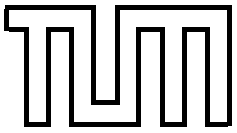


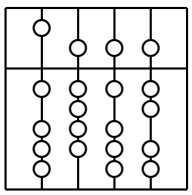
INSTITUT FÜR INFORMATIK

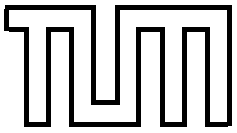
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwurf einer Architektur zur Integration von Netzplanungs- und -managementwerkzeugen in eine VR-Umgebung

Bearbeiter:	Timo Baur
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Dr. Gabi Dreo Rodosek Dr. Victor Apostolescu





INSTITUT FÜR INFORMATIK

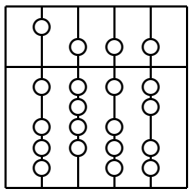
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Entwurf einer Architektur zur Integration von Netzplanungs- und -managementwerkzeugen in eine VR-Umgebung

Bearbeiter:	Timo Baur
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Dr. Gabi Dreo Rodosek Dr. Victor Apostolescu

Abgabetermin:	15. August 2002



Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. August 2002

. . . . . .

(Unterschrift des Kandidaten)

Zusammenfassung

In dieser Diplomarbeit wird ein Rahmenwerk entworfen, das es ermöglicht, die beim Management von Rechnernetzen anfallenden Daten dreidimensional und dynamisch zu visualisieren. Dabei wird dem Anwender die Möglichkeit gegeben, durch die erzeugte Darstellung zu navigieren und mit ihr zu interagieren.

Zentraler Gesichtspunkt beim Entwurf des Systems sind die allgemeine Integration in Managementplattformen und die aus einem Einsatz im Netzmanagement folgenden Anforderungen. Diese werden ausgehend von allgemeinen Aufgaben des Netzmanagements, den Erfordernissen des praktischen Rechenbetriebes und einem psychologischen Veranschaulichungsmodell festgelegt.

Um die Managementanwendungen und die von ihnen produzierten Daten in eine dreidimensionale Darstellung integrieren zu können, wird beim Entwurf des Systems ein Modell zur Informationsvisualisierung berücksichtigt und mit einem Modell für Managementplattformen kombiniert. Es werden semantische und funktionale Einheiten des Visualisierungsprozesses identifiziert und ein erweiterbares Datenmodell entworfen.

Das Ergebnis ist die Spezifikation eines grundlegenden Modells zur Integration von Managementanwendungen in eine VR-Umgebung. Dieses sieht eine Schnittstelle vor, über die verschiedenste Anwendungen angebunden werden können, um eine homogene Visualisierung des dem Management zugrundeliegenden Rechnernetzes und der Zustände seiner Komponenten erzeugen zu können.

Das allgemein gehaltene Rahmenwerk bietet dabei beispielhaft bereits die grundlegenden Definitionen für die visuelle Darstellung von Kommunikationsarchitekturen an. Da diese einfach erweiterungsfähig sind, kann das System auch zur Darstellung anderer Sachverhalte, beispielsweise für das Systemmanagement, genutzt werden.

Die Tragfähigkeit der Architektur wird anhand der Implementierung eines Prototypen nachgewiesen. Anhand von Beispielen wird gezeigt, dass auf der Grundlage der Spezifikation dreidimensionale Visualisierungen der Netztopologie, die Konfiguration von VLANs und eine dynamische Darstellung von Verkehrsflüssen möglich sind. Der Prototyp wurde für die Komponentenarchitektur JMX realisiert und stellt ein funktionsfähiges VR-Visualisierungssystem für diese Managementumgebung dar.

Inhaltsverzeichnis

1 Einführung

Mit der ständigen Entwicklung neuer Verkehrs- und Kommunikationssysteme und neuer Ansätze in den Wissenschaften treten Netzstrukturen immer mehr in den Mittelpunkt. Sie finden sich im Alltag in unzählbarer Vielfalt. Einige Beispiele sind Verkehrsnetze wie Autobahnnetze, Bahnlinien und Flugstrecken. Festnetztelefonleitungen, Mobiltelefonnetze, Fernsehkabel und Computernetze entwickelten sich in der modernen Gesellschaft zu komplexen Kommunikationsnetzen, die den Erdball dichtmaschig umspannen. In Wissenschaftsdisziplinen wie der Chemie werden Moleküle aus netzartigen Strukturen zusammengebaut und die modernen Sozialwissenschaften interessieren sich für Beziehungsnetzwerke.

