INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwicklung einer Methodik zur systematischen Gewinnung generischer Konfigurationsvorgänge am Beispiel ausgewählter LAN-Komponenten

Karl Ewald

INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwicklung einer Methodik zur systematischen Gewinnung generischer Konfigurationsvorgänge am Beispiel ausgewählter LAN-Komponenten

Bearbeiter:	Karl Ewald
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Dr. Sebastian Abeck
	Dipl.-Inform. Kirsten Heiler
Abgabedatum:	15. Mai 1995

Diese Arbeit entstand im Rahmen des Münchner Netzmanagement Team (MNM-Team), das sich unter der Leitung von Prof. Dr. Heinz-Gerd Hegering aus Wissenschaftlern der beiden Münchner Universitäten und des Leibniz-Rechenzentrums der Bayerischen Akademie der Wissenschaften zusammensetzt.

An dieser Stelle möchte ich Dr. Sebastian Abeck und Kirsten Heiler dafür danken, daß sie die Betreuung für diese Diplomarbeit übernommen haben. Insbesondere danke ich Kirsten Heiler dafür, daß sie in der hektischen Schlußphase viel Zeit für die Durchsicht des Manuskripts aufgewendet hat.

Allen Beteiligten am Projekt TERRA danke ich für die konstruktive Zusammenarbeit und die heitere und produktive Atmosphäre, die sie erzeugt haben.

Christian Fischer danke ich dafür, daß er mit seiner Arbeit am Rahmenbetriebskonzept und an TERRA den Grundstein für diese Arbeit gelegt hat.

Dieter Bertram danke ich für seine Ausführungen zu den Verfahren, die ebenfalls eine Basis meiner Arbeit bilden, und für die kritische Durchsicht und Kommentierung meines Manuskripts.

Hans Maurer danke ich für viele nützliche Hinweise und für die Unterhaltung insbesondere in der Schlußphase der Arbeit.

Ein weiterer Dank gilt all den Studienkollegen und Mitarbeitern des Lehrstuhls, die mir durch Diskussionen und Hinweise geholfen haben.

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

Abkürzungen

API Application Program Interface

ASN.1 Abstract Syntax Notation One [ISO8824]

AUI Attachment Unit Interface (zum Anschluß externer Transceiver)

BNC Bayonet Neill Concelman connector (für IEEE 10Base-2 Verkabelung)

BOOTP Bootstrap Protocol [RFC951, RFC1542]

CMIP Common Management Information Protocol [ISO9596]

EPROM Erasable Programmable Read-Only Memory

Flash-ROM (auch EEPROM) Electrically Erasable Programmable Read-Only Memory

FOIRL Fiber-Optic Inter Repeater Link (inzwischen 10Base-FL)

IAB Internet Activities Board

IETF Internet Engineering Task Force (Arbeitsgruppe des IAB)

IEEE Institute of Electrical and Electronic Engineers

IP Internet Protocol (RFC 791)

ISO International Organization for Standardization

LAN Local Area Network

LLC Logical Link Control (Schicht 2b im OSI-Referenzmodell)

MAC Media Access Control (Schicht 2a im OSI-Referenzmodell)

MIB Management Information Base

MO Managed Object

OSI Open Systems Interconnections

PC Personal Computer

RAM Random Access Memory (les- und schreibbar, flüchtig)

RARP Reverse Address Resolution Protocol [RFC903]

RFC Request for Comments (Bezeichnung für IAB-Dokumente)

RIP Routing Information Protocol (RFC 1058, RFC 1388)

ROM Read-Only Memory

SMF Systems Management Function [ISO10164]

SMI Structure of Management Information (OSI: [ISO10165], IAB: RFC 1155, RFC 1212)

SNMP Simple Network Management Protocol [RFC1157]

TCP Transmission Control Protocol (RFC 761, RFC 793)

TFTP Trivial File Transfer Protocol [RFC1350]

UDP User Datagram Protocol (RFC 768)

Einleitung

Motivation

Der Einsatz von Computersystemen in unserer Zeit wird bestimmt durch die Vernetzung von Workstations und PCs, die die früher vorherrschenden Mainframe-Installationen weitgehend verdrängt haben. Rechenleistung und Speicherkapazität, für die früher klimatisierte Hallen erforderlich waren, passen heute auf einen Schreibtisch. Außerdem erfordern die Ansprüche der Arbeitsplatzergonomie mehr Rechenleistung für die Benutzerschnittstelle, die zweckmäßigerweise und auch aus Kostengründen durch PCs am Arbeitsplatz bereitgestellt wird.

Aufgrund dieser Entwicklung sind in den Unternehmen Computernetze entstanden, die die verschiedensten Endgeräte miteinander verbinden. Solche Netze bestehen aus Teilnetzen, die durch Koppelelemente wie beispielsweise Hubs, Brücken und Router miteinander verbunden sind.

Da diese Netze eine für den Betrieb eines Unternehmens unverzichtbare Infrastruktur darstellen, muß ihre Funktionsfähigkeit durch das sogenannte Netzmanagement [HA93] sichergestellt werden.

Im Netzmanagement nimmt das Management der Koppelelemente eine Schlüsselstellung ein. Durch geeignete Verfahren ermöglicht es die Einhaltung der geforderten Dienstgüte (*Quality of Service*).

Das Management gliedert sich nach OSI [ISO7498] in fünf Funktionsbereiche:

- Konfigurationsmanagement
- Leistungsmanagement
- Fehlermanagement
- Sicherheitsmanagement
- Abrechnungsmanagement

Hierbei sind die Bereiche Konfigurations- und Fehlermanagement die elementarsten, für die schon sehr früh Werkzeuge entwickelt wurden. Auch das Leistungsmanagement hat beim Betreiben eines Netzes seinen festen Platz. Sicherheitsmanagement und Abrechnung (Accounting) sind dagegen insbesondere im LAN-Bereich erst Gegenstand der Forschung.

Herstellerspezifische Lösungen

Während für den Betrieb (Fehlermanagement, Leistungsmanagement) der Koppelelemente bereits Software verfügbar ist, die die Überwachung von Komponenten mehrerer Hersteller unter *einer* Benutzeroberfläche integriert, ist man beim Konfigurationsmanagement meist auf herstellerspezifische Lösungen angewiesen. Die Konfigurationsparameter sind nämlich zwischen den Geräten verschiedener Hersteller sehr uneinheitlich, ihre Manipulation geschieht folglich in der Regel durch herstellereigene Element-Management-Systeme; gelegentlich ist sogar die Verwendung einer Kommandosprache von einer Konsole aus die einzige Möglichkeit, Einstellungen durchzuführen.

In einer heterogenen Umgebung führt dies zu einer Ansammlung verschiedener Werkzeuge der unterschiedlichen Hersteller mit jeweils eigener Benutzerschnittstelle.

Wegen der unterschiedlichen Bedienphilosophien, Menüstrukturen und Tastenbelegungen ergibt sich ein erhöhter Lernaufwand sowie Bedienungsfehler durch Verwechslung. Außerdem erhöht sich der Wartungsaufwand, weil für jedes der Werkzeuge separate Konfigurationsdateien erstellt und Updates eingespielt werden müssen.

Integrationstechniken

Selbst wenn eine Komponente, meist durch Verwendung des Internet-Management-Protokolls SNMP [RFC1157], von einer Management-Plattform aus konfiguriert werden kann, so sind doch viele der MIB-Variablen herstellerspezifisch, da es Politik der IETF ist, nur solche Parameter zu normieren, die bei allen oder zumindest den meisten Koppelelementen einer Gattung eine wichtige Rolle für das Management spielen [RFC1213]. Der Wettbewerb zwischen den Herstellern zwingt diese jedoch dazu, über solche Mindeststandards hinaus Erweiterungen anzubieten, die dann von Konkurrenten aufgegriffen werden. SNMP-Normen hinken folglich immer hinter der technischen Entwicklung her.

Bei der Verwendung einer Managementplattform ist man also entweder gezwungen, einzelne MIB-Variablen zu manipulieren, oder man verwendet von der Plattform zur Verfügung gestellte Verfahren, die individuell für die Geräte eines Herstellers geschrieben wurden. Meist werden solche Verfahren von den Komponentenherstellern entwickelt, die das Knowhow über ihre Geräte einbringen können.

Bei der Einbindung von Software in eine Managementplattform kann zwischen drei Verfahren unterschieden werden [Abe95]:

- Oberflächenintegration,
- Datenintegration und
- Kommunikationsintegration.

Von Fremdherstellern bereitgestellte Konfigurationssoftware wird meist mittels Oberflächenintegration in die Managementplattform einbezogen. Diese Technik stellt jedoch eine „flache“ Integration dar: Es handelt sich um unabhängige Programme, von denen lediglich eines das andere startet und die sich gleichzeitig als visuelle Einheit dem Benutzer präsentieren.

Da ein solches Konfigurationsprogramm nur scheinbar Teil einer Managementplattform ist, legt es seine Daten, beispielsweise solche über die bereits erfolgte Konfiguration oder auch über die Topologie des Netzes, in ihm eigenen Datenbanken ab. Die Verwendung mehrerer solcher Programme in einer heterogenen Umgebung hat daher eine mehrfache Datenhaltung zur Folge. Neben doppelter Arbeit bei der Erstellung kann dies auch rasch zur Inkonsistenz insbesondere der Topologieinformation führen. Ferner bleibt das Problem unterschiedlicher Bedienungsphilosophien und -weisen bestehen, da die Programme verschiedenen Ursprungs sind und die Oberflächenintegration höchstens einen groben Rahmen für die Benutzerschnittstelle vorgeben kann.

Zusammenfassend läßt sich sagen, daß für die Konfiguration eines Koppelelementes gegenwärtig herstellerspezifische Kenntnisse, sowohl was die Eigenschaften der Geräte selbst betrifft als auch hinsichtlich der verwendeten Software, erforderlich sind, so daß ein Verwalter jedes Gerät einzeln „erlernen“ muß.

Zielsetzung und Vorgehensweise

Erstrebenswert ist hingegen eine „tiefe“ Integration, bei der *ein* Programm das Schema für den Konfigurationsvorgang bei einem Koppelelementetyp enthält und je nach vorliegender Hardware diese auf geeignete Weise anspricht, um die Konfiguration durchzuführen. Dies würde Datenintegration oder Kommunikationsintegration bedeuten. Bei der am weitesten gehenden Kommunikationsintegration wird ein standardisiertes Managementprotokoll verwendet, um auf die zu manipulierende Managementinformation zuzugreifen. Bei der Datenintegration greift die

Managementplattform über andere Wege auf diesen Datenbestand zu, beispielsweise durch Manipulation von Dateien oder Zugriff auf proprietäre Datenbanken. Eine Mischform aus Kommunikations- und Datenintegration ist die Verwendung eines Vermittlungsprogramms, genannt Proxy. Dabei wird der Proxy durch ein Standard-Protokoll angesprochen, setzt die Anfragen aber zur Weiterleitung an das betreffende Gerät in ein proprietäres Protokoll um. Zwischen Managementplattform und Proxy kann dann Kommunikationsintegration betrieben werden, obwohl das zu integrierende Gerät kein Standardprotokoll unterstützt.

Die vorliegende Arbeit soll die Grundlage dafür liefern, daß die Konfigurationsvorgänge durch ein „tief“ integriertes Werkzeug vereinheitlicht werden können. Dazu muß eine generische, also herstellerunabhängige, Beschreibung für die Konfigurationsvorgänge erarbeitet werden. Die Arbeit gliedert sich in zwei Teile.

Im ersten Teil wird nach Definition der verwendeten Kriterien eine Analyse bestehender, von Element-Management-Systemen unterstützter Konfigurationsschritte am Beispiel eines bestimmten Gerätetyps durchgeführt. Aus den gewonnenen Ergebnissen wird daraufhin ein gemeinsames Schema der für die Inbetriebnahme einer Komponente erforderlichen Aktionen herausgearbeitet.

Im zweiten Teil werden die entwickelten generischen Einzelaktionen zu Verfahren zusammengefaßt. Dabei stützt sich die Arbeit auf das am Lehrstuhl entwickelte Rahmenbetriebskonzept, das die Beschreibung von Arbeitsabläufen in einem Unternehmen formalisiert. Die hier verwendete Verfahrensbeschreibung baut auf die Verfeinerung dieser Ebene im Rahmenbetriebskonzept bei [Ber95] auf.

Um eine tiefere Integration verschiedener Geräte in ein Managementwerkzeug zu ermöglichen, werden die Verfahren herstellerunabhängig (generisch) dargestellt. Erst auf der Ebene der einzelnen Konfigurationsschritte erfolgt eine Abbildung auf die herstellerspezifische Vorgehensweise.

Abschließend wird ein Konzept für ein Werkzeug zur Automatisierung dieser Konfigurationsverfahren entwickelt. Dieser Teil der Arbeit knüpft an die Entwicklung des „Tools zur Erstellung eines Rahmenbetriebskonzepts“ (TERRA) am Lehrstuhl an [Fis95b, FBE+94, FE94]. Hierbei wird der Schwerpunkt auf die Umsetzung der Generik und der Verfahrenssteuerung in ein Werkzeug gelegt.

Ergebnisse

Durch Erweiterung des Rahmenbetriebskonzepts konnte ein Rahmen geschaffen werden, der sich gut für die Beschreibung generischer Verfahren und Aktionen und deren Umsetzung auf herstellerspezifische Zugriffsmethoden eignet.

Die detaillierte Analyse bestehender Konfigurationsverfahren mittels einer Bottom-

Up Strategie führt zu generischen Beschreibungen, die trotz ihres allgemeinen Charakters eine große Nähe zu den konkreten Konfigurationsverfahren aufweisen. Individuelle Fähigkeiten einzelner Geräte können verfügbar gemacht werden.

Für die Dokumentation der erforderlichen Objekte (generische Verfahren, generische und spezifische Aktionen, d. h. Konfigurationsschritte) wurde eine formale Sprache definiert. Durch die Einführung einer kleinen Zahl stark parametrisierter Basisoperationen können die verwendeten Verfahren weitgehend werkzeugunabhängig beschrieben werden.

Ein generisches Konfigurationswerkzeug, das mit Hilfe dieser Basisoperationen die formal beschriebenen Verfahren interpretiert, kann mit vergleichsweise geringem Aufwand implementiert werden. Dazu wird ein konkretes Implementierungskonzept vorgestellt.

Teil I

Untersuchung bestehender Konfigurationsaktionen

Kapitel 1

Entwurf eines Modells

Die vorliegende Arbeit will mit Hilfe einer Bottom-Up Strategie von der Untersuchung einiger Komponenten zu einer Verallgemeinerung des Konfigurationsvorgangs kommen. Um jedoch diese Untersuchung durchführen zu können, wird zunächst ein Modell für die Generik vorgestellt, das als Rahmen für die zu gewinnenden Erkenntnisse dient.

Nach der Auswertung der Untersuchung kann dieses Modell durch die gewonnenen Erkenntnisse bestätigt und verfeinert werden.

Da das hier entwickelte Modell einerseits generische Verfahren zur Verfügung stellt, sich andererseits aber auf konkrete Werkzeuge und Datenbestände abstützen soll, muß eine Zuordnung vom Generischen zum Speziellen erfolgen. Im folgenden wird die Ebene der Aktionen definiert und dargelegt, warum sie als Schnittstelle zwischen generischer und spezieller Beschreibung dient. Die hier verwendete Definition der Aktionen orientiert sich am Rahmenbetriebskonzept, das in Kapitel 5 vorgestellt wird.

1.1 Der Begriff der Aktion

Da die Struktur eines Konfigurationsvorganges von der verwendeten Hardware unabhängig ist – d. h. die meisten Arbeitsschritte müssen, wenn auch in verschiedener Ausprägung, bei allen Koppelelementen eines Typs gleichermaßen durchgeführt werden –, soll nicht schon auf der Ebene eines ganzen Vorganges eine Abbildung auf ein einzelnes Gerät erfolgen. Damit wäre nichts gewonnen, denn die Spezialisierung auf der Verfahrensebene ist die momentan verwendete Technik. Sie führt zu einer großen Schar ähnlicher Verfahren für die verschiedenen eingesetzten Geräte, außerdem muß für jedes neue Gerät im Grunde ein neues, wenn auch den vorhandenen ähnliches, Verfahren definiert werden.

Das in Kapitel 8 zu entwickelnde Werkzeug muß dazu in der Lage sein, bereits bestehende Element-Manager und Management-Plattformen in die Durchführung einer Konfiguration miteinzubeziehen, um die von diesen zur Verfügung gestellten komplexen Operationen auf Geräten oder Gerätegruppen auszunützen. Andererseits müssen auch solche Aktionen modelliert werden können, die von den eingesetzten Werkzeugen nur unzureichend oder nicht speziell unterstützt werden und daher typischerweise durch das Setzen einzelner MIB-Variablen erbracht werden.

Daraus ergibt sich eine große Bandbreite von Abstraktionsgraden der zu modellierenden Aktionen. Auf diese Problematik wird weiter unten eingegangen werden.

Eine Aufgliederung aller Konfigurationsvorgänge bis hin zum Setzen einzelner MIB-Variablen, wie das für SNMP-gemanagte Geräte nahe liegen würde, wird damit ausgeschlossen. Die Reduktion auf Variablenmanipulation ist aber auch aus anderen Gründen nicht erstrebenswert: erstens steigert eine solche Detaillierung den Umfang der Beschreibungen erheblich und macht sie somit unübersichtlich und schwer durchschaubar, andererseits vernachlässigt eine solche Sichtweise die im OSI-Managementmodell vorgesehene mächtigere Zugriffsweise auf die in den Managementobjekten enthaltenen *Actions*. Eine Action ist eine Operation auf einem MO insgesamt, die durch gleichzeitige Manipulation einer Reihe seiner Attribute realisiert wird, z. B. „Reset MO“; die Action wird von dem im Gerät enthaltenen Management-Agenten als atomare Operation bereitgestellt.

Attribute sind daher nicht als Schnittstelle von der generischen zur speziellen Beschreibung geeignet, außerdem unterscheidet sich nicht nur die Lage der Attribute (z. B. im MIB-Baum), sondern auch die zum Zugriff darauf erforderlichen Werkzeugfunktionen. Dieser Unterschied kann von der Attributebene nicht erfaßt werden.

Die gesuchte Schnittstelle muß also auf der Ebene der Aktionen liegen. Dieser Begriff, dessen Einordnung in ein Verfahrenskonzept der zweite Teil erläutert, wird bereits hier definiert:

Bei der *Aktion* handelt es sich um einen Arbeitsschritt der Konfiguration, der beispielsweise abgebildet werden kann auf eine Operation mit dem verwendeten Element-Manager oder Management-Plattform, oder – bei direktem SNMP-Zugriff – auf eine oder mehrere SET-Operationen auf MIB-Variablen. Diese Zuordnung erfolgt in der Umsetzung der generischen Aktion auf die jeweils spezifische Operation für ein Gerät.

Nicht alle von einem Gerät zur Verfügung gestellten Aktionen lassen sich gleich gut verallgemeinern. So ist einerseits zwischen

- bei allen Geräten des Typs zur Konfiguration erforderlichen (obligatorischen) Aktionen und
- nur von bestimmten Geräten unterstützten (optionalen) Aktionen

und andererseits zwischen

- generischen Aktionen, die herstellerunabhängig beschrieben werden, und
- spezifischen Aktionen in Verbindung mit einem bestimmten Koppelelement, die so eigenwillig sind, daß sie sich einer Verallgemeinerung entziehen,

zu unterscheiden.

Diese beiden Abgrenzungen sind keineswegs gleichbedeutend, vielmehr ist es sinnvoll, auch solche optionalen Aktionen, die sich bei mehreren Instanzen eines Koppelelementetyps wiederfinden, in die Generik miteinzubeziehen. Die möglichen Aktionentypen ergeben sich also wie in Bild 1.1 dargestellt.

