November 1989.

[SpecADG] Cabletron Systems, Inc., Rochester, NH 03867-5005, *System Administrator's Guide*, August 1994, Order Number: 9030264-01 E14.

[SpecAPIa] Cabletron Systems, Inc., Rochester, NH 03867-5005, *SpectroSERVER API Developer's Guide*, April 1994, Order Number: 9030624 E9.

[SpecAPIb] Cabletron Systems, Inc., Rochester, NH 03867-5005, *SpectroSERVER API Reference*, März 1994, Order Number: 9030486 E6.

[SpecApp] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum How to Integrate Applications with SPECTRUM*, August 1994, Order Number: 9031112 E2.

[SpecBas] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Basic Extensions Guide*, März 1992, Order Number: 9030453 E1.

[SpecCLI] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Command Line Interface User's Guide*, August 1994, Order Number: 9030664 E5.

[SpecCon] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Concepts Guide*, August 1994, Order Number: 9030647 E4.

[SpecEPI] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum MSAP/EPI Developers Guide*, Juni 1993, Order Number: 9030487 E4.

[SpecExt] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Extensions Integration Toolkit Developer's Guide*, Juli 1994, Order Number: 9030623 E8.

[SpecGCR] Cabletron Systems, Inc., Rochester, NH 03867-0505, *Spectrum Global Classes Reference*, März 1994, Order Number: 9030665 E6.

[SpecGen] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Management Module Guide for Generic SNMP*, September 1994, Order Number: 9030856 E3.

[SpecGIB] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Gib Editor Guide*, August 1994, Order Number: 9030660 E7.

[SpecIIB] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Icon Information Block (IIB) Editor Guide*, Juli 1994, Order Number: 9030724 E5.

[SpecInf] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Inference Handler Developer's Guide*, Dezember 1992, Order Number: 9030489 E5.

[SpecKBG] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Knowledge-Base Guide*, August 1994, Order Number: 9030663 E1.

[SpecMTE] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Model Type Editor Guide*, August 1994, Order Number: 9030659 E7.

[SpecPar] Cabletron Systems, Inc., Rochester, NH 03867-0505, *Spectrum VnmParmBlock Reference Guide*, März 1994, Order Number: 9030622 E7.

[SpecSof] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum Software Developer's Toolkit Overview*, August 1994, Order Number: 9030621 E4.

[SpecUSE] Cabletron Systems, Inc., Rochester, NH 03867-5005, *System User's Guide*, August 1994, Order Number: 9030263-02 E13.

[SpecVie] Cabletron Systems, Inc., Rochester, NH 03867-5005, *Spectrum View Application Programming Interface Developer's Guide*, März 1994, Order Number: 9030491 E5.

[SpecWat] Cabletron Systems, Inc., Rochester, NH 03867-5005, *SpectroWATCH User's Guide*, Juli 1994, Order Number: 9030919 E4.

[Sta94] William Stallings, *SNMP, SNMPv2, and CMIP: The Practical Guide to Network-Management Standards*, Addison-Wesley, Reading, Massachusetts, 1994.

[Sta95] Michael Stal, "CORBA 2.0 und weitere OMG-Standards — Der Zug rollt weiter", *iX*, Seiten 160–168, Mai 1995.

[Str92] Bjarne Stroustrup, *The C++ programming language*, Addison-Wesley, Reading, Massachusetts, 3. Auflage, 1992.

[SZ92] Alexander Schill und Martina Zitterbart, "Application- and Network-Management in Open Distributed Processing Environment", In *IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, International Federation of Information Processing, Oktober 1992.

[Wil94] Keith Willets, "Service Management — The Drive for Reengineering", In *IEEE/IFIP Network Operations and Management Symposium, Orlando*, International Federation of Information Processing, März 1994.

[Wör94] Oliver Wörl, "Enterprise Management für das EURIS Projekt der BASF AG", Fachkonzept, ICS GmbH, August 1994.

[X400] "Message Handling Services: Message Handling System and Service Overview", Recommendation X.400, International Telecommunication Union, März 1993.

[X402] "Message Handling Systems: Overall Architecture", Recommendation X.402, International Telecommunication Union, September 1992.

[X411] "Message Handling Systems - Message Transfer System: Abstract Service Definition and Procedures", Recommendation X.411, International Telecommunication Union, September 1992.

[X413] "Message Handling Systems - Message Store: Abstract-Service Definition", Recommendation X.413, International Telecommunication Union, September 1992.

[X419] "Message Handling Systems - Protocol Specifications", Recommendation X.419, International Telecommunication Union, September 1992.

[X420] "Message Handling Systems - Interpersonal Messaging System", Recommendation X.420, International Telecommunication Union, September 1992.

[X435] "Message Handling Systems: Electronic Data Interchange Messaging System", Recommendation X.435, International Telecommunication Union, 1991.

[X440] "Message Handling Systems: Voice Messaging System", Recommendation X.440, International Telecommunication Union, September 1992.

[X480] "Message Handling Systems and Directory Services - Conformance Testing", Recommendation X.480, International Telecommunication Union, September 1992.

[X481] "P2 Protocol: Protocol Implementation Conformance Statement (PICS) Proforma", Recommendation X.481, International Telecommunication Union, September 1992.

[X482] "P1 Protocol - Protocol Implementation Conformance Statement (PICS) Proforma", Recommendation X.482, International Telecommunication Union, September 1992.

[X483] "P3 Protocol - Protocol Implementation Conformance Statement (PICS) Proforma", Recommendation X.483, International Telecommunication Union, September 1992.

[X484] "P7 Protocol - Protocol Implementation Conformance Statement (PICS) Proforma", Recommendation X.484, International Telecommunication Union, September 1992.

[X485] "Message Handling Systems: Voice Messaging System Protocol Implementation Conformance Statement (PICS) Proforma", Recommendation X.485, International Telecommunication Union, September 1992.

[X500] "The Directory — Overview of Concepts, Models and Services", Recommendation X.500, International Telecommunication Union, 1988.

[X501] "The Directory — Models", Recommendation X.501, International Telecommunication Union, 1988.

[X509] "The Directory — Authentication framework", Recommendation X.509, International Telecommunication Union, 1988.

[X511] "The Directory — Abstract Service Definition", Recommendation X.511, International Telecommunication Union, 1988.

[X518] "The Directory — Procedures for Distributed Operation", Recommendation X.518, International Telecommunication Union, 1988.

[X519] "The Directory — Protocol Specifications", Recommendation X.519, International Telecommunication Union, 1988.

[X520] "The Directory — Selected Attribute Types", Recommendation X.520, International Telecommunication Union, 1988.

[X521] "The Directory — Selected Attribute Types", Recommendation X.521, International Telecommunication Union, 1988.

[Zim92] Martin Zimmermann, "Management of Distributed Applications", In *IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, International Federation of Information Processing, 1992.