In all diesen Beispielen ist der Begriff des Netzes zentral. Er bezeichnet ein aus irgendeinem Material geknüpftes oder geflochtenes Maschenwerk [Meyers]. Um sich ein Bild dieser zusehends unübersichtlich werdenden Geflechte zu machen, um Zusammenhänge in ihnen finden und um sie einfacher aufbauen und analysieren zu können, sind *Visualisierungen* notwendig. Visualisierungen sind grafische Repräsentationen von Daten, mit der Absicht erstellt, diese für den Menschen einfacher interpretierbar zu machen oder bestimmte Aspekte der Daten darzustellen.

In der Fertigungsindustrie, der Chemie und der Medizin werden zu diesem Zweck bereits immer öfter dreidimensionale Visualisierungen eingesetzt und auch in den Sozialwissenschaften gibt es Ansätze, die Analyse sozialer Netzwerke mit Hilfe von dreidimensionalen Darstellungen durchzuführen [FWK 98]. Dies kann mit sog. *VR-Umgebungen* erreicht werden. Hierbei handelt es sich um Hard- und Software-Umgebungen, die mit Hilfe des Paradigmas der ,,virtuellen Realität" (VR)[1] dreidimensionale Ansichten der untersuchten Daten produzieren.

Ungewöhnlich ist, dass in der Informatik selbst, die ja die Werkzeuge für VR-Visualisierungen entwickelt, im Gegensatz zu anderen Disziplinen weniger Gebrauch davon gemacht wird. Eines der Einsatzgebiete, die sich wegen der verwendeten Netzstrukturen sehr gut dafür eignen, ist das *Netzmanagement*. Das Netzmanagement beschäftigt sich schwerpunktmäßig mit dem Management von Kommunikationsdiensten und Netzkomponenten [HAN 99] in Kommunikationsnetzen. Häufig werden dabei *Netzmanagementwerkzeuge* verwendet, die für die Verwaltung, Planung und Administration der Netze eingesetzt werden. Sie produzieren zu diesem Zweck neben Listen von Meßdaten o.ä. oft auch grafische Ausgaben wie beispielsweise Topologie-Darstellungen von Netzen.

Während es einige Ansätze gibt, mit Hilfe isolierter Visualisierungswerkzeuge gewisse Bereiche des Netzmanagements abzudecken, so wurden bisher kaum Anstrengungen unternommen, VR-Visualisierungen in ein integriertes Netzmanagement einzubinden. In der Vergangenheit wurden zur Visualisierung meist zweidimensionale Darstellungen verwendet, was u.a. in den beschränkten Möglichkeiten der vorhandenen Hardware begründet war. Durch die immer größer werdende Komplexität moderner Kommunikationsnetze sowie eine immer größere Informationsfülle und einem kontinuierlich anwachsenden Datenvolumen ist es aber schwieriger geworden, auf diese Weise den Überblick über die Vielzahl von Netzkomponenten und internen Abläufe zu behalten.

[1] Kapitel 1.2 geht näher auf den Begriff der VR ein

Solche Darstellungen sind darüber hinaus oftmals unübersichtlich und erfordern vom Benutzer umfangreiche Vorkenntnisse und die Bereitschaft, sich in die komplexe Materie einzuarbeiten[2].

Die sinkenden Hardwarekosten in neuerer Zeit, die Entwicklung von plattformübergreifenden Softwarestandards zur Grafikprogrammierung (OpenGL) und preiswerte, aber grafisch dennoch sehr leistungsfähige Systeme bieten inzwischen die technischen Möglichkeiten für die Realisierung von dreidimensionalen Abbildungen an fast jedem Terminal. Damit werden Benutzerschnittstellen möglich, mit deren Hilfe mehr Information auf einmal gezeigt werden kann, während diese vom Benutzer durch eine intuitivere Darstellungsweise und die Möglichkeit eines aktiven Betrachtens schneller rezepiert (aufgenommen) wird. Neuere Forschungen aus der Psychologie zum Thema virtuelle Realität haben bereits entsprechende Modelle entwickelt[3].