<table>
<tr><td colspan="3">alle möglichen Aktionen</td></tr>
<tr><td>obligatorische Aktionen</td><td colspan="2">optionale Aktionen</td></tr>
<tr><td colspan="2">generische Aktionen</td><td>spezifische Aktionen</td></tr>
</table>

Abbildung 1.1: Einteilung der Aktionen in Typen

Der Begriff optional in obiger Aufstellung ist etwas mißverständlich: hier bedeutet er nur, daß nicht alle Koppelelemente einer Klasse die Aktion unterstützen. Daraus folgt jedoch nicht, daß bei einem gegebenen Koppelelement eine „optionale“ Aktion nicht ausgeführt werden müßte.

1.2 Stufen der Generik

Die Zusammenfassung aller Koppelelemente eines Typs (z. B. Repeater) ist primäres Ziel dieser Arbeit. Des weiteren wird zu untersuchen sein, inwieweit man Aktionen finden kann, die bei verschiedenen Typen von Koppelelementen (z. B. Sternkoppler und Router) gleich sind. In diesem Fall wäre eine weitere Abstraktion möglich (Bild 1.2).

Eine andere mögliche Abstraktionsrichtung, die bei einigen Element-Managern bereits besteht, ist die Zusammenfassung voneinander abhängiger Komponenten zu einer logischen Einheit. Ein offensichtliches Beispiel ist ein Paar Remote Bridges, jedoch ist auch eine Gruppe von Routern denkbar, beispielsweise ein zentraler Router und mehrere Access Router, die die WAN-Anbindung der Filialen bedienen und deren Konfiguration in engem Zusammenhang mit der des zentralen Routers steht.

Die später zu definierende Beschreibungssprache muß Aktionen auf solchen Gerätegruppen, sowie deren Abbildung auf Aktionen an den entsprechenden Geräten,

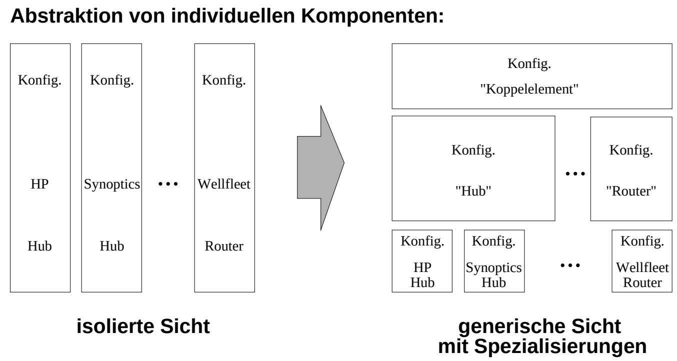

Abbildung 1.2: Stufen der Generik

soweit diese nicht durch den verwendeten Element-Manager durchgeführt wird, unterstützen.

Neben der Dokumentation bestehender Arbeitsabläufe besteht ein weiteres Ziel darin, die Möglichkeit zu schaffen, daß unter einer einheitlichen Benutzerschnittstelle mit einem Interpreter die in dieser Arbeit entwickelte Beschreibung der Konfigurationsvorgänge für die einzelnen Geräte abgearbeitet werden kann.

So wird der Verwalter von der Kenntnis der individuellen Konfigurationsmethoden und -parameter weitgehend entlastet. Sein Wissen über individuelle Geräte kann sich nun auf diejenigen herstellerspezifischen Erweiterungen beschränken, die nicht in die Generik einbezogen wurden. Auch in diesem Fall hilft die Benutzerführung eines integrierten Management-Werkzeuges durch gezielte Abfrage der benötigten Parameter und verhindert so, daß Konfigurationsschritte vergessen werden. Bei der Einordnung der Aktionen wäre es darüberhinaus wünschenswert, daß nur verzichtbare Aktionen als herstellerspezifisch definiert werden, so daß auch bei der Beschränkung auf generische Aktionen eine Inbetriebnahme erfolgen kann.

Diese Arbeit behandelt einen bestimmten Typ von Koppelelement, den Sternkoppler, versucht dafür aber, aus dem Vergleich unterschiedlicher Vertreter dieses Typs ein umfassendes Modell zu erstellen.

Auf die Anwendung des Modells für andere Koppelelementetypen wird im Ausblick eingegangen.

Kapitel 2

Planung der Untersuchung

Die Verallgemeinerung des Konfigurationsvorgangs erfordert die Untersuchung der erforderlichen Schritte an einer Reihe von Komponenten.

2.1 Gegenstand

Wie eingangs erläutert, sind die Koppelelemente Gegenstand des Konfigurationsmanagements eines Netzes. Daher werden in diesem Abschnitt beispielhaft einige Koppelelemente herausgegriffen und auf ihre Konfigurationsvorgänge hin untersucht. Aus dem Vergleich der Konfigurationsschritte bei diesen Komponenten ergeben sich Muster in bezug auf die Funktionen und Parameter, aus denen durch Verallgemeinerung eine Generik abgeleitet werden kann.

Nach dem OSI-Schichtenmodell [ISO7498] lassen sich drei Klassen von Koppelelementen unterscheiden: Sternkoppler auf Schicht 1, Brücken auf Schicht 2 und Router auf Schicht 3. Eine Betrachtung all dieser Komponentenklassen würde den Umfang dieser Arbeit sprengen. Da es hier um die Entwicklung einer Methodik geht, ist eine so breit angelegte Betrachtung auch nicht erforderlich.

Als Komponententyp wird daher die Klasse der Ethernet Sternkoppler (auch Hubs oder Repeater genannt) ausgewählt. Die zu untersuchende Verfahrensweise ist davon unberührt; da bei Hubs die Anzahl der Konfigurationsparameter vergleichsweise klein ist, bietet diese Klasse jedoch die Chance, innerhalb dieser Arbeit eine umfassende Betrachtung vorzunehmen.

2.2 Kriterien

Üblicherweise werden Konfigurationsabläufe für ein bestimmtes Gerät in ihrer Gesamtheit dargestellt. Ziel dieser Untersuchung ist hingegen, nicht ganze Verfahren, sondern einzelne Verfahrensschritte, nämlich die oben definierten Aktionen, zu identifizieren und zu dokumentieren, die im zweiten Teil der vorliegenden Arbeit als Bausteine generischer Verfahren verwendet werden.

Die Zielsetzung dieser Arbeit ist zweifach, und dies wirkt sich auch auf die Untersuchung aus. Einerseits werden die Aktionen unter Verwendung der zur Verfügung stehenden Software, insbesondere herstellereigener Element-Management-Systeme, dargestellt. So ergibt sich ein Bild des Status-Quo im Konfigurationsmanagement der untersuchten Komponenten. Die im dritten Teil der Arbeit entwickelte Software soll nämlich gleichsam als interaktives Handbuch zur Unterstützung manueller Konfigurationsverfahren eingesetzt werden können.

Andererseits wird die Entwicklung eines Werkzeugs angestrebt, das die vom generischen Verfahren vorgezeichneten Aktionen, abgebildet auf die konkreten Aktionen für das zu konfigurierende Gerät, automatisch durchführt. Hier ist es zweckmäßig, die Aktionen nicht auf der Basis eines Element-Managers darzustellen, der auf die Benutzerschnittstelle hin entwickelt wurde, sondern auf einer Ebene, die sich an den zugrundeliegenden Managementschnittstellen und -protokollen orientiert, beispielsweise auf der Basis von MIB-Variablen bei SNMP-gemanagten Geräten oder auf Funktionen, die von einer API eines Werkzeugs bereitgestellt werden.

Die Untersuchung des Element-Managers dient gleichzeitig dazu, Anforderungen an die Mächtigkeit und die Benutzerschnittstelle des zu entwickelnden generischen Managementwerkzeugs zu gewinnen.

Die Betrachtung der API verschiedener Managementplattformen oder Element-Manager würde den Rahmen dieser Arbeit sprengen. Daher wird sich die Betrachtung auf manuelle Aktionen an der Benutzerschnittstelle eines Managementwerkzeugs einerseits und der Manipulation von MIB-Variablen andererseits beschränken. Diese beiden Methoden werden im dritten Teil bei der prototypischen Implementierung eines Werkzeugs aufgegriffen.

Für jeden Konfigurationsschritt sind die erforderlichen Informationen (Eingabeparameter) nach Funktion (Semantik) und Datentyp zu erfassen.

Die Untersuchung soll zur Erstellung eines generischen Verfahrens „Konfiguration eines Sternkopplers“ dienen. Dazu müssen die Aktionen nach ihrem Stellenwert für die Generik (vgl. Bild 1.1) eingeteilt werden.

Da die grundsätzliche Funktionalität innerhalb einer Klasse von Koppelelementen, abgesehen vom Entwicklungsstand eines Geräts, weitgehend übereinstimmt, ist zu erwarten, daß ein großer Anteil der bei der Konfiguration benötigten Aktionen in die Generik einbezogen werden kann.

2.3 Profil für Aktionen

Bevor die einzelnen Geräte betrachtet werden, folgt hier als Ergebnis der obenstehenden Diskussion eine Zusammenfassung der bei jeder Aktion zu ermittelnden Daten.

Typ der Aktion:

- obligatorisch, generisch
- optional, generisch
- optional, spezifisch

für jeden Eingabeparameter:

- Bedeutung (Semantik)
- Datentyp

Beschreibung der Durchführung der Aktion:

- durch Element-Manager oder Konsole
- durch Setzen von MIB-Variablen oder Datei-Einträgen

Kapitel 3

Analyse ausgewählter Koppelelemente

In diesem Abschnitt werden die untersuchten Geräte vorgestellt und auf ihre Konfigurationsaktionen hin untersucht. Der Typ der Aktionen (vgl. Abschnitt 1.1) wird vorläufig bestimmt. Die Auswertung erfolgt im nächsten Abschnitt.

3.1 HP Fiber-Optic Hub Plus

Der HP Fiber-Optic Hub Plus ist ein 10-Port-Sternkoppler in einer kompakten, nicht-modularen Bauform. Er verfügt über acht Glasfaser-Anschlüsse nach dem IEEE FOIRL-Standard (Fiber-Optic Inter Repeater Link, inzwischen 10 Base-FL), je einen BNC- und AUI-Port sowie einen seriellen Konsolenport für Managementzwecke.

Neben der Repeater-Funktionalität nach IEEE 802.3 bietet der Hub die Möglichkeit, einen Port zu definieren, der laufend überwacht werden soll und bei Ausfall durch einen normalerweise inaktiven Backup-Port ersetzt wird. Die Rückschaltung der Verbindung vom Backup- auf den Primärport muß manuell durch den Verwalter (oder durch einen Automatismus in einer Management-Plattform) durchgeführt werden.

Das Management des HP Hub erfolgt entweder über ein am Konsolenport angeschlossenes Terminal oder von einer Management-Plattform aus über SNMP. Dabei wird die standardisierte MIB-II unterstützt, die durch eine HP-eigene MIB ergänzt wird. HP bietet keine Element-Manager Software für den Hub an.

Folgende Aktionen werden für die Konfiguration benötigt:

1. Anschluß eines Terminals am Konsolenport des Hubs

Einordnung: rein mechanischer Vorgang, der zur Installation gerechnet werden kann und daher hier nicht betrachtet wird.

2. Vergabe einer IP-Adresse

   Einordnung: obligatorische, generische Aktion.

   Dieser Schritt kann nur direkt über die Konsole ausgeführt werden, da die IP-Adresse Voraussetzung für den SNMP-Zugriff ist.

   Benötigte Daten:

   - *hubip* IP-Adresse des Hubs,
   - *netmask* Subnet Mask für das zugehörige Netzsegment,
   - *router* IP-Adresse des Routers in Richtung Netzmanagement-Station,
   - *backup* IP-Adresse eines Ersatzrouters, falls der unter *router* angegebene Router nicht verfügbar ist,
   - *ttl* Time-To-Live für vom Hub gesendete Pakete. Der Standardwert ist 32.

   Aktion: Kommando `IP` geben. Auf die Anfragen der Reihe nach *hubip*, *netmask*, *router*, *backup*, *ttl* eingeben.

Die folgenden Konfigurationsschritte können wahlweise über die Konsole oder von einer Managementplattform aus über das SNMP-Protokoll erfolgen.

3. nichtverwendete Ports deaktivieren

   Einordnung: obligatorische, generische Aktion

   Dieser Schritt ist für jeden Port (numeriert von 1 bis 10) auszuführen, der nicht verwendet wird. *n* ist die Portnummer.

   Bei Auslieferung sind alle Ports des Hubs aktiviert, daher braucht man mit den tatsächlich verwendeten Ports nichts tun.

   Benötigte Daten:

   - *hubip* IP-Adresse des Hubs (für SNMP)
   - *n* Portnummer

   Konsole: `PO` *n* `OFF`

   SNMP an *hubip*:
   in `.iso.org.dod.internet.mgmt.mib-2`:
   `interfaces.ifTable.ifEntry.ifAdminStatus.`*n* = `down`

4. Backup-Link definieren

   Einordnung: optionale, generische Aktion

   Benötigte Daten:

   - *hubip* IP-Adresse des Hubs (für SNMP)
   - *backup* Backup-Port: dieser Port wird nur benutzt, wenn *primär* ausfällt.
   - *primär* Primär-Port: die an diesem Port angeschlossene Verbindung wird vom Hub auf Funktionsfähigkeit überwacht, bei Ausfall wird der Port deaktiviert und stattdessen *backup* aktiviert.
   - *remote* Ethernet-Adresse am anderen Ende der Primär- und Backup-Link.
   - *seconds* Pause in Sekunden zwischen aufeinanderfolgenden Testpaketen, die die Primärlink prüfen.
   - *fails* Anzahl aufeinanderfolgender Fehler, die auftreten müssen, damit die Primärlink als fehlerhaft angesehen und das Backup aktiviert wird.

   Konsole: Kommando `BA` geben. Auf die Anfragen der Reihe nach *backup*, *primär*, *remote*, *seconds*, *fails* eingeben.

   SNMP an *hubip*:
   in `.iso.org.dod.internet.private.enterprises.hp.nm`:
   `icf.icfHub.hubBkpLinkTable.hubBackupPort.1` = *backup*
   `icf.icfHub.hubBkpLinkTable.hubPrimaryPort.1` = *primär*
   `icf.icfHub.hubBkpLinkTable.hubBackupAddress.1` = *remote*
   `icf.icfHub.hubBkpLinkTable.hubBackupTestTime.1` = *seconds*100*
   `icf.icfHub.hubBkpLinkTable.hubBackupConsecutiveFails.1` = *fails*

5. ggf. ThinWatch aktivieren

   Einordnung: optionale, herstellerspezifische Aktion

   Bei der Auslieferung des Hubs sind alle Ports aktiviert, ein nicht verkabelter 10Base2-Anschluß erweckt jedoch den Eindruck, es würden dauernd Kollisionen auftreten. Daher wird dieser Zustand nicht durch die Fehler-LED am Hub erfaßt, weil sonst bei Nichtverwendung dieses Ports immer ein Fehler angezeigt würde.

   Falls der ThinLAN- (10Base2-) Port verwendet wird, so sollte auch ein Fehler auf diesem Port angezeigt werden, ThinWatch sollte also aktiviert werden.

   Konsole: Kommando `TH ON` geben.

SNMP:
in `.iso.org.dod.internet.private.enterprises.hp.nm`:
`icf.icfHub.hubThinlanFault = 1`

Die Einbindung des neu installierten Hubs in eine Managementplattform sowie ggf. Netzdokumentationssoftware wird hier nicht betrachtet, da sie nicht von dem zu installierenden Gerät, sondern von der Betriebsumgebung abhängt. Wenn die weitere Konfiguration von dieser Managementplattform aus über SNMP durchgeführt werden soll, so muß diese Einbindung bereits vor Schritt 2 der Konfiguration erfolgen.

Es wäre denkbar, daß eine Konfigurationsplattform über ein serielles Kabel die Konfiguration der IP-Adresse vornimmt, jedoch müßte dann der Hub vor der Installation zum Arbeitsplatz des Netzmanagers gebracht und dort vorübergehend angeschlossen werden.

3.2 Fibermux Crossbow modularer Hub

Der Fibermux Crossbow ist ein modularer Hub, der neben Ethernet auch AppleTalk, Token Ring und Apollo Domain Segmente unterstützt. Es können bis zu vier Ethernet- oder AppleTalk-Segmente innerhalb des Hubs gebildet werden, die sich wie separate Hubs verhalten. Beim Einsatz als Token Ring Hub kann der modulübergreifende Ring des Hubs in Abschnitte jeweils aufeinanderfolgender Module geteilt werden, um mehrere unabhängige Token Ring Segmente im Hub zu ermöglichen.

Das Management erfolgt über spezielle, sogenannte *SmartLink*-Module. Ein Modul reicht aus, um den Zustand aller Ports zu steuern und den Hub zu konfigurieren, jedoch kann eine Überwachung des Datenverkehrs nur auf denjenigen Segmenten erfolgen, die ein eigenes SmartLink-Modul besitzen.

Die Zuordnung der einzelnen Module (von Fibermux *cards* genannt; je nach Modell 2, 4, 10 oder 14 pro Chassis) auf die vier möglichen Segmente innerhalb des Hubs sowie die Vergabe der Modulnummern (die zur besseren Orientierung mit den verwendeten Steckplätzen im Gehäuse übereinstimmen sollten) erfolgt über Steckbrücken auf dem Modul oder bei neueren Geräten softwaremäßig.

Zur Adressierung einzelner Ports wird ein Tupel (c, p) aus Modul- (*card*-)nummer und Portnummer auf dem Modul eingesetzt, um die modulare Bauform zu reflektieren. Maximal 16 Ports pro Modul sind vorgesehen (Bild 3.1). Im folgenden wird ein solches Tupel als Portnummer bezeichnet.

Fibermux bietet das Element-Management-System *LightWatch* an, das für PCs unter DOS/MS-Windows und für Sun SPARCstations unter SunOS 4.1.x und

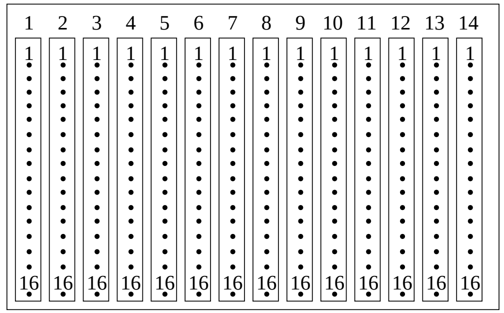

Abbildung 3.1: Portnummern beim Fibermux Crossbow

OpenWindows oder X11 verfügbar ist. Die Unix-Version dieser Software kann mit SunNet Manager oder HP OpenView oberflächenintegriert werden.

Folgende Schritte sind bei der Konfiguration erforderlich.

1. Zuweisung einer IP-Adresse

   Einordnung: obligatorische, generische Aktion.

   Dieser Konfigurationsschritt kann entweder über den Konsolenport des Hubs oder über das RARP-Protokoll von einer im gleichen Segment befindlichen Managementstation (oder einem anderen Host) aus erfolgen.

   Das RARP (Reverse Address Resolution Protocol, [RFC903]) bietet Einheiten, die das IP verwenden wollen, die Möglichkeit, ihre eigene IP-Adresse zu erfragen. Es ist hauptsächlich für diskless Workstations und Peripheriegeräte gedacht, die über keinen eigenen nichtflüchtigen Speicher verfügen, wo die eigene IP-Adresse permanent abgelegt werden könnte. Dies geschieht durch einen RARP-Request, der vom Gerät als Broadcast gesendet wird. Ein RARP-Server nimmt die Anfrage entgegen und teilt dem Gerät die aufgrund der MAC-Adresse ermittelte IP-Adresse mit.

   Im Gegensatz zum HP Fiber-Optic Hub Plus muß also kein Schritt der Konfiguration zwingend am Gerät selbst erfolgen. Das ist ein wesentlicher Vorteil, weil dann der gesamte Konfigurationsvorgang von zentraler Stelle aus durchgeführt werden kann.