Weitere Einsatzmöglichkeiten finden sich im Bereich rechnergestützter Gruppenarbeit mit dem Einsatz als *CSCW-System* (Computer-Supported Cooperative Work). Eine hinreichend integrierte VR-Umgebung kann eine Abbildung eines Rechnernetzes bieten, die sowohl als Interaktionsobjekt als auch als ,,Kulisse" sämtlicher anfallender Administrationstätigkeiten dienen kann. Durch eine eindeutige räumliche Positionierung jedes Anwenders innerhalb dieser Umgebung kann dieser auch alle anderen Personen, die zu diesem Zeitpunkt das System verwenden und an nahegelegenen Objekten arbeiten, sehen und mit ihnen kommunizieren. Diese sog. ,,Telepräsenz"[4] in Verbindung mit Text- oder Sprachübertragung ermöglicht auf einfache Art und Weise eine Zusammenarbeit interner wie externer Spezialisten am selben Projekt.

In der vorliegenden Arbeit wird ein Rahmenwerk entwickelt, auf dessen Basis solche Visualisierungen im Rahmen des Netzmanagements erzeugt werden können. Die Entwicklung eines Prototypen auf der Basis der entworfenen Architektur soll ferner erste Ergebnisse liefern und zu Anhaltspunkte bieten, um die Leistungsfähigkeit der Architektur abschätzen zu können.

[2] siehe Kapitel 1.1

[3] Kapitel 1.3 geht kurz auf entsprechende psychologische Modelle ein.

[4] ,,The team will get a sense of presence of the remotely located person, which is called telepresence" [Buxt 92]

1.1 Eine dreidimensionale Umgebung für das Netzmanagement

Das Management von Kommunikationsnetzen betrifft vielfältige physikalische und organisatorische Einheiten, deren sachgemäße Planung und Verwaltung[5] ihre Funktion sicherstellen. Dabei handelt es sich beispielsweise um Übertragungs- und Vermittlungseinrichtungen wie verschiedene Arten von Kabeln, Hubs, Switches, Bridges, Router und Medienkonverter oder um Protokollinstanzen.

Netzmanagement-Anwendungen haben die Aufgabe, die Planung und Verwaltung dieser Einheiten durchzuführen und zu erleichtern. Zu diesem Zweck erzeugen sie grafische Ausgaben für die Benutzer und Netzadministratoren. Meist beschränken sie sich dabei auf Textausgaben oder die Darstellung einfacher Grundformen der Netztopologie wie beispielsweise Bus, Ring oder Stern[6].

Die Topologie von Rechnernetzen wird bestimmt von logischen Zusammenhängen und Strukturen, die den Datenverkehr zwischen den Komponenten gestalten. Die elementaren Abläufe werden dabei durch Kommunikationsarchitekturen wie dem ISO/OSI-Referenzmodell oder dem Internet-Referenzmodell festgelegt. Eine Berücksichtigung aller Elemente dieser Architekturen in einer Visualisierung ist mit klassischen, auf einer zweidimensionalen Darstellung basierenden Benutzeroberflächen allerdings oft nur schwer und auf Umwegen möglich, denn alles auf einen Blick darzustellen, scheitert meist schon an der begrenzten Fläche des Bildschirms.

Um trotzdem umfassende Informationen über den Zustand eines Rechnernetzes grafisch erfassen zu können, wird deshalb meist eine sog. Fenstertechnik verwendet. Mit dieser Fenstertechnik können die jeweils problemrelevanten Abschnitte eines Netzes angezeigt werden. Jedes Fenster beinhaltet dabei eine bestimmte Sicht (sog. View) auf eine gewünschte Information über das Netz. Dies kann z.B. die Darstellung eines Subnetzes sein, eine aktuelle Statistik oder eine Darstellung der Portbelegung eines Switches. Die Navigation mit Hilfe von Fenstern ist praktisch und zweckmäßig, sie führt aber auch oft dazu, dass wichtige Fakten nicht intuitiv ins Auge fallen und gezielt gesucht werden müssen. Zwar bietet diese Fenstertechnik die Möglichkeit, größere Informationsmengen über den Zustand spezieller Aspekte des Netzes darzustellen, doch bleiben diese auf die untersuchten Bereiche begrenzt. Diese Details und Vorgänge, die darüber hinausgehen, können nicht gleichzeitig visualisiert werden. Meist beschränkt sich eine Sicht deshalb beispielsweise auf einen Verkehrsfluss oder auf einen statistischen Graphen. Abstraktionen, wie die Zuordnung zu Sender und Empfänger eines Datenpaketes, muss der Betrachter selbst vornehmen. Darüber hinaus müssen die bereitgestellten Sichten oftmals von Administratoren „händisch“ vor- und nachbereitet werden. Dies umfasst beispielsweise die Anordnung grafischer Repräsentationen von Komponenten im Netz und deren Zuteilung zu bestimmten Fenstern.

Eine Erhöhung der gleichzeitig erfassbaren Informationsmenge wäre deshalb wünschenswert und würde auch der steigenden Komplexität der darzustellenden Netze entgegenkommen. Eine Integration der Netzmanagementwerkzeuge und der anfallenden Daten in eine sinnvolle, automatisch erzeugte, dreidimensionale Umgebung könnte dies erreichen. Damit eröffnet sich die Möglichkeit, größere Strukturen darzustellen und große Datenmengen in abstrakter, intui-

[5] dazu gehört auch die Konfiguration

[6] siehe dazu Kapitel 2.2.1

tiver Form zu präsentieren. Die Darstellung von Bewegungen durch 3D-Modelle erleichtert es dem menschlichen Gehirn, auf vertraute Weise auch kompliziertere räumliche Anordnungen zu verstehen und zu behalten.

Die hinzukommende dritte Raumdimension (die Tiefe) hat dabei auch eine informationstragende Eigenschaft: Die in vielen Kommunikationsarchitekturen übliche Schichtung logischer Ebenen läßt sich auf der zusätzlichen dritten Koordinatenachse darstellen. Damit wird eine modellnahe Ansicht dieser Architekturen möglich.

Moderne Grafikbibliotheken und 3D-Engines gestatten es, den dargestellten Objekten verschiedene Oberflächen, Materialien und Verhaltenseigenschaften zu geben. Durch den Einsatz von Farben, Schattierungen, Beleuchtung und Transparenz können Unterschiede und Zustände organisatorischer Einheiten in einem Netz umfassend dargestellt werden. Die Möglichkeit der Anwendung von Animation in diesem Bereich erlaubt darüber hinaus eine dynamische Darstellung aller Objekte. Dadurch wird nicht nur eine ansprechende grafische Darstellung der Netzbestandteile denkbar, es kann auch die zeitliche Veränderung von Kennzahlen[7] über einen bestimmten zurückliegenden Zeitraum sichtbar gemacht werden. Beispielsweise kann sich die Breite der Darstellung eines Verkehrsflusses entsprechend der Auslastung in einem ausgewählten Intervall verändern.

Solche Umgebungen können also eine integrierte Darstellung verschiedener Management-Ebenen einschließlich der Protokolle unterstützen. Die Räumlichkeit erlaubt dabei eine detaillierte grafische Darstellung der Eigenschaften aller Managementobjekte. Was bislang hauptsächlich in Textform oder als flaches ,,Icon" dargestellt ist, kann nun durch dreidimensional dargestellte Objekte oder Objekteigenschaften ersetzt werden. Dadurch kann der Benutzer Netzmodelle schneller kognitiv erfassen und damit schneller reagieren.

[7] Kennzahl: Größe, die einen quantitativ messbaren Sachverhalt wiedergibt

1.2 Virtuelle Realität

Die Idee eines dreidimensionalen Informationsraumes, der eine flüssige Navigation erlaubt, ist im Konzept der ,,virtuellen Realität“ (VR) verwirklicht, dessen Kerncharakteristika dreidimensionale Simulation, Interaktion und Immersion einen solchen Raum realisieren.

Betrachtet man den Begriff ,,virtuelle Realität“, so scheint es eine Vielzahl verschiedener Definitionen zu geben. Zum einen kann VR als die Gesamtheit von Hard- und Software betrachtet werden, die dem Benutzer einen ihn einbeziehenden Bereich der Kommunikation als digitales, synthetisches System zur Verfügung stellt. In diesem Bereich kann der Benutzer als gleichberechtigter integrierter Bestandteil dieses Systems in Echtzeit agieren. VR basiert dabei auf einem mathematischem Raum, der architektonisch organisiert ist.