   Benötigte Daten:

   - *hubip* IP-Adresse des Hubs (oder 0.0.0.0, falls die Adresse über RARP zugewiesen wird; dies ist die Voreinstellung ab Werk),

- *netmask* Subnet Mask für das zugehörige Netzsegment,
- *router* IP-Adresse des Routers in Richtung Netzmanagement-Station (oder 0.0.0.0, falls die Netzmanagementstation direkt erreichbar ist oder der Router RIP (Routing Information Protocol) verwendet),
- *traphost* IP-Adresse des Hosts, der SNMP Trap-Pakete vom Hub erhalten soll (i. d. R. die Management-Station).

Konsole: zwei ESC-Zeichen senden, um das Menü zu erhalten. Punkt 1 "Set IP Addresses" wählen. Nacheinander *hubip*, *netmask*, *router*, *traphost* eingeben.

RARP: Die Adresse *hubip* ist als RARP-Adresse für die Ethernet-Adresse des Hubs einzutragen.

Das RARP-Protokoll sieht die Übermittlung einer Netmask nicht vor. Die Voreinstellung der Netmask erfolgt in Abhängigkeit von der eingegebenen oder über RARP ermittelten Adresse.

Ein RARP-Request wird nur nach einem Reset des Hubs durchgeführt. Als *traphost* wird nach einem RARP-Request die IP-Adresse des RARP-Servers eingetragen. Wird zur Kommunikation mit der Management-Station ein Gateway benötigt, so muß dieses dem Hub über RIP mitgeteilt werden.

2. Aktivierung oder Deaktivierung eines Ports

Einordnung: obligatorische, generische Aktion

Benötigte Daten:

- *hubname* Name des Hubs (für LightWatch)
- *hubip* IP-Adresse des zuständigen SmartLink-Moduls (für SNMP)
- *port* Portnummer

Zur Aktivierung:

SNMP auf *hubip*:

in `.iso.org.dod.internet.private.enterprises.fibermux.`
`fmxVariables.fmxHubs.fmxCrossbowGrp`:
`fmxCbCPortTable.fmxCbCPortEntry.fmxCbCPEnable.`*port-c*`.`*port-p* = 1

LightWatch:

- Selektieren des Hubs *hubname* mit der Maus
- Menü `Hub/Detail...` auswählen. Ein `Hub Detail` Fenster erscheint.
- Selektieren des Ports *port-p* auf dem Modul *port-c* mit der Maus
- Menü `Control/Enable` auswählen.

oder

- Selektieren des Hubs mit der Maus
- Menü `Hub/Detail...` auswählen. Ein `Hub Detail` Fenster erscheint.
- Selektieren des Moduls *port-c* mit der Maus
- Menü `Display/Card Info` auswählen. Ein `Card Info` Fenster erscheint.
- Anklicken der Checkbutton `Ena/Dis` in Zeile *port-p*, so daß der Haken erscheint.

Zur Deaktivierung:

SNMP an *hubip*:

in `.iso.org.dod.internet.private.enterprises.fibermux.`
`fmxVariables.fmxHubs.fmxCrossbowGrp`:
`fmxCbCPortTable.fmxCbCPortEntry.fmxCbCPEnable.`*port-c*`.`*port-p* = 0

LightWatch:

- Selektieren des Hubs *hubname* mit der Maus
- Menü `Hub/Detail...` auswählen. Ein `Hub Detail` Fenster erscheint.
- Selektieren des Ports *port-p* auf dem Modul *port-c* mit der Maus
- Menü `Control/Disable` auswählen.

oder

- Selektieren des Hubs mit der Maus
- Menü `Hub/Detail...` auswählen. Ein `Hub Detail` Fenster erscheint.
- Selektieren des Moduls *port-c* mit der Maus
- Menü `Display/Card Info` auswählen. Ein `Card Info` Fenster erscheint.
- Anklicken der Checkbutton `Ena/Dis` in Zeile *port-p*, so daß der Haken verschwindet.

3. Aktivierung oder Deaktivierung eines Moduls

Bei modularen Geräten wie dem Fibermux Crossbow gibt es über die Deaktivierung einzelner Ports hinaus auch die Möglichkeit, ganze Module (*cards*) auszuschalten.

Einordnung: optionale, generische Aktion

Benötigte Daten:

- *hubname* Name des Hubs (für LightWatch)
- *hubip* IP-Adresse des zuständigen SmartLink-Moduls (für SNMP)
- *modul* Modulnummer

Zur Aktivierung:

SNMP an *hubip*:

in `.iso.org.dod.internet.private.enterprises.fibermux.`
`fmxVariables.fmxHubs.fmxCrossbowGrp:`
`fmxCbCardTable.fmxCbCardEntry.fmxCbCEnable.`*modul* `= 1`

LightWatch:

- Selektieren des Hubs *hubname* mit der Maus
- Menü `Hub/Detail...` auswählen. Ein `Hub Detail` Fenster erscheint.
- Selektieren des Moduls *modul* mit der Maus
- Menü `Control/Enable` auswählen.

Zur Deaktivierung:
SNMP an *hubip*:

in `.iso.org.dod.internet.private.enterprises.fibermux.`
`fmxVariables.fmxHubs.fmxCrossbowGrp:`
`fmxCbCardTable.fmxCbCardEntry.fmxCbCEnable.`*modul* `= 0`

LightWatch:

- Selektieren des Hubs *hubname* mit der Maus
- Menü `Hub/Detail...` auswählen. Ein `Hub Detail` Fenster erscheint.
- Selektieren des Moduls *Modul* mit der Maus
- Menü `Control/Disable` auswählen.

4. Konfiguration redundanter Verbindungen

Fibermux erlaubt die Definition beliebig vieler Portpaare eines Hubs als redundante Verbindungen. Dabei wird zwischen Hub-to-Hub und Hub-to-Device Redundanz unterschieden. Voraussetzung für erstere ist, daß sich auch am anderen Ende des Verbindungspaares ein Fibermux Hub befindet, bei dem die gleiche Redundanzbeziehung programmiert werden muß. Wird eine bestimmte Fehlerschwelle überschritten, so erfolgt die Umschaltung auf die Backup-Link. Die Umschaltung erfolgt ferner, wenn das bei

Twisted Pair und Fiber-Optic Verbindungen definierte *Integrity*-Signal verloren geht. Im Gegensatz zum HP Fiber-Optic Hub Plus wird also die Leitung nicht durch eigene Testpakete sondern durch die Fehlerhäufigkeit überwacht. Die Tatsache, daß für Hub-to-Hub Redundanz auf beiden Seiten ein Fibermux-Adapter stehen muß, läßt auf ein proprietäres Protokoll zwischen den Einheiten schließen. Der Backup-Port ist ebenfalls fehlerüberwacht. Bei Überschreiten seiner Fehlerschwelle wird wieder auf den Primärport zurückgeschaltet. Primär- und Backupport verhalten sich im Betrieb also symmetrisch, der Primärport ist es jedoch, der nach einem Neustart aktiviert wird.

Trotz der unterschiedlichen Eigenschaften redundanter Verbindungen zwischen HP Fiber-Optic Hub Plus und Fibermux Crossbow werden beide Konfigurationsaktionen der gleichen generischen Aktion zugeordnet.

Benötigte Daten:

- *hubname* Name des Hubs (für LightWatch)
- *hubname2* Name des zweiten Hubs (für LightWatch, Hub-to-Hub)
- *hubip* IP-Adresse des zuständigen SmartLink-Moduls (für SNMP)
- *backup* Backup-Port: dieser Port wird nur benutzt, wenn *primär* ausfällt.
- *primär* Primär-Port: die an diesem Port angeschlossene Verbindung wird vom Hub auf Funktionsfähigkeit überwacht, bei Ausfall wird der Port deaktiviert und stattdessen *backup* aktiviert.
- *errors* Fehlergrenze: wird die hier angegebene Fehlerquote (in 1/1000 %) überschritten, so schaltet der Hub auf die Backup-Verbindung um.
- *backerr* Fehlergrenze: wird, während die Backup-Verbindung aktiv ist, die hier angegebene Fehlerquote (in 1/1000 %) überschritten, so schaltet der Hub auf die Primär-Verbindung zurück.

SNMP an *hubip*:

Die hier dargestellte Aktion gibt bei Hub-to-Hub Redundanz nur die Konfiguration einer Seite des redundanten Verbindungspaares wieder.

in `.iso.org.dod.internet.private.enterprises.fibermux.`
`fmxVariables.fmxHubs.fmxCrossbowGrp.fmxCbCPortTable:`
`fmxCbCPortEntry.fmxCbCPRedundCtrl.`*primär-c*`.`*primär-p* `= 1`
`fmxCbCPortEntry.fmxCbCPRedundPrimary.`*primär-c*`.`*primär-p* `= 1`
`fmxCbCPortEntry.fmxCbCPRedundCard.`*primär-c*`.`*primär-p* = *backup-c*
`fmxCbCPortEntry.fmxCbCPRedundPort.`*primär-c*`.`*primär-p* = *backup-p*
`fmxCbCPortEntry.fmxCbCPRedundErrorLimit.`*primär-c*`.`*primär-p* = *errors*

`fmxCbCPortEntry.fmxCbCPRedundCtrl.`*backup-c*.*backup-p* = 1
`fmxCbCPortEntry.fmxCbCPRedundPrimary.`*backup-c*.*backup-p* = 0
`fmxCbCPortEntry.fmxCbCPRedundCard.`*backup-c*.*backup-p* = *primär-c*
`fmxCbCPortEntry.fmxCbCPRedundPort.`*backup-c*.*backup-p* = *primär-p*
`fmxCbCPortEntry.fmxCbCPRedundErrorLimit.`*backup-c*.*backup-p* = *backerr*

LightWatch:

Hub-to-Hub Redundanz:

- Selektieren der beiden beteiligten Hubs *hubname*, *hubname2* mit der Maus
- Menü `Redundancy/Hub-to-Hub` auswählen
- in den beiden beteiligten Hubs jeweils den Port der Primärlink *primär* mit der linken und den der Backup-Link *backup* mit der mittleren Maustaste auswählen
- Die Fehlerschwelle *error* unter Primary eintragen
- Die Fehlerschwelle *backerr* unter Secondary eintragen

Hub-to-Device Redundanz:

- Selektieren des Hubs *hubname* mit der Maus
- Menü `Redundancy/Hub-to-Device` auswählen
- den Port der Primärlink *primär* mit der linken und den der Backup-Link *backup* mit der mittleren Maustaste auswählen
- Die Fehlerschwelle *error* unter Primary eintragen
- Die Fehlerschwelle *backerr* unter Secondary eintragen

5. Laden neuer Firmware in den Hub

Das Fibermux SmartLink-Modul wird in verschiedenen Versionen ausgeliefert, wobei die Firmware in RAM, Flash-ROM oder EPROM gespeichert sein kann. Bei der RAM-Version ist es erforderlich, bei jedem Einschalt- oder Bootvorgang die benötigten Programme und Daten über das Netz von einem Server (typischerweise der Management-Station) zu laden, bei der Flash-ROM Version ist dies möglich. Die EPROM-Version kann nur durch einen Eingriff am Gerät mit Austausch dieses Bausteins neuprogrammiert werden.

Zum Laden der Firmware wird das BOOTP-Protokoll [RFC951, RFC1542] verwendet. Wenn der Server eine BOOTP-Anfrage vom SmartLink-Modul erhält, so teilt er diesem seine IP-Adresse und den Namen der Boot-Datei

mit. Die Boot-Datei wird dann mittels TFTP [RFC1350] an das SmartLink-Modul übertragen.

Dazu muß der Name des BOOTP Servers und sein Port im Hub eingestellt werden. Ferner müssen BOOTP und TFTP auf dem Server konfiguriert werden. Wie letzteres geschieht, hängt davon ab, ob der Element-Manager LightWatch verwendet wird. Zusammen mit diesem wird ein eigene Version des BOOTP-Servers verwendet, der auf die Management-Datenbank des Element-Managers zugreift. Somit müssen bei der Konfiguration weniger Parameter angegeben werden, da die übrigen Daten bereits verfügbar sind.

Einordnung: generische, optionale Aktion

Benötigte Daten:

- *hubname* Name des Hubs
- *hubip* IP-Adresse des zuständigen SmartLink-Moduls (für SNMP)
- *bootserver* Name des BOOTP-Servers
- *bootport* Portnummer des BOOTP-Servers. Normalerweise ist dies 67.
- *bootfile* Name der Bootdatei auf dem Server

LightWatch:

- falls die physische Anzeige aktiv ist, mit `View/Logical Map` auf die logische Anzeige umschalten
- Selektieren des logischen Hubs *hubname*, für dessen SmartLink die Firmware konfiguriert werden soll
- Menü `Hub/Download` auswählen. Ein `Download` Fenster erscheint.
- in der Liste die Datei *bootfile* auswählen und mit `OK` bestätigen.

Wenn nicht LightWatch verwendet wird, müssen beide Seiten, der Server durch Editieren der Datei `bootptab` und der Hub mittels SNMP, konfiguriert werden. Zum Inhalt der Konfigurationsdatei `bootptab` siehe Abschnitt 3.3.

SNMP an *hubip*:

in `.iso.org.dod.internet.private.enterprises.fibermux`:
`fmxBootGrp.fmxBootpServer` = *bootserver*
`fmxBootGrp.fmxBootpPort` = *bootport*
`fmxBootGrp.fmxBootpMode` = 1

Konfigurationsdatei `bootptab` auf dem Server:

Einfügen einer Zeile mit folgendem Inhalt:
hubname`:ht=ether:ip=`*hubip*`:bf=`*bootfile*

6. Zuordnung eines Moduls zu einem *SmartLink* Management-Agentenmodul

   Der Fibermux Crossbow erlaubt die Installation von bis zu vier unabhängigen Repeatern in einem Chassis. Dabei wird jedes Modul über Steckbrücken einem der vier Repeaterkanäle des Ethernet-Bus zugeordnet. Bei älteren Karten erfolgte auch die Zuordnung zu einer der vier Management-Gruppen mit Steckbrücken, bei neueren Modellen muß diese jedoch softwaremäßig erfolgen.

   Eine Schwäche der Fibermux-MIB ist, daß die Software nicht feststellen kann, welchem Ethernet-Segment das Modul durch Steckbrücken zugeordnet wurde. Es liegt daher in der Verantwortung des Netzverwalters, die richtige Zuordnung vorzunehmen. Es ist daher auch nicht möglich, bei Verwendung von SNMP selbst festzustellen, welcher Management-Gruppe das Modul zugeordnet werden soll.

   Benötigte Daten:

   - *hubname* Name des Hubs (für LightWatch)
   - *mgmtcard* Nummer des SmartLink-Moduls im Hub
   - *hubip* IP-Adresse des zuständigen SmartLink-Moduls (für SNMP)
   - *card* Nummer der nicht zugeordneten Karte

   LightWatch:

   - falls die logische Anzeige aktiv ist, mit `View/Physical Map` auf die physische Anzeige umschalten
   - Selektieren des Hubs,
   - Menü `Hub/Detail` auswählen. Ein `Detail` Fenster erscheint.
   - Die Module *card* (es ist durch ein ? gekennzeichnet) und *mgmtcard* selektieren.
   - Menü `Assignment/Assign` auswählen.

   SNMP:

   in `.iso.org.dod.internet.private.enterprises.fibermux.`
   `fmxVariables.fmxHubs.fmxCrossbowGrp`:
   `fmxCbCardTable.fmxCbCardEntry.fmxCbCTimeslot.`*card* =
   `fmxCbCardTable.fmxCbCardEntry.fmxCbCTimeslot.`*mgmtcard*

3.3 Synoptics Workgroup Hub

Synoptics bietet einen Workgroup Hub an, der aus einem oder mehreren Hub-Modulen und einem Managementmodul besteht. Die Hubmodule verfügen über jeweils 16 10Base-T und einen fiber-optic Port, die Managementmodule nur über einen fiber-optic Port. Dieser Hub wird bei der Telekom [Tel94] eingesetzt. Die folgende Beschreibung kann mangels anderer Quellen nur die Konfigurationsverfahren beschreiben, die bei der Telekom verwendet werden. Über die vom Synoptics Hub unterstützten MIBs liegen dem Autor keine Informationen vor.

Jedes Hub-Modul erhält seine Konfiguration bei einem Bootvorgang über BOOTP (Bootstrap Protocol). Auch die Firmware wird so übertragen. Der von BOOTP mitgeteilte Dateiname stellt den Namen einer Konfigurationsdatei dar, die anschließend mittels TFTP (Trivial File Transfer Protocol) abgerufen wird. Diese enthält neben den übrigen Konfigurationsparametern auch den Namen der Firmware-Datei, die ebenfalls per TFTP übertragen wird.

Auf dem BOOTP-Server befindet sich eine Datei `bootptab`, die Informationen für jedes Gerät enthält. Eine Zeile kann beispielsweise folgendermaßen aussehen:

```
xp103314:ht=ethernet:ha=00008159ab1234:ip=164.25.242.3:bf=XP103314.cfg
```

Sie enthält der Reihe nach:

- den Namen des zu bootenden Geräts,
- den Typ des Mediums, über das es am Netz angeschlossen ist (`ht`),
- seine MAC-Adresse (`ha`),
- seine IP-Adresse (`ip`) und
- den Namen der Bootdatei (`bf`).

Die erforderlichen Konfigurationsschritte sind:

1. Vergabe einer IP-Adresse

   Einordnung: obligatorische, generische Aktion.

   Diese Aktion wird teilweise durch Konfiguration des BOOTP-Servers, teilweise durch Eintrag von Werten in der Konfigurationsdatei durchgeführt.

   Benötigte Daten:

   - *hubname* Name des Hubs für BOOTP,
   - *hubmac* MAC-Adresse des Hubs,
   - *hubip* IP-Adresse des Hubs,

- *netmask* Subnet Mask für das zugehörige Netzsegment,
- *router* IP-Adresse des Routers in Richtung Netzmanagement-Station,

Aktion:

Eintrag einer Zeile in `bootptab` mit folgendem Inhalt:
hubname`:ht=ethernet:ha=`*hubmac*`:ip=`*hubip*`:bf=`*hubboot*

router in Zeile `default-router` in der Konfigurationsdatei eintragen.

netmask in Zeile `netmask` in der Konfigurationsdatei eintragen.

2. Laden neuer Firmware in den Hub

   Einordnung: optionale, generische Aktion

   Diese Aktion wird ebenfalls durch Konfiguration des BOOTP-Servers durchgeführt.

   Benötigte Daten:

   - *hubname* Name des Hubs für BOOTP,
   - *hubboot* Name der Boot-Datei für den Hub

   Aktion:

   Eintrag der Bootdatei in der durch die vorangegangene Aktion erzeugte Zeile in `bootptab`:
   hubname`:ht=ethernet:ha=`*hubmac*`:ip=`*hubip*`:bf=`*hubboot*

In der Telekom-Dokumentation werden die beiden hier genannten Aktionen als Einheit betrachtet, weil sie die gleiche Zeile der Konfigurationsdatei bearbeiten. Um eine Einordnung in ein generisches Verfahren vornehmen zu können, müssen sie jedoch, wie hier durchgeführt, getrennt werden. An diesem Beispiel sieht man, daß die Entwicklung einer Generik in Einzelfällen zu weniger effizienten Abläufen führt. Das ist jedoch ein geringer Preis für das erreichte Ziel, verschiedene Hubs einheitlich konfigurieren zu können.

3.4 allgemeiner Repeater nach RFC 1516

Abschließend soll noch betrachtet werden, inwieweit der Konfigurationsvorgang für einen unbekannten Hub vorgezeichnet werden kann, unter der Annahme, daß die Standard-MIB für Ethernet-Repeater [RFC1516] unterstützt wird.

1. Vergabe einer IP-Adresse

   Einordnung: obligatorische, generische Aktion.