Zum anderen kann man den Begriff von der Wortbedeutung her verstehen, indem man ihn in seine Bestandteile zerlegt. Der Ursprung des Begriffes ,,virtuell“, findet sich im Lateinischen Wort ,,virtus“, und bedeutet ,,Kraft“ oder besser ,,Vermögen“ (der Kraft, dem Vermögen nach, potentiell [Eisl 10]). Bislang fand er vorwiegend im Bereich der Physik Bedeutung: ,,In der Optik kennt man den Begriff des virtuellen Bildes, und beschreibt damit den Effekt von Zerstreuungslinsen, ein nur scheinbar vorhandenes Bild zu erzeugen.“ [Borm 94]. Übertragen steht der Begriff des öfteren in der Bedeutung für ,,scheinbar“ oder dem Präfix ,,Schein-“ (im Zusammenhang mit anderen Begriffen). Im Sinnzusammenhang ergibt sich daraus, dass das, was virtuell ist, etwas anderes ist, als das, was tatsächlich vorhanden ist. Bei dem Begriff ,,Realität“ ist die Sachlage etwas schwieriger. ,,Nicht selten wird zwischen Realität und Wirklichkeit unterschieden.“ [Borm 94]. Das Reale ,,wird nicht etwa von uns geschaffen, sondern nur als solches bestimmt, methodisch im fortschreitenden Prozess der Wissenschaftsentwicklung . . . Diese objektiv-empirische Realität schließt eine gewisse Identität der Objekte nicht aus, sie ist [aber] von der absoluten Wirklichkeit des ‚An sich' zu unterscheiden, auf die sie hinweist.“ [Eisl 10]. Die Frage nach dem Wesen der Wirklichkeit zieht sich durch die ganze abendländische Philosophie und kann im Rahmen dieser Arbeit nicht erschöpfend behandelt werden. Deshalb soll ein definitorisches Postulat von [Sand 90] begnügen, das auch [Borm 94] seiner Begriffsklärung zugrunde legt. Demnach ist Wirklichkeit (im Gegensatz zur Realität) das, was vom Bewusstsein und vom Denken möglicherweise unabhängig existieren kann. In diesem Sinne ist ,,W[irklichkeit] . . . dann das Synonym für ‚objektive W[irklichkeit]', oder ‚materielle W[irklichkeit]': dasjenige, was nicht nur vorgestellt oder gedacht wird, sondern unabhängig von unserem Vorstellen oder Denken an sich besteht.“ [Sand 90]

Man könnte bei ,,virtueller Realität“ deshalb auch von einer *Simulation* von Wirklichkeit sprechen. Eine gute Simulation läßt etwas real erscheinen, was nicht oder nur annährend real ist. Tatsächlich handelt es sich bei dem Dargestellten immer nur um ein künstliches Modell der Wirklichkeit, womit ein weiteres Merkmal, die *Künstlichkeit* (Artificiality) genannt wäre. Virtuelle Landschaften in Flugsimulatoren z.B. des Militärs sind heute schon fotorealistisch und basieren auf realen Landschaften, die von Satelliten vermessen wurden. In den bisherigen Hauptanwendungsgebieten der VR wie Raumfahrt, Militär, Medizin, Physik, Mathematik, Architektur und Design war oftmals die Möglichkeit der Simulation der entscheidende Aspekt für den Einsatz

der Technologie.

Ein weiterer wichtiger Aspekt der VR ist die *Interaktion*. Sie bewirkt, dass das Verhältnis zwischen dem Betrachter und dem betrachteten Objekt ein anderes ist, als bei anderen Medien, nämlich nicht nur passiv, sondern auch aktiv. Dies erlaubt, in die simulierte Modellwelt einzutauchen, sie zu manipulieren, und führt gemeinsam mit fortgeschrittenen Visualisierungstechniken zur *Immersion* (in etwas eingebettet sein). Pioniere der VR wie Sutherland, Fisher oder Brooks versuchten anhand sensorischer Immersion durch HMDs (Head-mounted Displays) und Data-Gloves (Datenhandschuh) hochwertige Illusionen zu suggerieren. Die von Myron Krueger entwickelten Projektionsräume erlauben sogar eine Interaktion zwischen Mensch und Computer, ohne den Körper des Menschen durch HMD Hardware zu bedrängen, eine ,,Full Body Immersion" durch in den Raum projizierte Bilder. Um Immersion zu erreichen, ist eine Anpassung an die menschlichen Sinne notwendig, deshalb ist *Dreidimensionalität* ein weiteres Charakteristikum der VR.

Obgleich auch der Begriff Cyberspace in manchen Publikationen bedeutungsgleich mit VR benützt wird, erweitern einige Definitionen von Cyberspace die bisher genannten Eigenschaften um die der *Vernetzung*. Als Cyberspace werden über ein Netz gekoppelte Systeme betrachtet, in denen sich mehrere Teilnehmer gleichzeitig in einem Erlebnisraum befinden und in diesem auch miteinander interagieren können und/oder der Teilnehmer durch visualisierte Datenbanken navigieren kann. Der Begriff geht auf den Roman ,,Neuromancer" von William Gibson [Gibs 87] zurück, eine gute Definition, wie er verallgemeinert und für eine wissenschaftliche Verwendung verstanden werden kann, findet sich bei [Nova 92] auf Seite 225:

> ,,Cyberspace is a completely spatialized visualization of all information in global information processing systems, along pathways provided by present and future communications [sic] networks, enabling full copresence and interaction of multiple users, allowing input and output from and to the full human sensorium, permitting simulations of real and virtual realities, remote data collection and control through telepresence, and total integration and intercommunication with a full range of intelligent products and environments in real space."