   Diese Aktion kann (wie schon oben erwähnt) nicht durch SNMP unterstützt werden, da die Verwendung von SNMP erst nach diesem Schritt möglich wird. Es wird in den allermeisten Fällen möglich sein, die Konfiguration über die Konsole vorzunehmen, auch wird von einer Reihe von Geräten RARP oder BOOTP unterstützt. Ein einheitlicher Standard ist jedoch nicht vorhanden.

2. nichtverwendete Ports deaktivieren

   Einordnung: obligatorische, generische Aktion

   Die verwendeten Ports müssen aktiviert, die übrigen deaktiviert werden. Für diesen Konfigurationsschritt bestehen zwei Ansätze: Die Verwaltung von Ports ist sowohl im Teil `interfaces` der MIB-II vorgesehen als auch in der Repeater-MIB.

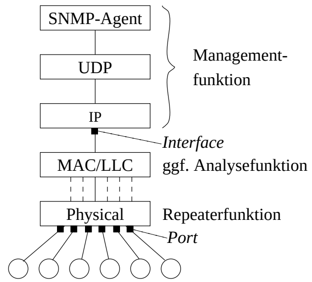

Abbildung 3.2: Ports und Interfaces nach RFC 1516

Von der Intention der MIB-II her scheint sie nicht so sehr für Repeater ausgelegt zu sein, sondern wurde sicherlich ursprünglich für Bridges und Router konzipiert. Diese Annahme wird auch von der Repeater-MIB mit eigenen Portstatus-Variablen bestärkt. Dort wird ausdrücklich betont, daß Repeater-Ports keine *Interfaces* im Sinne der MIB-II sind; ein Repeater enthält nur ein *Interface*, nämlich zum SNMP-Agenten (Bild 3.2). Das Beispiel des HP Hubs, der oben analysiert wurde, zeigt jedoch, daß manche Hersteller, deren Firmware möglicherweise vor Erscheinen des RFC 1516 entwickelt wurde, in ihrer MIB dennoch Ports mit Interfaces gleichsetzen. Deshalb werden im folgenden beide Möglichkeiten dargestellt.

In der MIB-II werden die Ports linear durchnumeriert (n); die modernere Repeater-MIB berücksichtigt den modularen Aufbau vieler moderner Hubs und verwendet entsprechend ein zweidimensionales Adressierungsschema (g,p = Gruppe,Port) für die Schnittstellen.

Der Status (*status*) ist übereinstimmend als 1 für enabled bzw. up und 2 für disabled bzw. down definiert.

SNMP (MIB-II):
in `.iso.org.dod.internet.mgmt.mib-2`:
`interfaces.ifTable.ifEntry.ifAdminStatus.`n = *status*

SNMP (RFC 1516):
in `.iso.org.dod.internet.mgmt.mib-2.snmpDot3RptrMgt.rptrPortInfo`:
`rptrPortTable.rptrPortEntry.rptrPortAdminStatus.`$g.p$ = *status*

Für die übrigen Konfigurationsparameter wie beispielsweise redundante Links steht aufgrund der durch die IETF vorgegebenen Politik der nachhinkenden Standardisierung (vgl. Einleitung und [RFC1213]) keine Standard-MIB zur Verfügung.

Daraus ergibt sich ein dürftiges Bild für die Standardisierung des Repeater-Managements durch die IETF. Es ist also nicht sehr verwunderlich, daß Hersteller wie Fibermux zugunsten einer in sich schlüssigen MIB auf die standardkonforme Implementierung dieses einzelnen Parameters verzichten. Es bleibt anzumerken, daß der Standardisierungsgrad für Konfigurationsparameter bei den anderen Koppelelementetypen Bridge und Router deutlich höher ist [RFC1213, RFC1286].

Kapitel 4

Generische Konfigurationsaktionen

In diesem Abschnitt werden die aus den vorhergehenden Betrachtungen entwickelten generischen Aktionen zusammengefaßt. Dabei wird eine informelle Notation verwendet, die sich an dem in Abschnitt 2.3 dargestellten Schema orientiert. Hier werden nur die Übergabeparameter angegeben, da eine generische Aktion zur Durchführung eines Konfigurationsschrittes die dem vorliegenden Gerät entsprechende spezifische Aktion verwendet. Die dabei verwendeten Datentypen werden in Abschnitt 6.3.1 definiert.

In manchen Fällen wurden generische Aktionen identifiziert, obwohl die erforderlichen Parameter nicht für alle betroffenen Geräte übereinstimmen. Wie schon bei den Aktionen selbst, werden daher obligatorische (immer zutreffende) und optionale (nur bei manchen Geräten, die diese Aktion unterstützen, erforderliche) Parameter eingeführt.

4.1 Koppelelement, schichtübergreifend

Wie in Abschnitt 1.2 dargelegt wurde, ist es wünschenswert, über die Verallgemeinerung innerhalb einer Klasse von Koppelelementen hinaus generische Aktionen zu formulieren, die bei allen Koppelelementen unabhängig davon, auf welcher OSI-Schicht sie arbeiten, Anwendung finden.

Obwohl in dieser Arbeit nur Repeater untersucht wurden, läßt sich aufgrund der Natur einiger Aktionen absehen, daß sie bei jedem Koppelelement auftreten werden. Solche Aktionen werden im folgenden beschrieben.

4.1.1 generische, obligatorische Aktionen

- Vergabe einer IP-Adresse

  Um ein Koppelelement (in der Regel mittels SNMP) managen zu können, muß es durch Zuweisung einer IP-Adresse adressierbar gemacht werden. Bei Repeatern und Brücken ist dies unzweifelhaft; bei Routern dagegen könnte der Einwand erhoben werden, daß jede Schnittstelle ihre eigene IP-Adresse besitzt und insofern deren Vergabe eine andersartige Aktion darstellt. Hier handelt es sich jedoch um die IP-Adresse des Management-Agenten im Koppelelement, und diese Adresse wird auch bei Routern eindeutig festgelegt.

  Parameter:

  - OwnAddress: IpAddress;
  - Netmask: IpAddress;
  - Gateway: IpAddress;

  optional:

  - BackupGateway: IpAddress;
  - TTL: INTEGER;
  - SnmpTrapHost: IpAddress;

- Aktivierung bzw. Deaktivierung eines Ports

  Jedes Koppelelement verfügt über Schnittstellen, die zum Anschluß der zu verbindenden Teilnetze dienen. In der Regel besitzt ein Gerät eine Reihe solcher Schnittstellen, die im konkreten Einsatzfall nicht alle belegt sein müssen. Daraus ergibt sich die Notwendigkeit, nichtverwendete Schnittstellen abzuschalten. Auch wenn die IETF zwischen *ports* bei Repeatern und *interfaces* bei Brücken und Routern unterscheidet (vgl. Bild 3.2), erscheint es daher gerechtfertigt, *eine* generische Aktion für diese Fälle zu definieren.

  Parameter:

  - Port: PortNumber;
  - TargetState: BOOLEAN; (aktiv oder inaktiv)

4.1.2 generische, optionale Aktionen

Wie in Abschnitt 2.2 ausgeführt, bedeutet der Begriff optional nicht, daß für ein gegebenes Gerät diese Aktion verzichtbar ist, sondern nur, daß die Aktion nicht von allen Geräten unterstützt wird. Wenn beispielsweise ein Koppelelement nur

über flüchtiges RAM zur Speicherung des Betriebsprogramms verfügt, dann muß das Laden von Firmware konfiguriert werden, um das Gerät in Betrieb nehmen zu können.

- Aktivierung bzw. Deaktivierung eines Moduls

  Modular aufgebaute Geräte finden sich nicht nur im Bereich der Sternkoppler, sondern auch bei Brücken und Packet-Switches sowie insbesondere bei Routern (vgl. [HE94]). Sie ein- bzw. auszuschalten ist daher ein schichtübergreifender Vorgang.

  Parameter:

  - Module: ModuleNumber;
  - TargetState: BOOLEAN; (aktiv oder inaktiv)

- Laden von Firmware

  Viele moderne Koppelelemente verfügen über die Möglichkeit, im Rahmen des Managements neue Versionen ihres Betriebsprogramms über das Netz einzuspielen. Bei den in dieser Arbeit betrachteten Hubs ist das nicht unbedingt die Regel, da sie eine vergleichsweise einfache Aufgabe zu erfüllen haben. Bei Brücken und insbesondere Routern dagegen ist es üblich, durch Softwareanpassungen beispielsweise die Filtermöglichkeiten zu verbessern oder einem Router weitere Netzprotokolle beizubringen. Aus diesem Grunde wird auch das Laden von Firmware als schichtübergreifende Aktion angesehen.

  Parameter:

  - Bootserver: String;
  - Bootfile: FileName;

  optional:

  - BootPort: INTEGER;
  - DeviceIP: IpAddress;
  - DeviceMac: EtherAddress;

4.2 Repeater

4.2.1 generische, obligatorische Aktionen

Da sich die beiden obligatorischen Aktionen „Vergabe der IP-Adresse“ und „(De-)Aktivierung von Ports“ als schichtübergreifend erwiesen haben, gibt es keine obligatorischen Aktionen speziell für Repeater.

4.2.2 generische, optionale Aktionen

Die folgenden Aktionen werden aufgrund der vorangegangenen Untersuchung als generische, optionale Aktionen für Repeater angesehen. Ein Repeater muß diese Aktionen nicht unterstützen, es besteht jedoch Grund zu der Annahme, daß sie von mehreren Geräten unterstützt werden, so daß man sie in die Generik einbeziehen sollte.

- Definition einer Backup-Link

  Um die Ausfallsicherheit eines Netzes zu erhöhen, ist es wünschenswert, wichtige Leitungen durch redundante Verbindungen gegen Leitungsbruch abzusichern.

  Es ist jedoch nicht möglich, einfach mehrere Leitungen zwischen zwei Repeatern zu legen, da es aufgrund der Broadcast-Eigenschaft dieses Koppelelements zu ständigen Kollisionen kommen würde. Daher bieten die meisten modernen Repeater die Möglichkeit, redundante Verbindungen zu verwalten. Eine der Verbindungen wird zunächst aktiviert, aufgrund eines bestimmten Verfahrens wird jedoch ein Ausfall dieser Verbindung erkannt und auf einen anderen Port umgeschaltet.

  Eine Einordnung als schichtübergreifende Aktion kommt nicht in Frage, da Brücken die Auswahl zwischen redundanten Verbindungen, die auch mehr als zwei Geräte einbeziehen können, dynamisch durch den Spanning-Tree-Algorithmus bestimmen. Bei Routern stellt sich das Problem nicht, da sie in der Lage sind, mehrere redundante Verbindungen gleichzeitig zu benutzen und die Last zwischen ihnen zu verteilen.

  Leider ist das verwendete Verfahren nicht standardisiert (vgl. 3.1 Ziffer 4 und 3.2 Ziffer 4), so daß sich die Parameter für diese Aktion zwischen verschiedenen Geräten unterscheiden.

  Parameter:

  - PrimaryPort: PortNumber;

  - BackupPort: PortNumber;

  optional:

  - ErrorThreshold: Percentage;
  - ErrorCount: INTEGER;
  - RemoteAddress: EtherAddress;
  - TestInterval: TimeSpan;

4.2.3 herstellerspezifische Aktionen

Die folgenden Aktionen, die nur bei einem Gerät gefunden wurden, lassen kein Potential für eine Einbeziehung in die Generik erkennen.

- HP: Aktivierung von ThinWatch

  Diese Funktion erlaubt es HP, einen Hub mit 10Base2-Anschluß als Plug-and-Play-Gerät auszuliefern. Bei ordentlicher Konfiguration sollte die Überwachung dieses Ports mit dessen Aktivierung einhergehen.

  Es wäre möglich, diese Aktion in die HP-spezifische Version der Funktion „(De-)Aktivierung eines Ports“ miteinzubeziehen, so daß sie dem Verwalter verborgen bleibt.

  Da der HP Optical Hub über nur einen BNC-Anschluß verfügt, entfällt die Portnummer als Parameter.

  Parameter:

  - TargetStatus: BOOLEAN; (aktiv oder inaktiv)

Teil II

Generische Verfahren

Kapitel 5

Das Rahmenbetriebskonzept

Diese Arbeit soll die Grundlage dafür liefern, daß die Konfigurationsvorgänge für verschiedene Koppelelemente durch ein integriertes Werkzeug vereinheitlicht werden können. Dazu muß eine generische, also herstellerunabhängige, Beschreibung für die Konfigurationsvorgänge erarbeitet werden. Zur Durchführung dieser Konfigurationsvorgänge muß dann eine Abbildung auf die herstellerspezifische Vorgehensweise erfolgen.

Das im nächsten Kapitel entwickelte Modell generischer Verfahren orientiert sich am Rahmenbetriebskonzept, daher wird dieses zunächst vorgestellt. Anschließend werden die notwendigen Anpassungen diskutiert, um generische Verfahren modellieren zu können.

5.1 Überblick

Die Forschungsgruppe „Münchner Netzmanagement Team“ am Lehrstuhl von Prof. Hegering beschäftigt sich schon seit geraumer Zeit damit, am Beispiel des Netzmanagements Arbeitsabläufe in Betrieben zusammen mit den erforderlichen Zuständigkeiten zu sogenannten Betriebskonzepten zu formalisieren. Aus dieser Arbeit hat sich das Rahmenbetriebskonzept entwickelt, das einen Beschreibungsrahmen für solche Betriebskonzepte zur Verfügung stellt.

Das Rahmenbetriebskonzept ist bereits an anderer Stelle [HAW95, AEH+94] vorgestellt worden. Mit Hilfe dieses Gerüsts ist es möglich, betriebliche Abläufe in einem Unternehmen zu erfassen und zu analysieren. Es ermöglicht eine genauere Zuordnung der entstehenden Kosten in Abteilungen eines Betriebes zu den erbrachten Diensten und dient damit als Basis für Outsourcing-Entscheidungen, d. h. ob bestimmte Dienste vom Unternehmen selbst erbracht oder von einem externen Dienstleister eingekauft werden sollen. Gleichzeitig können an verschiedenen Stellen erbrachte Leistungen koordiniert werden, um Synergieeffekte zu erzielen. Das Rahmenbetriebskonzept kann auch eingesetzt werden, um die Betriebsabläufe zu formalisieren, wie das für die Zertifizierung nach ISO 9000ff erforderlich ist.

Es gliedert sich in die Schichten:

- Dienstebene
- Aufgabenebene und
- Verfahren- und Werkzeugebene.

Bild 5.1 zeigt das Objekt- und Beziehungsmodell des Rahmenbetriebskonzeptes [HAW95].

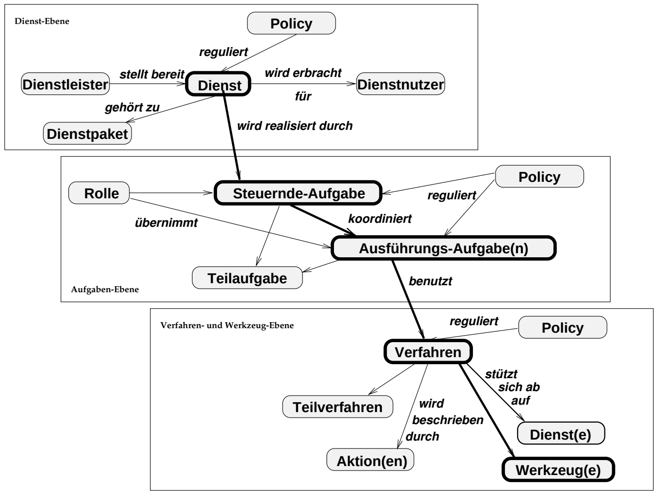

Abbildung 5.1: Objekt- und Beziehungsmodell des Rahmenbetriebskonzeptes

In vorangehenden Diplomarbeiten [Fis95b, Ber95] wurde es verfeinert und weiterentwickelt. [Ber95] hat die Verfahren- und Werkzeugebene strukturiert:

Ein Verfahren besteht aus Teilverfahren, die durch eine Ablaufsteuerung mit Verzweigungen miteinander verbunden sind. Diese Teilverfahren ihrerseits sind Folgen von Aktionen, die stets sequentiell abgearbeitet werden. Je nach Struktur eines Verfahrens können auch mehrere Teilverfahren parallel ausgeführt werden. [Ber95] stellt hierfür eine visuelle Ablaufbeschreibung vor. Die Teilverfahren sind

durch ihre Ergebnisse aneinander gekoppelt; dabei schließt der Begriff Ergebnis die dem Teilverfahren ursprünglich übergebenen Parameter ein, man könnte also von einem kumulierten Ergebnis sprechen. Ferner ist die Behandlung von Ausnahmesituationen durch Fehlerverfahren vorgesehen. Einen Überblick gibt Bild 5.2.

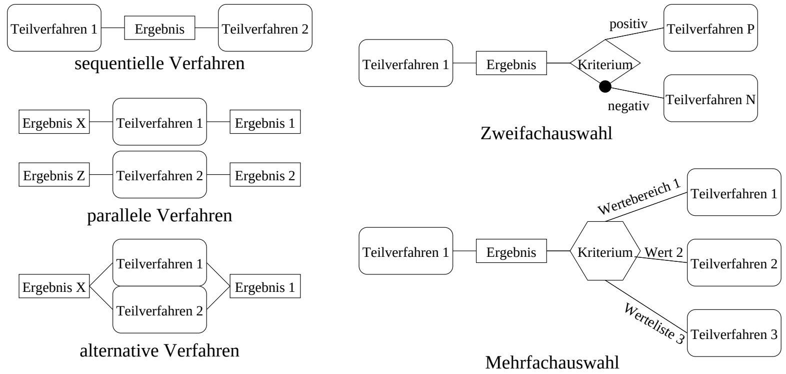

Abbildung 5.2: Varianten für Verfahren nach [Ber95]

5.2 Anpassungen

Das Rahmenbetriebskonzept sieht nicht die Darstellung generischer Aktionen und Verfahren vor. Vielmehr muß für jede Konstellation von Komponente und benutztem Werkzeug ein separates Verfahren definiert werden [vdH94, Wei94].

Angewandt auf die im ersten Teil betrachteten Vorgänge könnten beispielsweise folgende Verfahren dokumentiert werden:

- Konfiguration eines HP Optical Hub über HP OpenView
- Konfiguration eines HP Optical Hub über die Konsole
- Konfiguration eines Fibermux Crossbow über HP OpenView
- Konfiguration eines Fibermux Crossbow über LightWatch
- Konfiguration eines Synoptics Workgroup Hub mittels Konfigurationsdatei

Es ist einleuchtend, daß diese Verfahren Parallelen in ihrer Struktur aufweisen müssen, da die Zielsetzung ja die gleiche ist, nur die eingesetzten Mittel unterscheiden sich. Zentrales Anliegen dieser Arbeit ist es daher, durch Definition von generischen Aktionen Beschreibungsmöglichkeiten für generische Verfahren zur Verfügung zu stellen, so daß nur ein Verfahren definiert werden muß:

- Konfiguration eines Sternkopplers

Um jedoch solche generischen Verfahren beschreiben zu können, ist es erforderlich, generische Teilverfahren und Aktionen zu schaffen, auf die sie sich stützen können. Im nächsten Abschnitt wird die Verfahren- und Werkzeugebene entsprechend erweitert.

Zielsetzung der hier entwickelten Beschreibung von Konfigurationsvorgängen ist sowohl die Dokumentation manuell durchgeführter Arbeitsschritte (Modellierung des Ist-Zustands einer gegebenen Management-Situation) als auch die automatische Abarbeitung einer solchen Beschreibung durch einen Interpreter (angestrebter Zielzustand). Auch dies schafft neue Anforderungen an die Verfahren- und Werkzeugebene.

Bei der bisherigen Anwendung des Rahmenbetriebskonzepts wurden nur manuelle Aktionen beschrieben, die sich zum Zugriff auf einen Management-Datenbestand immer eines Werkzeuges bedienen müssen (und sei es `snmpset`). [Seg95] definiert eine Funktions-/Informationsebene, die den unteren Teil der Verfahren- und Werkzeugebene strukturiert, wie in Bild 5.3 dargestellt. Für eine Aktion ergeben sich also gewissermaßen zwei „Standbeine“, nämlich Werkzeug und Information.

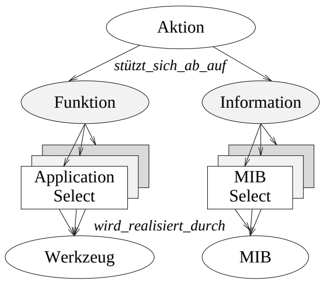

Abbildung 5.3: Funktions-/Informationsebene nach [Seg95]

In dieser Arbeit sollen dagegen auch automatisch durchgeführte Konfigurationsschritte betrachtet werden. Der in Kapitel 8 entworfene Interpreter für generische Verfahren wird die benötigte Funktionalität bestimmter Aktionen selbst zur Verfügung stellen, so daß die Abstützung auf eine Funktion eines externen Werkzeugs entfällt.

Ob diese Variante umgesetzt werden kann oder sogar muß, hängt davon ab, auf welche Weise die Konfigurationsdaten zugänglich sind.

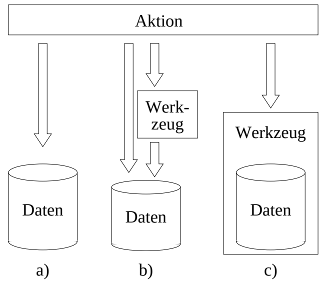

Abbildung 5.4: Varianten des Zugriffs auf Managementdaten

Wie in Bild 5.4 dargestellt, gibt es dabei die folgenden drei Möglichkeiten:

a) Der Zugriff auf die Daten erfolgt direkt, z. B. via SNMP. Hier wird der Interpreter selbst eine Funktion zur Verfügung stellen, da der Einsatz einfacher externer Programme wie `snmpset` weniger effizient wäre.

b) Der Zugriff erfolgt durch ein Werkzeug, die Daten stehen jedoch über ein genormtes Managementprotokoll auch zur direkten Manipulation zur Verfügung. Hier kann die Implementierung entscheiden, wie der Zugriff erfolgen soll.

c) Die Daten können nur durch ein externes Werkzeug manipuliert werden, da das Werkzeug ein proprietäres Protokoll für den Zugriff benutzt. In solchen Fällen ist die Anbindung externer Werkzeuge unausweichlich.

Kapitel 6

Entwicklung einer Generik

Bereits in Abschnitt 1.1 wurde die Ebene der Aktionen als Schnittstelle zwischen generischer und spezieller Beschreibung identifiziert.

Da der Begriff der Aktion nun im Rahmenbetriebskonzept verankert wurde, sollen die sich daraus ergebenden Änderungen für die Beziehungen im Rahmenbetriebskonzept erläutert werden.

6.1 Erweiterungen des Rahmenbetriebskonzeptes

Die Umsetzung von der generischen Beschreibung auf ein konkretes Gerät findet auf der Ebene der Aktionen statt. Bild 6.1 veranschaulicht die so entstehende Einbindung der hier zu definierenden generischen Aktion in die von Rahmenbetriebskonzept vorgezeichneten Strukturen.

Entsprechend den oben ausgeführten Überlegungen werden die Verfahren und Teilverfahren generisch dargestellt. Erst auf der Ebene der Aktionen, die als Arbeitsschritte ja auch mehrfach in dem gleichen oder verschiedenen Verfahren auftreten können, erfolgt die gerätspezifische Umsetzung. Somit entstehen zwei Ebenen für Aktionen, nämlich eine generische und darunter eine spezifische.

Wie schon in der Auswertung der Untersuchung im ersten Teil deutlich wurde, müssen allgemeine Parameter für die generischen Aktionen definiert werden. Daher erfolgt parallel zu der Angabe generischer Aktionen die Definition eines generischen *Konfigurations-MOs* für einen Koppelelementetyp, auf dem diese generischen Aktionen wirken. Dieses *Managed Object* stellt die Summe derjenigen Attribute aus allen Management-Ressourcen dar, die für die Konfiguration von Bedeutung sind.

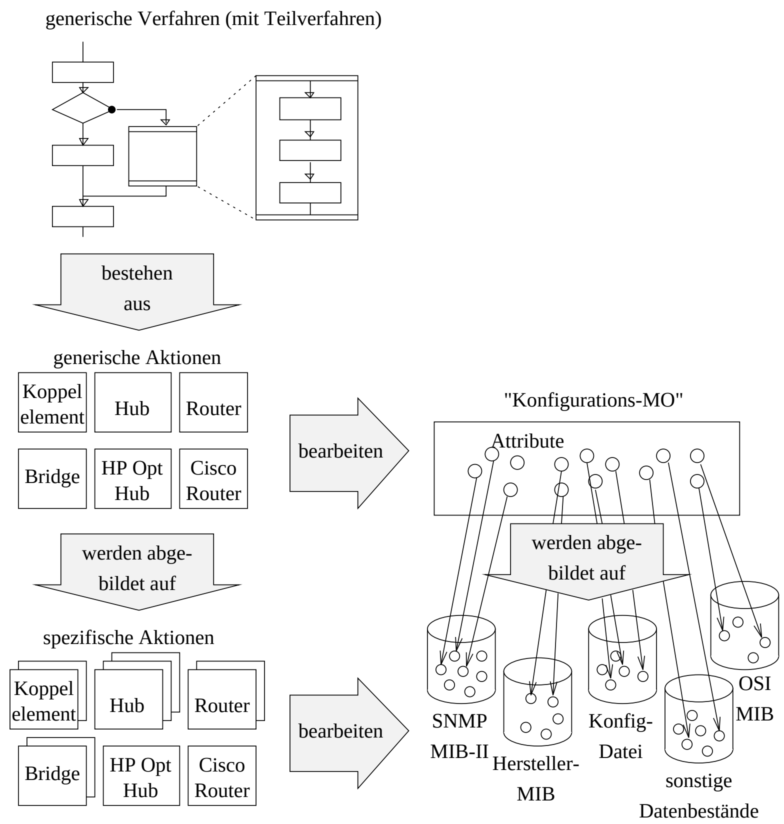

Abbildung 6.1: Einbindung der Generik in das Rahmenbetriebskonzept

Beim Aufruf eines generischen Verfahrens werden die konkreten Angaben über Gerät und Werkzeug als Parameter übergeben. Diese werden an die verwendeten generischen Aktionen weitergeleitet, die sie zur Auswahl der spezifischen Aktion benötigen.

Gleichzeitig mit dem Übergang von generischen zu spezifischen Aktionen erfolgt die Zuordnung tatsächlicher Konfigurationsparameter zu den Attributen des Konfigurations-MO.

Zusammenfassend läßt sich festhalten, daß mit diesem Schema die Voraussetzungen für die Entwicklung eines tief integrierten Managementwerkzeugs geschaffen werden.

6.2 Alternativen für den Beschreibungsrahmen

Als Ansatzpunkt für die Umsetzung der Generik kämen grundsätzlich zwei vorhandene Beschreibungsrahmen in Frage:

- die Informationsstruktur des OSI Managements und
- das MIB-Variablenkonzept des Internet-Managements (SNMP).

Der Vorteil der OSI-Management-Architektur liegt in der Unterstützung einer Abstraktionshierarchie mit Vererbung. Ferner wird die Abstraktion von *Actions* unterstützt, die neben der Veränderung einzelner Variablen auch komplexe Operationen auf mehreren Variablen ermöglicht.

Der Vorteil der Internet-Management-Architektur liegt in der Nähe zum Management real existierender Komponenten. Abstraktionsschritte auf Variablen könnten durch Definition einer „Super-MIB" erfolgen. Zur Umsetzung auf die einzelnen Geräte kann man Untermengen bilden sowie bei Bedarf Variablen auf herstellerspezifische MIBs abbilden.

Die Internet-Management-Architektur würde sich also für eine losgelöste Beschreibung der Konfigurations-MOs eignen, und auch das nur auf niedrigster Ebene, da eine Aggregation von Attributen zu Gruppen nicht möglich ist. Dies ist jedoch für die Beschreibung des Zugriffs von Element-Managern sowie für die Behandlung von Gerätegruppen als abstrakte Geräte erforderlich.

Auch die OSI-Beschreibungen beschränkt sich jedoch nur auf die Deklaration von Variablen als Parameter der im Managementprotokoll vorgesehenen Operationen. Die OSI-Actions sind hier nicht einsetzbar, da sie zusammengesetzte Operationen beschreiben, die vom Agenten des Geräts bereitgestellt werden.

Die Umsetzung einer generischen auf eine spezifische Aktion ist jedoch eine prozedurale Einheit, die auf Seiten der Managementanwendung stattfindet.

Es scheint sich anzubieten, die erforderlichen prozeduralen Angaben im Sinne der im OSI-Management definierten *Systems Management Functions (SMFs)* [ISO10164] zu beschreiben. Jedoch ist der dort verwendete Beschreibungsrahmen für die hier zu spezifizierenden Verfahren wenig geeignet. Zunächst werden in den SMFs Abläufe nur umgangssprachlich definiert. Der Schwerpunkt dieser Standards liegt in der Definition der Parameter dieser Funktionen, die in ASN.1 [ISO8824] angegeben werden. Diese Arbeit muß aber einen formalen Beschreibungsrahmen auch für die Ablaufbeschreibung definieren. Ferner sind die dort

beschriebenen Funktionen zum Aufruf über das Netz im Rahmen von CMIP [ISO9596] in die Elemente Request, Indication, Response und Confirmation aufgeteilt. Hier geht es dagegen um die Beschreibung lokaler Funktionen innerhalb des zu entwickelnden Management-Werkzeugs. Außerdem handelt es sich bei den Systems Management Functions in Wirklichkeit um Prozedurbibliotheken; in dieser Arbeit sollen aber einzelne Prozeduren definiert werden, zunächst ohne sie in Bibliotheken einzuteilen, da es sich entsprechend dem Stand der Untersuchung um eine dynamische Menge von Prozeduren handelt.

Diese Überlegungen zeigen, daß beide ins Auge gefaßten Beschreibungsrahmen nicht für die Beschreibung der generischen Verfahren und der Umsetzung von generischen auf spezifische Aktionen geeignet sind.

Daher wird in dieser Arbeit willkürlich eine eigene Syntax für die geforderte Beschreibung definiert. Sie ist möglichst einfach gehalten, damit die so abgefaßten Beschreibungen gegebenenfalls auch automatisch in die für ein Implementierungswerkzeug verwendete Form übersetzt werden können.

Zunächst soll nun zusammenfassend dargestellt werden, welche Elemente die Beschreibung der generischen Verfahren und Aktionen sowie der spezifischen Aktionen haben muß. Außerdem wird die zugehörige Syntax in der Beschreibungssprache definiert, um die erforderlichen Objekte eindeutig darstellen zu können.

6.3 Elemente der Beschreibungssprache

Die Beschreibung umfaßt vier Objekttypen:

- generische Verfahren,
- generische Aktionen,
- spezifische Aktionen,
- Basisoperationen.

Die ersten drei wurden bereits eingeführt und erläutert. Aus praktischen Erwägungen wird hier die Kategorie der Basisoperationen hinzugefügt.

Es wird nämlich eine große und mit der Einbeziehung weiterer Komponenten wachsende Menge von spezifischen Aktionen geben. Daher ist es nicht erstrebenswert, alle diese Aktionen in Abhängigkeit des zu entwickelnden generischen Konfigurationswerkzeugs implementieren zu müssen. Vielmehr sollte von dem Werkzeug nur eine kleine Zahl von stark parametrisierten Primitiven verlangt werden, die hier Basisoperationen genannt werden.

Die Beziehung des hier definierten Objektbestands zur Architektur eines solchen Werkzeugs ist in Bild 6.2 skizziert. Fast alle Teile der Beschreibung können so werkzeugunabhängig gehalten werden.

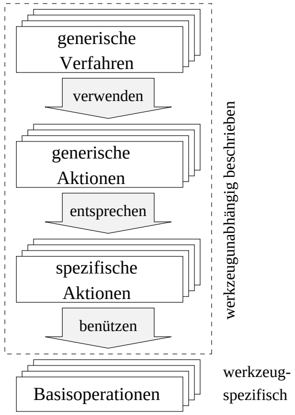

Abbildung 6.2: Beziehung zwischen Beschreibungssprache und Werkzeug

6.3.1 allgemeine Syntax

In dieser Arbeit wird eine Syntax für die Beschreibung der für die Generik erforderlichen Objekte benötigt, die knapp und leicht verständlich, gleichzeitig jedoch hinreichend formal ist, um auch automatisch in die für ein Werkzeug benötigte Form umgesetzt zu werden. Diese Syntax wird hier definiert, um in Kapitel 7 mit den im ersten Teil gewonnenen Informationen gefüllt zu werden.

Die Beschreibung eines Objekts in der hier definierten Beschreibungssprache hat die folgende Form:

Objekttyp Objektname {
`const`
  variable: *Typ* = *Wert*
  ...
`input`
  variable: *Typ*
  ...
  variable: *Typ* `optional for` *Selektor*, *Selektor*...
  ...
`output`

  variable: *Typ*
  ...
  variable: *Typ* `optional for` *Selektor*, *Selektor*...
  ...
`semantics`
  "*textuelle Beschreibung des Ergebnisses des Objekts*"
`body`
  Ablaufbeschreibung wie unten definiert
}

Nicht alle Objekte enthalten alle hier vorgestellten Felder. *Selektor* beschreibt ein Paar (*Gerät*, *Werkzeug*). Trifft ein Parameter für ein bestimmtes Gerät unabhängig vom verwendeten Werkzeug zu, dann kann auch (*Gerät*,`*`) geschrieben werden.

Ferner sind Zeilenkommentare erlaubt, die mit `#` eingeleitet werden.

Da es sich bei den hier betrachteten Vorgängen um einfache Schritte der Konfiguration handelt, werden keine Rückgabeparameter der Aktionen und Verfahren benötigt. Der Abschnitts `output` wird daher im folgenden nicht verwendet. Für ihn gilt alles, was weiter unten für `input` gesagt wird, entsprechend.

Variablentypen

Wie im obigen Schema gezeigt, wird für die verwendeten Parameter jeweils ein Typ angegeben. Dies dient der Klarheit der Definition. Jedoch handelt es sich bei den hier verwendeten Typenangaben um abstrakte Typen.

So werden beispielsweise die Typen `IpAddr`, `EtherAddr`, `String`, `DisplayString`, `TimeSpan`, `INTEGER`, `BOOLEAN`, `PortNumber`, `ModuleNumber`, `FileName` verwendet. Die Umsetzung in konkrete Typen ist Sache der Implementierung und hängt von der verwendeten Programmiersprache ab. Es ist beispielsweise zu berücksichtigen, daß `PortNumber`, wie unter Abschnitt 3.2 gezeigt, ein Tupel zweier Werte, aber auch eine einzelne Zahl sein kann. Ob diese Unterscheidung abhängig von zu konfigurierenden Gerät schon bei der Eingabe der Parameter eines generischen Verfahrens gemacht wird, oder eine Konsistenzprüfung erst bei der Auswertung durch die spezielle Aktion erfolgt, hängt ebenfalls von der Implementierung ab. Ebenfalls ist denkbar, daß bei der Eingabe einer Variablen von Typ `FileName` eine Liste existierender Dateien zur Auswahl angezeigt wird.

Um komplexe Typen zu erhalten, werden die beiden Typkonstruktoren

`ARRAY [`*idxtyp*`]` *typ*
  für ein durch `idxtyp` indiziertes eindimensionales Feld und

`RECORD` {*element*:*typ*...}
: für eine Struktur

definiert. Der Zugriff erfolgt entsprechend mit *feld*`[`*index*`]` und *struktur*`.`*element*.

Eigenschaften einzelner Geräte

Unter dem hier gewählten bottom-up Ansatz ist es unausweichlich, daß auf bestimmte Eigenschaften und Besonderheiten einzelner Geräte eingegangen wird. Im Sinne der Akzeptanz des zu entwickelnden Werkzeugs ist dies ja auch durchaus sinnvoll (vgl. 6.3.3). Folgende Eigenschaften eines Sternkopplers müssen daher festgelegt werden (und stellen Elemente in der Struktur `hubtype` dar):

`id`
: Der Bezeichner dient zur eindeutigen Angabe des Geräts in Selektoren.

`firstport, lastport`
: Bei der Erstkonfiguration werden Ports aktiviert bzw. deaktiviert. Diese Grenzen geben an, für welche Werte eine entsprechende Schleife durchlaufen werden muß.

`ismodular`
: Dieser Parameter gibt an, ob das Gerät aus Modulen aufgebaut ist. Daraus ergibt sich einerseits, ob Module aktiviert oder deaktiviert werden müssen, ferner bestimmt diese Information den Datentyp von `PortNumber`.

`firstmodule, lastmodule`
: Bei modularen Geräten gilt das für `firstport, lastport` Gesagte auch für ihre Module.

Die hier genannten Parameter sind vom Typ des Koppelelementes unabhängig, gelten also auch für andere Geräteklassen.

Elegant wäre ferner, wenn auch die zutreffenden optionalen Parameter in dieser Struktur verzeichnet wären. Das setzt jedoch voraus, daß eine Variable eine Liste von Variablendeklarationen enthalten kann. Aufgrund der damit verbundenen Komplexität wurde daher die Deklaration mit Selektoren gewählt.

6.3.2 generische Verfahren

Hier kann die Verfahrensbeschreibung von [Ber95] aufgegriffen werden (vgl. Bild 5.2). Jedoch tritt anstelle der grafischen Beschreibung eine textuelle. Ferner wird zur Vereinfachung auch ermöglicht, Aktionen direkt in Verfahren einzubinden, um triviale Teilverfahren mit nur einer Aktion zu vermeiden.

Alle von den beteiligten Aktionen benötigten Parameter sind bei der Verfahrensbeschreibung mit anzugeben, so daß ein Interpreter vom Benutzer diese Angaben auch gemeinsam, beispielsweise in einem Formular, erfragen kann. So werden Verzögerung oder Abbruch bei der Ausführung eines Verfahrens vermieden, die durch das Fehlen eines Konfigurationsparameters entstehen könnten.

Zusätzlich zu den hier angegebenen Parametern wird einem generischen Verfahren immer auch das zu konfigurierende Gerät und das zu verwendende Werkzeug übergeben. Diese Parameter wertet das Verfahren selbst nicht aus; in den generischen Aktionen dienen sie dazu, die zutreffende spezifische Aktion auszuwählen. Auch können optionale Parameter von diesen Angaben abhängig gemacht werden.

Wie die Daten über das zu verwendende Gerät und Werkzeug gewonnen werden, ist Sache der Implementierung eines Werkzeugs. Auf die Einbindung eines neuen Gerätes in vorhandene Managementsoftware, beispielsweise in der Beschreibung der Netztopologie oder einer Bestandsdatenbank, geht diese Arbeit nicht ein. Dennoch stellt diese Tätigkeit einen wichtigen Teil im Rahmen der Erstkonfiguration dar. Bei der Implementierung ist es natürlich wünschenswert, diese in den Ablauf der Konfiguration zu integrieren; schon aus diesem Grunde wird die hier benötigte Information ermittelt werden müssen.

Es stellt sich die Frage, wie optionale generische Aktionen berücksichtigt werden sollen, da es ja vom Gerät abhängt, ob sie verfügbar sind. Zur Vereinfachung der Verfahrensdarstellung wird hier ein Kunstgriff angewandt: Es wird so getan, als ob alle Aktionen verfügbar seien; wenn eine generische Aktion jedoch für ein gegebenes Gerät über keine spezifische Entsprechung verfügt, wird sie einfach übergangen.