Eine Integration von Netz- und Systemmanagementwerkzeugen und deren Objekten in eine VR-Umgebung könnte also, sofern das System Möglichkeiten zu einer Verteilung vorsieht, auch einen grundlegenden Schritt zur Umsetzung des Konzeptes Cyberspace bedeuten: eine verteilte Visualisierung der globalen Informationsverarbeitungssysteme auf Grundlage von Kommunikationsnetzen.

1.3 Perspektiven der Veranschaulichung

Mit seinen Aspekten Interaktion und dreidimensionale Simulation bietet das Konzept der VR vor allem sinnvolle Perspektiven zur Veranschaulichung der in einem Rechnernetz enthaltenen Informationen, was beispielsweise dem Bereich der Lehre und einer Förderung des öffentlichen Verständnisses der Funktionsweise des Internets entgegen käme. Es bietet aber auch die Chance, Arbeitsumgebungen zu schaffen, die sich durch eine bessere Rezeption des Dargestellten durch den Nutzer und damit eine höhere Effektivität auszeichnen.

S. Schwan und J. Buder, die sich mit der Psychologie und Pädagogik der VR, vor allem als Anwendung für Lernumgebungen beschäftigt haben [ScBu 01], nennen drei verschiedene Veranschaulichungsprinzipien der VR: Die abbildungstreue Veranschaulichung, die schematisierende Veranschaulichung sowie die konkretisierende Veranschaulichung.

Bei der *abbildungstreuen Veranschaulichung* sind reale Sachverhalte der Gegenstand, der möglichst realistisch und abbildungstreu dargestellt wird, wie z.B. ein historisches Gebäude oder ein dreidimensionales Modell eines Routers. Solche VR-Umgebungen weisen eine hohe Authentizität und somit ein hohes Transferpotential zum Benutzer auf. Der Realismus birgt aber hier die Gefahr in sich, dass die Inhalte eine elaborierte und reflektierte mentale Verarbeitung nur in geringem Maße anregen [Weid 89]. Zudem werden viele Informationen gezeigt, die nicht unbedingt relevant sind. Dies kann zur Folge haben, dass Benutzer ihre Aufmerksamkeit nicht auf die relevanten Informationen fokussieren, weil sie von anderen Darstellungsaspekten abgelenkt werden.

Als weitere Art der Veranschaulichung findet sich die *schematisierende Veranschaulichung*, bei der bewusst irrelevante Details ausgeblendet oder relevante Details eingeblendet werden. Dadurch erfolgen eine Fokussierung der Aufmerksamkeit und eine Vorwegnahme bestimmter Abstraktionsprozesse durch die dargestellten Objekte als ,,kognitive Werkzeuge“. Unter diese Kategorie fallen z.B. auch größenskalierte Veranschaulichungen, wenn der betrachtete Gegenstand zu klein oder zu groß ist, um wahrgenommen zu werden. Diese Art von Veranschaulichung macht außerdem auch eine kontextsensitive Bereitstellung von Information möglich.

Die *konkretisierende Veranschaulichung* präsentiert abstrakte Sachverhalte in bildlich analoger Weise [DSL 96]. Sie hat zwei Prinzipien als Grundlage: Einerseits das Prinzip der Sinnesskalierung[8]. Diese veranschaulicht für den Menschen nicht direkt wahrnehmbare Sachverhalte wie z.B. elektrische Felder. Andererseits das Prinzip der Verdinglichung[9]: gänzlich abstrakte Konzepte (z.B. ISO/OSI-Rahmenwerk) werden dabei in Objektform übergeführt [Winn 93]. Hier findet eine Umwandlung aufwändiger kognitiver Verarbeitungsprozesse in Prozesse perzeptueller Mustererkennung statt, was insgesamt zu einer reichhaltigeren kognitiven Repräsentation führt [ScBa 99]. Der Nutzen für die tägliche Praxis, aber auch für die Lehre, ist nicht zu unterschätzen.
Aufgrund der heterogenen und abstrakten Natur eines Rechnernetzes bieten vor allem die beiden letzteren Modelle der Veranschaulichung aussichtsreiche Ansatzpunkte, wie die in Rechnernet-

[8] engl.: transduction
[9] engl.: reification

zen enthaltene Information sinnvoll und anschaulich abgebildet werden kann. Deshalb werden diese im Verlauf der Anforderungsanalyse in Kapitel 2 in den Anforderungskatalog mit aufgenommen.