Aufgrund des in Abschnitt 6.1 beschriebenen Modells ist es nicht möglich, spezifische Aktionen direkt in generische Verfahren einzubinden. Dies ist im allgemeinen ja auch nicht erwünscht, da das Verfahren seine allgemeine Anwendbarkeit verlieren würde. In dieser Arbeit wurde bei der Definition der Aktionen darauf geachtet, daß herstellerspezifische Aktionen, die nicht in die Generik einbezogen wurden, keinen hohen Stellenwert für die Konfiguration haben. Dies wird sich jedoch nicht immer durchhalten lassen. Damit also auch solche Aktionen für Verfahrensbeschreibungen zugänglich gemacht werden, muß eine generische Aktion, die nur für dieses Gerät eine spezifische Entsprechung hat, definiert werden.

Ein generisches Verfahren wird mit `genproc` bezeichnet und enthält die Angaben `input`, `semantics` und `body`. Ein Teilverfahren erhält stattdessen die Bezeichnung `subproc` und besteht definitionsgemäß nur aus einer Sequenz generischer Aktionen.

Im Body werden die folgenden Konstrukte definiert:

Aufruf einer Aktion

`action` *Name* (*Parameter*)

Aufruf eines Teilverfahrens
`subproc` *Name* (*Parameter*)

Sequenz
aufeinanderfolgende Zeilen

Schleife
`loop` *Variable* `from` *Startwert* `to` *Endwert* {
Schleifenrumpf
}

Mehrfachauswahl
`case` *Variable*
Wertebereichsliste: *Anweisung*
...
`esac`

Um die Implementierung zu erleichtern, wird die Komplexität der Verfahrensbeschreibung bewußt niedrig gehalten. Auf folgende Konstrukte wird deshalb hier verzichtet:

Zweifachauswahl
kann durch Mehrfachauswahl ausgedrückt werden.

Alternative
Alternative Ausführung in [Ber95] bezieht sich im wesentlichen auf die Wahlmöglichkeit des eingesetzten Werkzeugs. Dies ist im hier vorgestellten Schema durch Parametrisierung berücksichtigt und geht daher nicht in die Verfahrensbeschreibung ein.

Parallelausführung
Parallelisierungsmöglichkeiten ergeben sich aus der Überschneidungsfreiheit von Parametern und müssen daher nicht explizit angegeben werden.

Behandlung von Fehlern (Exceptions)
Die Fehlerbehandlung würde die Komplexität erheblich steigern. Daher wird sie in die Beschreibungssprache – jedenfalls zunächst – nicht einbezogen.

6.3.3 generische Aktionen und Informationen

Generische Aktionen bestehen nur aus Parameterdefinitionen. Implizit enthalten sie eine Mehrfachauswahl abhängig von den Gerät- und Werkzeugparametern, die dem generischen Verfahren mitgegeben worden sind. Der jeweils zutreffende

Prozedurrumpf findet sich in den spezifischen Aktionen, denn er ist abhängig von dem im Einzelfall zu konfigurierenden Gerät.

Die hier angegebenen Parameter ergeben sich aus den Parametern für die zugehörigen speziellen Aktionen. Im Sinne einer Top-Down-Strategie wäre es wünschenswert, aufgrund der Managementanforderung die erforderlichen Parameter zu spezifizieren. Jedoch stimmen diese dann meist nicht mit den von den real verfügbaren Geräten durch ihre Management-Agenten bereitgestellten Attributen überein. In der Folge muß eine Übersetzung der Parameter vorgenommen werden. Auch werden die Fähigkeiten einzelner Geräte typischerweise nicht ausgenützt. Daher wurde in dieser Arbeit eine andere Strategie gewählt, nämlich bottom-up von den verfügbaren Geräten aus zu ermitteln, welche Parameter zur Konfiguration zur Verfügung stehen. So ist sichergestellt, daß die Fähigkeiten der Geräte verfügbar gemacht werden.

Diese Entscheidung führt zu einfacherem Design des in Kapitel 8 entworfenen prototypischen Werkzeugs und erhöht dessen Akzeptanz.

Das Muster für die Definition generischer Aktionen wird daher folgendermaßen festgelegt:

`action` *Objektname* {
`input`
variable: *Typ*
...
variable: *Typ* `optional for` *Selektor*
...
`semantics`
"*textuelle Beschreibung des Ergebnisses der Aktion*"
}

6.3.4 spezifische Aktionen und Informationen

Hier finden sich die erforderlichen Schritte, um eine generische Aktion bei gegebenem Gerät und Werkzeug umzusetzen. Diese Beschreibungen beziehen sich ihrerseits auf eine kleine Zahl von Basisoperationen, die vom Interpreter zur Verfügung gestellt werden.

Die spezifischen Aktionen enthalten keine Angaben über die Semantik, da diese sich aus der generischen Aktion ergibt, die sie verwirklichen. Sie enthalten Angaben über die erforderlichen Parameter (`input`), da es sich hier nur um eine Untermenge der bei der generischen Aktion definierten Parameter handelt.

Um die Nähe zur generischen Aktion auszudrücken, wird die gleiche Notation verwendet, wobei der Name der Aktion um den Selektor erweitert wird. Optionale

Parameter gibt es nicht; entweder ein generisch optionaler Parameter trifft für *Selektor* zu, dann wird er angegeben, oder nicht, dann wird er weggelassen.

Um ggf. notwendige Konversionen von Parametertypen oder -werten vorzunehmen, kann hier ein Abschnitt `const` zur Definition von Konstanten verwendet werden.

Daher wird folgende Syntax festgelegt:

```
action Objektname(Gerät, Werkzeug) {
const
    variable: Typ = Wert
    ...
input
    variable: Typ
    ...
body
    Ablaufbeschreibung
}
```

Die Ablaufbeschreibung entspricht von ihrer Struktur her der für `genproc` definierten. Jedoch ist es natürlich nicht möglich, Teilverfahren oder generische Aktionen aufzurufen. Stattdessen kann man auf Basisoperationen zurückgreifen:

Aufruf einer Basisoperation

`basisop` *Name* (*Parameter*)

6.3.5 Basisoperationen

Die Basisoperationen stellen keinen Teil der hier vorgestellten Beschreibung dar, sondern werden von der Beschreibung als Elemente des Interpreters gefordert. Da sie bei jeder Implementierung neu entwickelt werden müssen, wird versucht, möglichst wenige, einfache Funktionen zu definieren. Sie sind daher sehr allgemein gehalten und stark parametrisiert. So können möglichst viele Informationen innerhalb der Beschreibung abgedeckt werden.

Die Implementierung der Basisoperationen liegt wie gesagt außerhalb dieser Beschreibung, hier werden daher nur die für sie geforderte Funktionalität (Semantik) und ihre Schnittstellen (Parameter) definiert.

```
basisop Objektname {
input
    variable: Typ
    ...
semantics
```



```
    "textuelle Beschreibung des Ergebnisses der Basisoperation"
}
```

Kapitel 7

Anwendung der generischen Beschreibung

In diesem Kapitel wird die oben entwickelte Beschreibung auf die im ersten Teil untersuchten Konfigurationsvorgänge angewandt. Die hier formalisierten Verfahren und Aktionen eignen sich zur Umsetzung in ein Werkzeug, dessen Konzeption in Kapitel 8 vorgestellt wird.

Wie in Abschnitt 6.3.3 erläutert, wurde ein bottom-up Ansatz zur Entwicklung der Generik verfolgt. Folgerichtig werden hier die Objekte von unten nach oben präsentiert.

7.1 Gerätetypen

Abschnitt 6.3.1 fordert die Deklaration der Geräteparameter. Daher werden die entsprechenden Daten aus Kapitel 3 hier zusammengefaßt:

```
(id = "HPFiberOpticHubPlus"
 firstport = 1
 lastport = 10
 ismodular = false
)

(id = "FibermuxCrossbow"
 firstport = (c = 1
              p = 1)
 lastport = (c = 14
             p = 16)
 ismodular = true
```



```
 firstmodule = 1
 lastmodule = 14
)

(id = "SynopticsWorkgroup"
 firstport = (c = 1
              p = 1)
 lastport = (c = 5
             p = 17)
 ismodular = true
 firstmodule = 1
 lastmodule = 5
)
```

7.2 Basisoperationen

Da in dieser Arbeit keine APIs von Managementplattformen behandelt werden, werden keine externen Werkzeuge angesteuert. Die folgenden Basisoperationen umfassen daher nur:

- Anleitung zur manuellen Operation (für die Managementplattform);
- Setzen einer Variable mittels SNMP;
- Dateimanipulation (z. B. für die BOOTP-Konfigurationsdatei `bootptab`).

Sobald API-Schnittstellen in das Werkzeug aufgenommen werden, müssen hier natürlich entsprechende Basisoperationen definiert werden. Um die Werkzeugabhängigkeit gering zu halten, ist auch dabei auf möglichst breite Parametrisierung zu achten, so daß wenige Basisoperationen für alle zugehörigen spezifischen Aktionen ausreichen.

```
basisop snmpset {
input
  ip: IpAddr
  var: MibNode
  value: ANY
semantics
  "weist den SNMP-Agenten ´ip´ an, die MIB-Variable ´var´
   auf ´value´ zu setzen"
}

basisop manual {
```

```
input
  explanation: DisplayString
semantics
  "zeigt ´explanation´ dem Benutzer an und wartet auf
   Bestaetigung"
}

basisop addline2file {
input
  filename: FileName
  contents: String
semantics
  "fuegt an Datei ´filename´ eine Zeile mit Inhalt ´contents´ an"
}

basisop editfile_keyfield {
input
  filename: FileName
  fsep: String
  vsep: String
  lineselect: String
  key: String
  value: String
semantics
  "Sucht in Datei ´filename´ nach einer Zeile, die mit
   ´lineselect´ ´fsep´ beginnt, und ersetzt in
   ´fsep´ ´key´ ´vsep´ <Wert>
   den Wert durch ´value´ oder fuegt dieses Feld an."
}
```

Aufeinanderfolgende Strings werden automatisch konkateniert. Dies gilt auch für Variablen.

7.3 spezifische Aktionen

Aufbauend auf die oben angebenenen Basisoperationen können nun die spezifischen Aktionen formuliert werden. Sie ergeben sich aus der in Kapitel 3 durchgeführten Untersuchung verschiedener Hubs.

Die in Abschnitt 4.2.3 aufgezählten herstellerspezifischen Aktionen werden hier nicht berücksichtigt. Andernfalls müßten entsprechende generische Aktionen definiert werden, um sie für Verfahren und Teilverfahren verfügbar zu machen.

Aufgrund der Menge der spezifischen Aktionen wird auf eine vollständige Dokumentation an dieser Stelle verzichtet. Die folgende Liste zeigt daher nur einige Beispiele, um die Anwendung der Basisoperationen zu verdeutlichen.

```
# ´*´ als Werkzeug, da dieser Schritt auch bei SNMP-Management
# durch die Konsole erfolgen muss

action set_ipaddr(HPFiberOpticHubPlus,*) {
input
  ip: IpAddr
  netmask: IpAddr
  router: IpAddr
  router2: IpAddr
  ttl: INTEGER
body
  basisop manual("Schliessen Sie ein Terminal an den Konsolenport
      des Hubs an. Geben Sie das Kommando IP. Geben Sie folgende
      Daten ein:
      IP Address: " ip "
      Netmask: " netmask "
      Primary Router: " router "
      Backup Router: " router2 "
      Time-To-Live: " ttl )
}

action set_ipaddr(FibermuxCrossbow,Konsole) {
input
  ip: IpAddr
  netmask: IpAddr
  router: IpAddr
  traphost: IpAddr
body
  basisop manual("Schliessen Sie ein Terminal an den Konsolenport
      des Hubs an. Druecken Sie zweimal ESC. Waehlen Sie Punkt 1
      ´Set IP Addresses´. Geben Sie folgende Daten ein:
      IP Address: " ip "
      Netmask: " netmask "
      Router: " router "
      SNMP Trap Host: " traphost)
}

action set_ipaddr(FibermuxCrossbow,RARP) {
input
```

```
  ip: IpAddr
  hubmac: EtherAddr
body
  basisop addline2file("/etc/rc.d/rc.rarp",
            "rarp -s " ip " " hubmac)
}

action set_ipaddr(SynopticsWorkgroup,KonfigFile) {
input
  ip: IpAddr
  netmask: IpAddr
  router: IpAddr
  hubname: String
  hubmac: EtherAddr
body
  basisop addline2file("/etc/bootptab",
            hubname ":ht=ethernet:ha=" hubmac ":ip=" ip)
}

action set_port_status(FibermuxCrossbow,LightWatch) {
input
  hubname: String
  port: PortNumber
  state: BOOLEAN
body
  basisop manual("Selektieren Sie den Hub " hubname " mit der Maus.
      Oeffnen Sie das Menue ´Hub/Detail´.
      Selektieren Sie Port " port " im ´Hub Detail´-Fenster.")
  case state {
    false: basisop manual("Waehlen Sie Control/Disable.")
    true: basisop manual("Waehlen Sie Control/Enable.")
  }
}

action set_port_status(FibermuxCrossbow,SNMP) {
const
  downup: ARRAY [BOOLEAN] INTEGER = (0,1)
input
  ip: IpAddr
  port: PortNumber
  state: BOOLEAN
body
# .iso.org.dod.internet.private.enterprises(1.3.6.1.4.1).
```



```
# fibermux(120).fmxVariables(2).fmxHubs(1).fmxCrossbowGrp(2).
# fmxCbCPortTable(2).fmxCbCPortEntry(1).fmxCbCPEnable(2).<c>.<p>
  basisop snmpset(ip,
                "1.3.6.1.4.1.120.2.1.2.2.1.2." port.c "." port.p,
                downup[state])
}

action config_firmware(SynopticsWorkgroup,KonfigFile) {
input
  hubname: String
  hubboot: String
body
  basisop editfile_keyfield("/etc/bootptab",":","=",
          $hubname,"bf",$hubboot)
}
```

7.4 generische Aktionen

Die generischen Aktionen wurden in Kapitel 4 auf der Grundlage der untersuchten Geräte ermittelt.

```
action set_ipaddr {
input
  ip: IpAddr
  netmask: IpAddr
  router: IpAddr
  router2: IpAddr optional for (HPFiberOpticHubPlus,Konsole)
  ttl: INTEGER optional for (HPFiberOpticHubPlus,Konsole)
  traphost: IpAddr optional for (FibermuxCrossbow,Konsole)
  hubname: String optional for (SynopticsWorkgroup,KonfigFile)
  hubmac: EtherAddr optional for (SynopticsWorkgroup,KonfigFile)
semantics
  "Weist dem Geraet eine IP-Adresse fuer das Komponentenmanage-
    ment zu. ´ttl´ ist die TTL fuer vom Geraet gesendete IP-Pakete.
   ´traphost´ ist die IP-Adresse, die SNMP-Trap-Pakete erhaelt."
}

action set_port_status {
input
  port: PortNumber
```

```
  state: BOOLEAN
semantics
  "Port ´port´ wird enabled, falls ´state´==true;
   ´port´ wird disabled, falls ´state´==false."
}

action set_module_status {
input
  module: ModuleNumber
  state: BOOLEAN
semantics
  "Modul ´module´ wird enabled, falls ´state´==true;
   ´module´ wird disabled, falls ´state´==false."
}

action config_firmware {
input
  bootsrv: String
  bootport: String optional for (FibermuxCrossbow,SNMP)
  bootfile: FileName
  hubname: String optional for (SynopticsWorkgroup,KonfigFile)
semantics
  "Konfiguriert Hub ´hubname´ und Managementstation ´bootsrv´ fuer
   das Laden einer Firmware-Datei ´bootfile´ beim Bootvorgang"
}

action backup_pair {
input
  primport: PortNumber
  secoport: PortNumber
  errorcount: INTEGER optional for (HPFiberOpticHubPlus,*)
  errorthreshold: INTEGER optional for (FibermuxCrossbow,*)
  remote_ether: EtherAddr optional for (HPFiberOpticHubPlus,*)
  test_interval: TimeSpan optional for (HPFiberOpticHubPlus,*)
semantics
  "Definition eines Backup-Portpaares ´primport´,´secoport´.
   Die Umschaltung erfolgt, falls :
   - HPFiberOpticHubPlus:
       ´errorcount´ aufeinanderfolgende Testpakete im Intervall
       ´test_interval´ an ´remote_ether´ ohne Antwort blieben
   - FibermuxCrossbow:
       ´errorthreshold´ angekommene Pakete fehlerhaft waren oder
       bei 10Base-T oder 10Base-F kein Signal gefunden wird."
```



```
}
```

7.5 generische Verfahren

Hier ist nur ein generisches Verfahren anzuführen, nämlich die Erstkonfiguration eines Sternkopplers. Darin zeigt sich die erhebliche Vereinfachung, denn ohne Verwendung der hier entwickelten Generik hätte man für die in Kapitel 3 untersuchten Abläufe fünf Verfahren angeben müssen:

- Erstkonfiguration eines HP Fiber-Optic Hub Plus via Konsole
- Erstkonfiguration eines HP Fiber-Optic Hub Plus via SNMP
- Erstkonfiguration eines Fibermux Crossbow via LightWatch
- Erstkonfiguration eines Fibermux Crossbow via SNMP
- Erstkonfiguration eines Synoptics Workgroup Hub via Konfig-Datei

Aufgrund der Einfachheit des untersuchten Verfahrens ist es nicht erforderlich, Teilverfahren zu verwenden.

In `erstkonfig_hub` werden zwei Schleifen verwendet, deren Parameter vom zu konfigurierenden Gerät abhängen. Man könnte also fragen, ob es nicht sinnvoller wäre, generische und spezifische Aktionen beispielsweise zum Setzen aller Port-Zustände zu definieren, die als Parameter entsprechend ein Feld erhalten. Jedoch ist zu berücksichtigen, daß die hier verwendete Aktion `set_port_status` nicht nur in der Erstkonfiguration eingesetzt wird. Bei Veränderungen im Betrieb ist die Manipulation einzelner Ports erforderlich. Und eine zusätzliche Aktion würde dem Prinzip der Schichtung zuwiderlaufen und die Menge der erforderlichen spezifischen Aktionen unerträglich steigern.

```
genproc erstkonfig_hub {
input
  ip: IpAddr
  netmask: IpAddr
  router: IpAddr
  router2: IpAddr optional for (HPFiberOpticHubPlus,Konsole)
  ttl: INTEGER optional for (HPFiberOpticHubPlus,Konsole)
  traphost: IpAddr optional for (FibermuxCrossbow,Konsole)
  hubname: String optional for (SynopticsWorkgroup,KonfigFile)
  hubmac: EtherAddr optional for (SynopticsWorkgroup,KonfigFile)
  bootsrv: String
  bootport: String optional for (FibermuxCrossbow,SNMP)
```

```
  bootfile: FileName
  modul_aktiv: ARRAY [ModuleNumber] BOOLEAN
  port_aktiv: ARRAY [PortNumber] BOOLEAN
  backup_pairlist: ARRAY [INTEGER] RECORD {
                                     primary: PortNumber
                                     backup: PortNumber
                                   }
  errorcount: INTEGER optional for (HPFiberOpticHubPlus,*)
  errorthreshold: INTEGER optional for (FibermuxCrossbow,*)
  remote_ether: EtherAddr optional for (HPFiberOpticHubPlus,*)
  test_interval: TimeSpan optional for (HPFiberOpticHubPlus,*)
body
  action set_ipaddr(ip,netmask,router,router2,ttl,traphost);
  case hubtype.ismodular {
  true: loop i from hubtype.firstmodule to hubtype.lastmodule {
          action set_module_status(i,modul_aktiv[i])
        }
  }
  loop i from hubtype.firstport to hubtype.lastport {
    action set_port_status(i,port_aktiv[i])
  }
  action config_firmware(bootsrv,bootport,bootfile,hubname)
  loop i from 1 to length(backup_pairlist) {
    action backup_pair(backup_pairlist[i].primary,
                       backup_pairlist[i].backup,
                       error_threshold, error_count,
                       remote, test_interval)
  }
}
```

7.6 Einschränkungen der Beschreibungssprache

Die hier vorgestellte Beschreibungssprache ist bewußt einfach gehalten, sowohl damit sie dem Leser leicht verständlich ist, als auch damit die Konvertierung in eine werkzeugspezifische Syntax ohne größere Schwierigkeiten erfolgen kann. Daher ist es nicht verwunderlich, daß dieser erste Ansatz zur Beschreibung Mängel aufweist.