1.4 Aufgabenstellung und Aufbau der Arbeit

Im Netzmanagement werden zur Überwachung von Netzen und Systemen oftmals Visualisierungen der Netztopologie und der Zustände von Netzkomponenten eingesetzt, für die in der Vergangenheit meist zweidimensionale Darstellungsweisen verwendet wurden. Kapitel 1.1 wirft die Frage auf, ob dreidimensionale Visualierungen für das Management von Rechnernetzen einen Vorteil bringen könnten. Das Konzept der virtuellen Realität, das Dreidimensionalität mit Navigation und Interaktion verbindet, bietet dabei eine Alternative zu bisherigen grafischen Darstellungen an und wird in Kapitel 1.2 diskutiert. Neuere Arbeiten aus der psychologischen Pädagogik lassen erwarten, dass die Arbeit mit Rechnernetzen durch VR-Darstellungen erleichtert und das Verständis des Netzaufbaus und der Funktionsweise gefördert werden könnte. Im Abschnitt 1.3 werden diesbezügliche Modelle aufgezeigt.

Um erste Erfahrungen mit dem Einsatz solcher Systeme sammeln zu können, ist die Realisierung eines Visualisierungssystems, das diese Aufgaben übernehmen kann, notwendig. Im Zuge dieser Diplomarbeit soll deshalb eine Möglichkeit gefunden werden, Netzmanagementwerkzeuge in das Konzept der virtuellen Realität zu integrieren und dabei auch eine Interaktion mit den dargestellten Ressourcen zu ermöglichen. Weil sich Netzmanagementanwendungen aus Gründen der Integrationsfähigkeit, Kompatibilität und Interoperabilität meist auf Managementplattformen befinden (sollten), wird dabei bereits von Anfang an auf eine Integration in Managementplattformen abgezielt.

Zur Abschätzung der Komplexität des Problems und des von einem solchem System zu erbringenden Aufgabenspektrums wird in Kapitel 2 eine Anforderungsanalyse durchgeführt. Eine makroskopische Sichtweise auf Netzmanagementaufgaben mit Hilfe eines Top-Down Ansatzes hilft dabei, die zentralen Anforderungen aus dem Netzmanagement in großer Breite abzudecken. Zur Verifikation und zur weiteren Sammlung von Anforderungen dienen ferner einige Szenarien, die aus praktischen Problemstellungen des Managements des Münchner Wissenschaftsnetzes (MWN) abgeleitet sind, wie die Darstellung der Netztopologie und die Überwachung und Planung von Verkehrsflüssen und virtuellen LANs. Das Kapitel schließt mit einem zusammenfassenden Anforderungskatalog.

Auf den so gesammelten Anforderungen baut anschließend der Entwurf einer Architektur auf. Kapitel 3 zeigt, wie ein solches System aufgebaut sein sollte und spezifiziert dabei ein allgemeines Rahmenwerk. Basierend auf einem allgemeinen Visualisierungsmodell werden Dienste und Objekte definiert, die eine dynamische Darstellung der Topologie und der Zustände der Netzkomponenten und Kennzahlen erlauben. Ein erweiterbares Datenmodell für die Repräsentation von Daten und diesen zugehörigen grafischen Objekten bildet dabei die Grundlage. Es sieht die Integration verschiedenartiger logischer Konzepte wie Kommunikationsarchitekturen vor und erlaubt eine Anpassung der Darstellung, des Layouts und der Funktionalität an weitere Anwendungsfälle. Ferner werden Schnittstellen für die Anbindung an Netzmanagementanwendungen und Benutzer-Interfaces festgelegt und ein einfaches Übertragungsprotokoll für die Übertragung der erzeugten Grafiken vom Server zum Client definiert.