So ist es beispielsweise wünschenswert, den *Selektor* allgemeiner anzugeben, damit einzelne spezifische Aktionen, die für mehrere Geräte gleich sind, auch nur einmal spezifiziert werden müssen. Oft kommt es vor, daß verschiedene Geräte eines Herstellers bestimmte Parameter gemeinsam haben, um dies zu unterstützen, kann zur Geräteangabe statt eines einfachen Bezeichners ein Tupel (*Hersteller*, *Modell*) verwendet werden.

Ebenfalls ist es unschön, daß die Parameter der generischen Aktionen und Verfahren aus den Deklarationen der spezifischen Verfahren manuell abgeschrieben werden müssen. Hier wäre ein unterstützendes Eingabewerkzeug oder ein Prüfprogramm eine sinnvolle Ergänzung.

Wie im Ausblick angesprochen, würde ein objektorientiertes Konzept dem Modell besser gerecht. Hier wurde jedoch Wert auf eine implementierungsfreundliche Konzeption gelegt, und dies ist auch gelungen, wie im nächsten Kapitel deutlich werden wird.

Kapitel 8

Implementierungskonzept

8.1 Das Projekt TERRA

Im Zusammenhang mit der Entwicklung des Rahmenbetriebskonzeptes entstand am Lehrstuhl von Prof. Hegering ein Werkzeug, um die Erfassung der erforderlichen Daten zu vereinheitlichen und zu automatisieren.

8.1.1 TERRA aus Benutzersicht

Das Werkzeug TERRA (**T**ool zur **Er**stellung eines **Ra**hmenbetriebskonzeptes) [Fis95b, FE94, FBE+94] ermöglicht die detaillierte Erfassung der Betriebsabläufe eines Unternehmens. Dabei wird für jedes Objekt im Rahmenbetriebskonzept ein Formular angelegt, das dessen Beschreibung enthält. Ferner werden die Beziehungen zu anderen Objekten in einem Grapheneditor festgelegt.

Ergebnis ist demzufolge eine duale Darstellung des Betriebskonzeptes einer Abteilung (oder eines Unternehmens) in grafischer und textueller Form. Die Formulare gewährleisten eine einheitliche und systematische Beschreibung der Objekte. Die Bedienung des Programms erfolgt durch Menüs, die der grafischen Darstellung zugeordnet sind. Ferner kann durch Auswahl eines Objekts im Graphen das zugehörige Formular dargestellt werden und umgekehrt durch Auswahl eines Verweises im Formular der zugehörige Knoten im Graphen markiert werden.

Konkret werden die folgenden Objekttypen des Rahmenbetriebskonzeptes erfaßt:

Dienst
: abrechenbare Leistung, die innerhalb einer Abteilung oder nach außen erbracht werden kann. Eine Zusammenfassung in Dienstpakete und Dienstkataloge ist möglich.

Aufgabe
: aus dem Dienst sich ergebendes Arbeitsziel, einschließlich der erforderlichen Qualifikationen und Berechtigungen des ausführenden Mitarbeiters. Die Bildung von Aufgabenfeldern und Teilaufgaben ist möglich.

Eine Sonderform der Aufgabe ist die Basisaufgabe. Sie ist keinem Dienst zugeordnet, da sie die Erbringung verschiedener Dienste indirekt unterstützt. Ein typisches Beispiel ist das Aufrechterhalten der zur Diensterbringung benutzten Infrastruktur.

Verfahren
: Arbeitsablauf, der eine Aufgabe erfüllt. Typischerweise erfolgt eine weitere Untergliederung in Teilverfahren, die sich aus Aktionen zusammensetzen. Einer Aufgabe steht im Prinzip ein Verfahren gegenüber, jedoch kann es mehrere Verfahren geben, je nachdem, welche Geräte und Werkzeuge verwendet werden.

Aktion
: elementarer Arbeitsschritt eines (Teil-)Verfahrens.

Funktion
: genaue Bezeichnung des von der Aktion benutzten Teilaspektes des verwendeten Werkzeugs.

Werkzeug
: von einer Aktion benutzte Hard- und Software.

Demnach erfolgt der Übergang von der generischen zur spezifischen Beschreibung zwischen den Ebenen Aufgabe und Verfahren. Bild 8.1 zeigt einen Ausschnitt aus der grafischen Darstellung eines Betriebskonzeptes mit einigen Daten aus der im ersten Teil durchgeführten Untersuchung.

Das Werkzeug TERRA wird erfolgreich zur Analyse von Betriebskonzepten eingesetzt, jedoch weist es gravierende Mängel in der Darstellung der Verfahren und Aktionen auf. Erstens erlaubt es keine Darstellung von Verfahrensabläufen, sondern kann nur die an einem Verfahren beteiligten Aktionen gewissermaßen als unsortierte Liste darstellen. In der Praxis läßt sich nicht einmal die Reihenfolge der Aktionen beeinflussen, da die verwendete Graph-Visualisierungssoftware zur Reduzierung sich kreuzender Kanten selbständig die Reihenfolge der Knoten vertauscht. Nur in der Formularbeschreibung kann der Ablauf eines Verfahrens beschrieben werden, und mangels fester Regeln geschieht dies nur umgangssprachlich. Zweitens wird eine Gesamtdarstellung aller Objekte dem Wunsch nach Übersicht nicht gerecht. Schon bei kleinen Beispielen entstehen sehr viele Aktionen, die vom Programm alle in einer Linie angeordnet werden.

TERRA ist daher als Analysewerkzeug nur insoweit geeignet, als man Verfah-

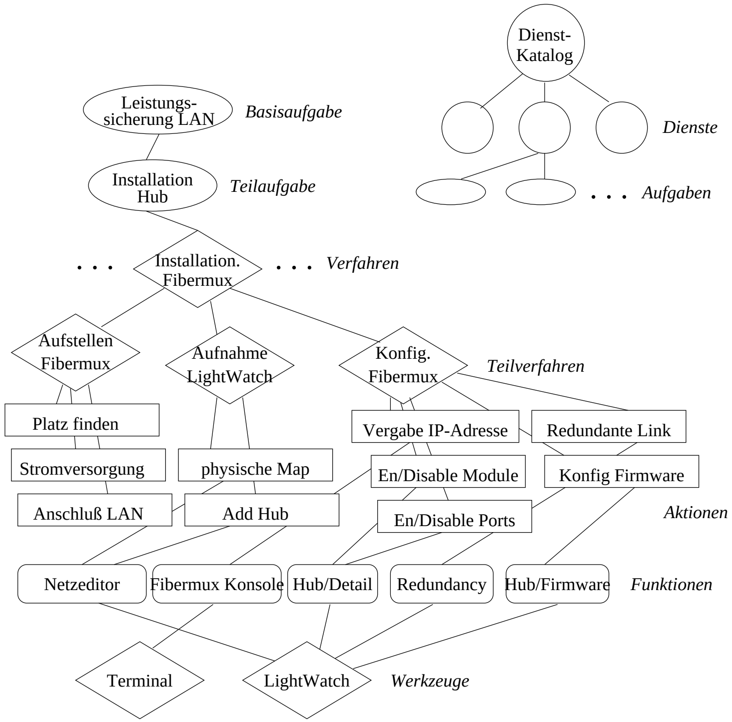

Abbildung 8.1: Beispiel für einen TERRA-Graphen (Schema)

ren als *black Box* ansieht und auf die Ebene der Aktionen verzichtet. Da jedoch die Modellierung der Aktionen Ziel dieser Arbeit ist, stellt TERRA dafür keine geeignete Implementierungsgrundlage dar.

8.1.2 TERRA aus Entwicklersicht

Die Architektur von TERRA (Bild 8.2) ist bestimmt durch ein Client-Server Konzept. Der Client stellt die Benutzeroberfläche zur Verfügung und besteht aus mehreren Funktionsmodulen, die von einem Steuerprogramm verwaltet werden. Der Server ist für die Datenhaltung zuständig. Eine Motivation für diese Trennung war es, paralleles Arbeiten an einem Datenbestand insbesondere in der Erfassungsphase zu ermöglichen. Jedoch wurden die erforderlichen Schutzmechanismen für gleichzeitigen Zugriff nicht implementiert.

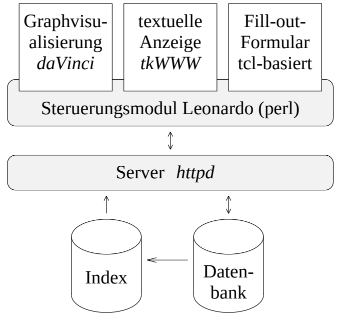

Abbildung 8.2: Architektur von TERRA

Die ursprüngliche Planung sah vor, dieses zunächst rein passive, Abläufe dokumentierende Werkzeug durch die Anbindung an eine Management-Plattform selbst zu einem Management-Werkzeug zu machen. Aufbauend auf die Ergebnisse in [Ber95] hat die vorliegende Arbeit dafür die nötige Beschreibungssprache festgelegt.

Wie oben ausgeführt, hat sich TERRA in der vorgestellten Form jedoch als ungeeignet für eine Ablaufsteuerung erwiesen. Daher wurde ein Projekt zur Implementierung eines anderen Werkzeuges begonnen.

8.1.3 Das Nachfolgeprojekt TERRA-II

TERRA-II [HS95] soll im Gegensatz zu seinem Vorgänger nicht zur Dokumentation vollständiger Betriebskonzepte benutzt werden. Vielmehr beschränkt es sich auf den unteren Bereich des Rahmenbetriebskonzeptes, die Verfahrens- und Werkzeugebene (vgl. Bild 5.1). Diese Ebene soll jedoch genauer modelliert werden, außerdem ist der Einsatz als Managementwerkzeug (durch eigene Funktionalität und/oder Anbindung an eine Management-Plattform) von Anfang an Entwicklungsziel. Daraus ergibt sich die Forderung nach einer formalen Ablaufbeschreibung, wie sie in Kapitel 6.3 entwickelt worden ist.

Als Grundlage für TERRA-II wurde das Software-Entwicklungssystem Khoros ausgewählt, da es die grafische Programmierung und damit Visualisierung von Abläufen gut unterstützt. Durch ein Fortgeschrittenenpraktikum wurde die grundsätzliche Eignung von Khoros für die Erfassung von Verfahrensbeschreibungen nach dem Rahmenbetriebskonzept festgestellt [Fis95a].

Im Gegensatz zu TERRA-I stellt das neue Werkzeug keine Client-Server-Lösung dar. Dies ist jedoch kein Problem, da im Gegensatz zu TERRA-I nicht die Erfassung sondern die Benutzung der Verfahrensdaten im Vordergrund steht. Gleichzeitige, verteilte Korrekturen am gleichen Datenbestand sind daher nicht erforderlich. Da sich die Verfahrensdaten in der Anwendungsphase kaum noch ändern, können diese Daten bei einer Installation des Werkzeugs dupliziert werden.

Khoros wird im folgenden Abschnitt vorgestellt, um anschließend ein Implementierungskonzept für TERRA-II auf dieser Basis zu entwickeln.

8.2 Implementierung generischer Verfahren in Khoros

8.2.1 Das Software-Entwicklungssystem Khoros

Das Software-Entwicklungssystem Khoros [Kho94a, Kho94b] unterstützt die Entwicklung von Programmen durch Kombination einfacher, Khoros-verträglicher Einzelmodule zu einem komplexeren Ganzen. Die Entwicklung dieser Module wird ebenfalls durch automatische Erstellung von Code- und Dokumentationsrahmen und durch Bereitstellung verschiedener auf Khoros abgestimmter Bibliotheksfunktionen unterstützt.

Die Kombination dieser Module erfolgt in dem graphischen Programm `cantata`, wobei die benötigten Module in einem Datenflußdiagramm verknüpft werden. Dabei werden Nebenläufigkeiten automatisch erkannt und ausgenützt. Khoros wird schwerpunktmäßig in der Numerik angewandt, daher finden sich für diesen Bereich viele Bibliotheksfunktionen (Fast Fourier Transformation, punktweise Bearbeitung eines dreidimensionalen Objektes und dergleichen). Es ist durch Erstellung eigener Module jedoch auch für andere Bereiche geeignet.

Khoros stellt eine Reihe von Objekttypen für die grundlegenden Bausteine (oben Module genannt) eines visuellen Programms zur Verfügung. Für die Implementierung der generischen Verfahren im Sinne des Rahmenbetriebskonzepts sind folgende Typen von Interesse:

Script Object
: Dieses Objekt gestattet die Einbindung von Skripten in die Khoros-Umgebung. Neben einigen Unix Shells wird auch Perl [WS91] als Skriptsprache unterstützt. Die Kommandozeilenparameter für Khoros-Objekte folgen festen Regeln; eine Funktion zu ihrer Auswertung steht zur Verfügung.

  Es besteht die Möglichkeit, auch für Skripten eine grafische Benutzerschnittstelle zu definieren.

Kroutine Object
: Eine Kroutine ist ein C-Programm, das auf Khoros-Libraryfunktionen aufbaut. Im Gegensatz zu Skripten ist hier die Einbindung in Khoros und die Unterstützung des Entwicklungsvorgangs besser. Ferner läuft ein kompiliertes C-Programm natürlich schneller ab als ein Perl-Skript, für das jedesmal der umfangreiche Perl-Interpreter geladen werden muß. Jedoch ist bei einer prototypischen Umsetzung die Entwicklungszeit vorrangig, so daß Kroutinen nur ausnahmsweise Verwendung finden werden.

Pane Object
: Pane ist die Bezeichnung für ein Eingabefenster, das die grafische Benutzerschnittstelle eines Khoros-Objekts darstellt. Ein Pane-Objekt beinhaltet selbst keinen Code, sondern stellt nur eine alternative Möglichkeit zum Aufruf eines anderen Programms (Kroutine oder Script Object, oder auch ein „fremdes" Programm) zur Verfügung. Typischerweise geht damit eine Spezialisierung einher, Khoros selbst enthält beispielsweise ein mathematisches Kroutine Objekt `karith1` für mathematische Operationen mit einem Parameter, und Spezialisierungen dazu in Form von Pane Objekten wie `abs` oder `sin`.

Diese Objekttypen werden anschließend mit Hilfe des visuellen Programmierwerkzeugs `cantata` durch Definition des Datenflusses (und in Einzelfällen darüberhinaus des Kontrollflusses) zu visuellen Programmen zusammengestellt. Dabei ist es möglich, Unterprogramme zu Prozeduren oder Encapsulated Workspaces zusammenzufassen. Die erstere Möglichkeit ist eine rein optische Veränderung, um eine Folge von Schritten in der Darstellung zu verbergen. Nur ein Encapsulated Workspace erlaubt die Wiederverwendung eines Unterprogramms mit Parameterübergabe, wie man es von Prozeduren gewohnt ist. Ein solcher Workspace kann mit Parametern versehen werden und stellt selbst ein Khoros-Objekt dar.

Beim Ablauf eines visuellen Programms in `cantata` werden die jeweils aktiven

Objekte optisch gekennzeichnet. So kann der Benutzer den Programmablauf verfolgen. Tritt ein Fehler in einem der Schritte auf, so wird das betroffene Objekt markiert und das grafische Programm angehalten.

Neben dem Einbau eines Objekts in Khoros-Anwendungen durch grafische Programmierung besteht auch die Möglichkeit, ein einzelnes Objekt von der Shell-Kommandozeile aus direkt aufzurufen; den Einträgen im visuellen Parameterformular entsprechen dabei Kommandozeilenoptionen. Bei Encapsulated Workspaces kann darüberhinaus ausgewählt werden, ob deren Flußdiagramm grafisch am Bildschirm angezeigt werden soll, so daß daran die Verarbeitung verfolgt werden kann. Wird dies nicht gewünscht, so unterscheidet sich der Ablauf eines Khoros-Objekts aus Benutzersicht nicht von der Ausführung irgendwelcher anderer Unix-Kommandos oder Skripten.

8.2.2 Darstellung generischer Verfahren in Khoros

In Abschnitt 6.3 wurden die Objekttypen

- generische Verfahren,
- generische Aktionen,
- spezifische Aktionen und
- Basisoperationen

definiert. Nun ergibt sich die Frage, wie diese auf systematische Weise in Khoros-Objekte umgesetzt werden können.

Um die Stärken von Khoros, wie grafische Flußdiagramme, Überwachung der Programmausführung am Flußdiagramm und Wiederverwendung von Objekten ausnutzen zu können, ist es erstrebenswert, möglichst viele Objekttypen im Rahmen der grafischen Programmbeschreibung auszudrücken. Dabei kann eine Wiederverwendung nur erfolgen, wenn die Objekte als Encapsulated Workspaces vorliegen.

Diese Workspaces werden üblicherweise interaktiv mit `cantata` entwickelt, jedoch ist es auch möglich, die entsprechenden Skriptdateien automatisch zu erzeugen. Es liegt daher nahe, eine automatische Umsetzung von der in Abschnitt 6.3 definierten werkzeugunabhängigen Sprache in die für Khoros erforderlichen Dateien vorzunehmen.

Wirklich implementieren muß man nur die Basisoperationen.

Im folgenden werden die einzelnen Objekttypen betrachtet.

generische Verfahren

Das generische Verfahren wird als Encapsulated Workspace implementiert.

Entsprechend den Vorgaben des Verfahrensablaufs, wie er durch Anwendung der in dieser Arbeit entwickelten Methodik gewonnen wurde, besteht es aus einer Verknüpfung generischer Aktionen oder Teilverfahren.

Teilverfahren werden auf die gleiche Weise implementiert, sie enthalten jedoch einen linearen Ablauf ohne Strukturelemente.

Um die Benutzerschnittstelle übersichtlich zu halten, müssen die erforderlichen Parameter gesammelt beim Aufruf des Verfahrens abgefragt werden. Die so gewonnenen Daten werden dann im Rahmen des Verfahrens den einzelnen Aktionen weitergegeben. *Ein* solches Datum ist der Typ des zu konfigurierenden Koppelelements. Der generische Teil des Verfahrensablaufs sollte davon in seiner Struktur unbeeinflußt bleiben, erst die Aktionen werten diese Information aus.

Es wird zu prüfen sein, auf welche Weise herstellerspezifische Aktionen in die generischen Verfahren einbezogen werden. Um der Bedeutung der Schichten gerecht zu werden, bietet sich an, auch herstellerspezifische Aktionen in generische Aktionen „einzupacken", um sie so verfügbar zu machen. Darüberhinaus kann es sinnvoll sein, ein allgemeines, generisches Teilverfahren „herstellerspezifische Schritte" zu definieren, das die Steuerung solcher Aktionen übernimmt. Andernfalls würde nämlich das eigentliche Verfahren mit zuvielen Aktionen belastet, die nicht generell zutreffen, so daß die Übersichtlichkeit und Aussagekraft des grafischen Programms darunter litte.

generische Aktionen

Das Objekt für die generische Aktion ist für die Auswahl der entsprechenden spezifischen Aktion zuständig. In der in Abschnitt 6.3 verwendeten Syntax ergibt sich diese Auswahl ohne Angabe einer prozeduralen Vorschrift aus der Benennung der spezifischen Aktionen. Dies kann jedoch so in Khoros nicht wiedergegeben werden.

Daher wird eine Mehrfachauswahl verwendet, die anhand des Gerätetypen sowie des zu verwendenden Werkzeuges die entsprechende spezifische Aktion ausführt. Diese Mehrfachauswahl wird durch einen Encapsulated Workspace realisiert. Dabei werden ausschließlich die dem Workspace entsprechenden spezifischen Aktionen aufgerufen.

spezifische Aktionen

Hier wird eine bestimmte Aktion bei einem bestimmten Gerät beschrieben. Die Aktion hat eine Benutzerschnittstelle, die von Khoros beim Aufbau visueller Programme benötigt wird und daneben die Aktion auch isoliert zugänglich macht.

Da viele Aktionen nur parametrisierte Versionen derselben Basisoperationen darstellen, können sie als Pane Objects implementiert werden. Sind jedoch mehrere Basisoperationen durchzuführen, wie beispielsweise im Falle von `backup_pair(Crossbow,SNMP)`, wo mehrere `snmpset`-Operationen erforderlich sind, so reicht dafür die Mächtigkeit des Pane Object nicht aus. In diesem Fall ist auch hier ein Encapsulated Workspace zu verwenden, der jedoch nur Basisoperationen aufruft.

Es wäre auch denkbar, die spezifischen Aktionen als Procedures innerhalb „ihrer" generischen Aktion zu realisieren, da auf sie nur über die generische Aktion zugegriffen werden soll. Jedoch würde dieses Vorgehen das Testen erschweren, weil nur auf der Ebene der generischen Aktionen getestet werden könnte.

Es gibt noch ein weiteres Argument dafür, auch hier Encapsulated Workspaces zu verwenden. Bei der Implementierung soll ja eine automatische Übersetzung von der hier definierten Beschreibungssprache in Khoros erfolgen. Die Beschreibungssprache weist zwischen den verschiedenen Ebenen deutliche Parallelen auf, so daß im wesentlichen der gleiche Übersetzer für die verschiedenen Ebenen eingesetzt werden kann, wenn die Darstellung des Zielobjekts ebenfalls übereinstimmt.

Basisoperationen

Die Basisoperationen sind die einzigen Objekte, die individuell für das Khoros-basierte Werkzeug inplementiert werden müssen. Sie werden als Kroutine oder Script Objects realisiert. Zur Abwägung zwischen Skripten und in C geschrieben Kroutinen siehe den obigen Abschnitt 8.2.1.

8.2.3 Integration von TERRA-II in TERRA-I

Das neue Werkzeug TERRA-II stellt aufgrund seiner Spezialisierung keinen Ersatz für TEERRA-I dar. Vielmehr handelt es sich um eine Ergänzung, da TERRA-II genau die Funktionen enthält, die bei TERRA-I zu kurz kommen. Um ein vollständiges Werkzeug zur Unterstützung der Erstellung und Analyse von Betriebskonzepten zu erhalten, bietet es sich an, beide Werkzeuge miteinander zu verbinden.

TERRA-I ist für die Darstellung der Aktionen ungeeignet, es liegt daher nahe diese Ebene, sowie die damit eng verbundene Ebene der Funktionen, aus der Darstellung von TERRA-I zu entfernen. Da die eingesetzten Werkzeuge jedoch für eine Gesamtanalyse von großer Bedeutung sind, sollte die Werkzeugebene erhalten bleiben, sie kann direkt an die Verfahrensebene angebunden werden.

Wenn nun genauere Informationen über Verfahren, insbesondere die verwendeten Aktionen, benötigt werden, könnte über einen zusätzlichen Menüpunkt in der grafischen Darstellung von TERRA-I das neue Werkzeug TERRA-II mit dem

Flußdiagram für das ausgewählte Verfahren gestartet werden. Von dort aus ist eine weitere Erforschung der zugehörigen Aktionen möglich.

Gegenwärtig sind generische Verfahren in TERRA-I nicht berücksichtigt. Um zu verdeutlichen, welche Geräte durch generische Verfahren unterstützt werden, könnte unterhalb der Verfahrensebene eine Geräteebene eingefügt werden. Da die verwendeten Werkzeuge (beispielsweise im Falle von Element-Managern) von den zu bearbeitenden Geräten abhängen, liegt es nahe, die Werkzeugebene unterhalb der Geräteebene anzuordnen und diesen Zusammenhang durch Beziehungslinien zu verdeutlichen (Bild 8.3).

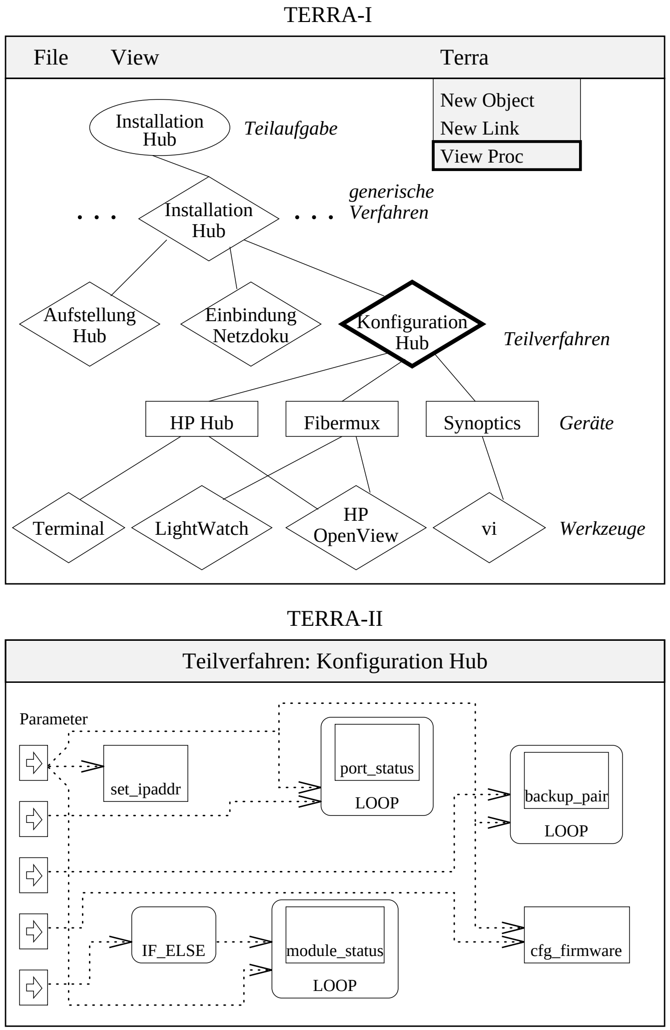

Abbildung 8.3: Koppelung von TERRA-I und TERRA-II

Zusammenfassung und Ausblick

Zusammenfassung

Die vorliegende Arbeit hat im ersten Teil existierende Netzkomponenten im Hinblick auf das Konfigurationsmanagement untersucht. Aufgrund der dabei vorgefundenen Parallelen konnten generische Aktionen abgeleitet werden.

Im zweiten Teil wurde eine Methodik gezeigt, wie durch einen bottom-up Ansatz mit vertretbarem Arbeitsaufwand die Grundlage für eine generische Sicht auf die Konfigurationsvorgänge geschaffen werden kann. Mithilfe einer einfachen formalen Sprache wurden die im ersten Teil erfaßten Aktionen sowie das generische Verfahren „Erstkonfiguration eines Sternkopplers“ dargestellt.

Schließlich wurde ein Implementierungskonzept entwickelt, wie diese Generik in das gegenwärtig vom „Münchner Netzmanagement Team“ entwickelte Werkzeug TERRA-II eingebunden und mit dem bestehenden Werkzeug TERRA-I verknüpft werden kann.

Damit wurde am Beispiel des Konfigurationsmanagements demonstriert, daß generisches Management nicht nur möglich, sondern auch mit vertretbarem Aufwand praktisch umsetzbar ist.

Ausblick

Das Implementierungskonzept ist hinreichend detailliert, daß eine erste Implementierung innerhalb eines Fortgeschrittenenpraktikums erfolgen kann. Es erscheint zweckmäßig, zunächst manuell einige hier vorgestellte Objekte in Khoros einzugeben, um die Praxistauglichkeit des Implementierungskonzeptes zu überprüfen. Gleichzeitig können die Basisoperationen verwirklicht werden. Wenn das Konzept so bestätigt worden ist, kann in der zweiten Stufe ein automatischer Übersetzer von der hier vorgestellten Beschreibungssprache in die Encapsulated Workspaces von Khoros verwirklicht werden.

Die Anbindung anderer Managementwerkzeuge über eine API wird gegenwärtig im Rahmen einer weiteren Diplomarbeit [Kin95] untersucht. Diese Einbindung stellt einen wichtigen Schritt zur Praxistauglichkeit des Werkzeuges dar. Ausgehend von der hier vorgestellten Verfahrensbeschreibung müßten nämlich die gleichen Daten bezüglich eines Geräts bei jedem Managementvorgang (entsprechend einem generischen Verfahren) erneut eingegeben werden, weil eine Netzdatenbank fehlt, die diese Informationen bei der Erstkonfiguration aufnehmen und zum Abruf bei späteren Änderungen bereithalten kann. Das ist im tatsächlichen Betrieb niemandem zuzumuten.

Mit dieser Entwicklung geht die Einbeziehung der plattformspezifischen Teilverfahren in die Verfahrensbeschreibungen einher. Diese Problematik wurde in der vorliegenden Arbeit ausgeklammert.

Architekturelle Überlegungen

Bedingt durch den Bottom-up Ansatz ergibt sich eine große Zahl gerätespezifischer Parameter, die mehr oder weniger elegant in die Generik einbezogen wurden. Dies beruht auf der Entscheidung, dem Benutzer des zu entwickelnden Werkzeugs möglichst keine Konfigurationsparameter vorzuenthalten, um die Akzeptanz des Werkzeugs zu fördern. Durch entsprechende Gestaltung des Werkzeugs können die im konkreten Fall nicht zutreffenden Parameter verborgen werden, um eine bessere Übersicht zu gewährleisten.

Längerfristig wäre ein top-down Ansatz vorzuziehen (zu den damit einhergehenden Problemen siehe jedoch Abschnitt 6.3.3). Dazu eignet sich eine objektorientierte Darstellung: Ausgehend von der Definition generischer Objekte, die die Gerätetypen darstellen und als Methoden die für sie zutreffenden generischen Aktionen beinhalten, erhält man durch Spezialisierung Objekte für individuelle Geräte (Bild Z.1). Dabei sorgt die Vererbung der generischen Methoden für die Einhaltung der generischen Schnittstelle, während bei der Spezialisierung die konkreten Ausführungsvorschriften (entsprechend den hier vorgestellten spezifischen Aktionen) ergänzt werden, ferner können Methoden hinzugefügt werden, die spezifischen Sonderfunktionen entsprechen. Als Beschreibungsrahmen dafür bietet sich das Informationsmodell des OSI-Managements [ISO10165] an (vgl. 6.2).

Die in Kapitel 7 erarbeiteten Beschreibungen können dabei zu einem großen Teil weiterverwendet werden. Die generischen Verfahren bleiben bestehen, arbeiten jedoch auf den Methoden der Objekte, die den generischen Aktionen entsprechen. Die spezifischen Aktionen geben die Implementierung der Methoden an. Ob es sinnvoll ist, sich auch hier auf Basisoperationen abzustützen, hängt davon ab, wie unabhängig die Methoden in der verwendeten Umgebung implementiert werden. Abweichungen ergeben sich insofern, als man in der Top-Down Entwicklung

die Parameter eher erfordernisorientiert bestimmt und die Umsetzung auf die im Einzelfall erforderte Form lieber in die Implementierung der Methoden verlagert.

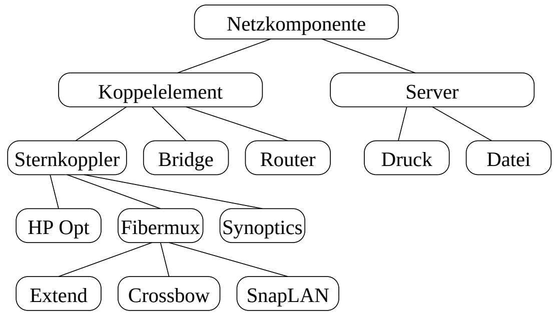

Abbildung Z.1: Beispiel für Objekthierarchie

Es wäre jedoch zu prüfen, wie systemübergreifende Objekte, zum Beispiel Komponentenpaare oder -gruppen, modelliert werden können, da das OSI-Management-Informationsmodell solche Objekte nicht vorsieht. Falls das verwendete Objektmodell multiple Inheritance unterstützt, böte sich an, solche verteilte Objekte sowohl von der Klasse ihrer Bestandteile abzuleiten als auch von einer weiteren Oberklasse „Komponentengruppe“. Ein Anwendungsbeispiel für die Konfiguration von Gruppen, das in dieser Arbeit dargestellt wurde, ist die Konfiguration der Hub-to-Hub Redundanz beim Fibermux Crossbow. Der Element-Manager bietet hierfür die Möglichkeit, beide beteiligten Komponenten auszuwählen, um sie gemeinsam zu konfigurieren. Eine solche Möglichkeit wäre zur Arbeitserleichterung und zur Vermeidung von Fehlkonfigurationen dringend erforderlich.

zukünftige Entwicklungen

Es wurde gezeigt, daß die Standardisierung im Internet-Management der technischen Entwicklung wesentlich hinterherhinkt, was entscheidende Ursache für den oben beklagten „Wildwuchs“ hinsichtlich der Konfigurationsparameter ist. Mit der hier vorgestellten Methodik läßt sich jedoch auch unter diesen eher ungünstigen Umständen ein tief integriertes Werkzeug schaffen.

Langfristig ist zu hoffen, daß durch die Entwicklung generischer Werkzeuge wie dem hier entworfenen mehr Übereinstimmung der Hersteller hinsichtlich der von ihren Geräten unterstützten Parameter erfolgt, weil ein herstellerunabhängiger Rahmen für solche Parameter wesentlich früher geschaffen werden kann, als dies durch die Standardisierungsvorgänge im Internet geschieht.

Weil jedoch die verschiedenen Hersteller auf der Basis des Funktionsumfangs ihrer Produkte konkurrieren, wird eine völlige „Gleichschaltung“ ihrer Geräte wohl nie eintreten.

Literaturverzeichnis

[Abe95] Sebastian Abeck. *Integrationstechniken im Netzmanagement.* In *Proceedings der KiVS 95*, Februar 1995.

[AEH+94] Sebastian Abeck, Th. Eckardt, Heinz-Gerd Hegering, Jürgen Lohrmann und Rene Wies. *Rahmenbetriebskonzept: Motivation, Begriffsbildung, prototypisches Beispiel.* TTI TECTRAN, März 1994.

[Ber95] Dieter Bertram. *Analyse bestehender Verfahren des Problemmanagements im Hinblick auf deren Unterstützung durch ein Trouble Ticket System.* Diplomarbeit, Technische Universität München – Institut für Informatik, Februar 1995.

[FBE+94] Christian Fischer, Oliver Blume, Karl Ewald, Martin Korndörfer, Gernot Riegert und Manfred Weis. *Programmierhandbuch zum Tool zur Erstellung eines Rahmenbetriebskonzepts (Terra)*, Dezember 1994.

[FE94] Christian Fischer und Karl Ewald. *Benutzerhandbuch zum Tool zur Erstellung eines Rahmenbetriebskonzepts (Terra)*, Dezember 1994.

[Fis95a] Bernhard Fischer. *Bewertung des Programmiersystems KHOROS hinsichtlich der Beschreibung und Anwendung von Verfahren.* Fortgeschrittenenpraktikum, Technische Universität München – Institut für Informatik, Mai 1995.

[Fis95b] Christian Fischer. *Konzeption eines Werkzeugs zur Erfassung von Betreiberanforderungen an ein integriertes Netz- und Systemmanagement.* Diplomarbeit, Technische Universität München – Institut für Informatik, Februar 1995.

[HA93] Heinz-Gerd Hegering und Sebastian Abeck. *Integriertes Netz- und Systemmanagement.* Addison-Wesley, 1. Auflage, 1993.

[HAW95] Heinz-Gerd Hegering, Sebastian Abeck und René Wies. *Rahmenbetriebskonzepte als Voraussetzung für ein betreibergerechtes integriertes*

Management von Corporate Networks. in: *Praxis der Informationsverarbeitung und Kommunikation*, 18. Jg.(1), 1995.

[HE94] Rainer Hauck und Karl Ewald. *Koppelelemente: Produktbeispiele – Hubs, Brücken und Router.* Hauptseminar, Technische Universität München – Institut für Informatik, 8. Juni 1994.

[HL93] Heinz-Gerd Hegering und Alfred Läpple. *Ethernet – Building a Communications Infrastructure.* Addison-Wesley, 1993.

[HS95] Kirsten Heiler und Peter Segner. *TERRA-II: Ein Werkzeug zur Spezifikation und Anwendung von Verfahren aus dem Netz- und Systemmanagement*, Februar 1995.

[ISO7498] International Organization for Standardization. *Information Technology – Open Systems Interconnections (OSI) Basic Reference Model.* International standard 7498, 1984.

[ISO8824] International Organization for Standardization. *Information Technology – Open Systems Interconnections – Specification of Abstract Syntax Notation One (ASN.1).* International standard 8824, 1987.

[ISO9596] International Organization for Standardization. *Information Technology – Open Systems Interconnections – Common Management Information Protocol Specification (CMIP).* International standard 9596, 1991.

[ISO10164] International Organization for Standardization. *Information Technology – Open Systems Interconnections – System Management Functions.* International standards 10164-x, 1992–1994.

[ISO10165] International Organization for Standardization. *Open Systems Interconnections – Structure of Management Information.* International standards 10165-x, 1992–1994.

[Kho94a] Khoral Research, Inc. *Khoros 2.0 – Toolbox Programming Manual*, 1994.

[Kho94b] Khoral Research, Inc. *Khoros 2.0 – Visual Programming Manual*, 1994.

[Kin95] Thomas Kinshofer. *Anforderungen an die Integration des Verfahrensbeschreibungswerkzeugs TERRA-II in eine Management-Plattform.* Diplomarbeit, Technische Universität München – Institut für Informatik, August 1995.

[RFC903] R Finlayson, T Mann, J Mogul und M Theimer. *Reverse Address Resolution Protocol.* IETF Network Working Group, June 1984.

[RFC951] W Croft und J Gilmore. *Bootstrap Protocol.* IETF Network Working Group, January 1985.

[RFC1157] J D Case, M Fedor, M L Schoffstall und C Davin. *A Simple Network Management Protocol (SNMP).* IETF Network Working Group, May 1990.

[RFC1213] K McCloghrie und M Rose. *Management Information Base for Network Management of TCP/IP-based Internets – MIB-II.* IETF Network Working Group, March 1991.

[RFC1286] K McCloghrie, E Decker, P Langille und A Rijsinghani. *Definitions of Managed Objects for Bridges.* IETF Network Working Group, November 1991.

[RFC1350] K Sollins. *The TFTP Protocol (Revision 2).* IETF Network Working Group, Oktober 1992.

[RFC1516] D McMaster und K McCloghrie. *Definitions of Managed Objects for IEEE 802.3 Repeater Devices.* IETF Network Working Group, September 1993.

[RFC1542] W Wimer. *Clarifications and Extensions for the Bootstrap Protocol.* IETF Network Working Group, October 1993.

[Seg95] Peter Segner. *Ein betreibergerechtes View-Konzept für das Netz- und Systemmanagement.* Dissertation, Technische Universität München – Institut für Informatik, 1995.

[Sta93] William Stallings. *SNMP, SNMPv2 and CMIP: the practical guide to network management standards.* Addison-Wesley, 1993.

[Tel94] Telekom TD*6243 und FA Bielefeld NeZ. *Einführungskonzept Netzwerk-Management Version 1.2.* Telekom, Januar 1994.

[vdH94] Andreas von der Heyde. *Analyse von Anforderungen an das WAN-Management unter besonderer Berücksichtigung einer Integration von LAN- und WAN-Komponenten.* Diplomarbeit, Technische Universität München – Institut für Informatik, August 1994.

[Wei94] Hans-Peter Weiß. *Erstellung eines Konzepts zur Datenintegration im Netzmanagement und dessen Anwendung auf eine konkrete Netzbetreiber-Organisation.* Diplomarbeit, Technische Universität München – Institut für Informatik, November 1994.

[WS91] Larry Wall und Randal L Schwartz. *Programming Perl.* O'Reilly and Associates, Inc, 1991.