An den Entwurf anschließend wurde das Rahmenwerk implementiert und ein Prototyp geschaf-

fen, um die Möglichkeiten und Grenzen der Architektur aufzuzeigen. Kapitel 4 beschreibt die Besonderheiten der Implementierung dieses Prototypen, der auf den Java Management Extensions aufbaut, und zeigt anhand der bereits in der Anforderungsanalyse verwendeten Szenarien die Funktionstüchtigkeit und die Anwendungsmöglichkeiten des Systems. Einige Testläufe geben mit konkreten Zahlen Aufschluß über die Skalierbarkeit der darstellbaren Netze und lassen eine erste Einschätzung der Hardware-Anforderungen an den praktischen Einsatz zu.

Eine abschließende Diskussion der Ergebnisse in Kapitel 5 zeigt auf, ob der Ansatz eine Alternative zur zweidimensionalen Visualisierung darstellt, wo weiterer Entwicklungsbedarf besteht und wo die Grenzen eines solchen Systems liegen, aber auch, welche künftigen Weiterentwicklungen denkbar sind.

2 Analyse der Anforderungen

Jede Entwicklung eines Systems beginnt damit, dass für das spätere konkrete Design die grundsätzlichen Zielsetzungen und Anforderungen an die von dem System zu erbringenden Dienste festgelegt werden. Dies ermöglicht die Identifikation zentraler Problemfelder und stellt sicher, dass eventuell aus den Anforderungen resultierende Probleme bei der Entwicklung genügend berücksichtigt werden.

Die Sammlung von Anforderungen erfolgt in Kapitel 2.1 zunächst aus der Sicht des Netzmanagements. Aus fünf funktionalen Aufgabenbereichen des Netzmanagements werden die Aufgaben eines 3D-Visualisierungssystems für das Netzmanagement abgeleitet.

Anschließend werden in Abschnitt 2.2 Anforderungen aus der Visualisierung von Anwendungsfällen betrachtet. Hierzu wird die Visualisierung der Netztopologie, die Visualisierung und Konfiguration virtueller LANs und die Visualisierung von Verkehrsflüssen diskutiert.

Ein Katalog mit den gesammelten Anforderungen findet sich in Kapitel 2.3.

Zur besseren Übersichtlichkeit werden die im Folgenden gesammelten Anforderungen in ein Klassifikationsschema eingeordnet, das sich als übersichtlich und für die praktische Durchführung als gut geeignet erwiesen hat:

- Architekturanforderungen (AANF)
- Managementanforderungen (MANF)
- Visualisierungsanforderungen (VANF)
- Anforderungen an die Interaktion (IANF)

Zur besseren Auffindbarkeit der Anforderungen wird jede Anforderung mit dem entsprechenden Klassifikationsbereich und einer laufenden Nummer, wie beispielsweise "MANF 1" bezeichnet.

2.1 Anforderungen aus der Sicht des Managements

Die zu verwaltenden Komponenten in einem Rechnernetz sind äußerst heterogen. Informationen über die Ressourcen müssen deshalb in einer herstellerunabhängigen Weise übertragen werden. Deshalb wurden in den letzten Jahren vermehrt Anstrengungen unternommen, durch integrierte Managementarchitekturen mit definierten Schnittstellen und Protokollen zu den Managementanwendungen einheitliche Standards zum Datenaustausch zu schaffen. Dies eröffnet einem Visualisierungssystem die Möglichkeit, auf die verwalteten Komponenten in einer herstellerunabhängigen Weise über Schnittstellen zuzugreifen.

Heute gibt es Managementplattformen, die als Trägersysteme für Managementanwendungen fungieren. Indem sie eine gemeinsame Plattforminformationsbasis stellen, können die moderneren unter ihnen eine Datenintegration gewährleisten. Damit wird es möglich, dass die in jedem ihrer Werkzeuge gesammelte und verarbeitete Managementinformation auch allen anderen Werkzeugen der Plattform zur Verfügung steht. Zur Integration von Netzmanagementwerkzeugen in eine VR-Umgebung wird eine solche modulare, offene und datenintegrierte Managementplattform benötigt (AANF 1).
Die verschiedenen Managementanwendungen und Werkzeuge sollten über eine gemeinsame grafische Oberfläche angesprochen und gesteuert werden können. Deshalb ist eine Oberflächenintegration der mit dem Visualisierungssystem arbeitenden Managementwerkzeuge notwendig (AANF 2).

Um einer möglicherweise hohen Komplexität der beteiligten Managementanwendungen gerecht zu werden, wurde für die weitere Analyse der allgemeinen Anforderungen ein Top-Down Ansatz gewählt. Von der ISO/OSI-Managementarchitektur [ISO 7498-4] ausgehend[10], werden die einzelnen Funktionsbereiche des Managements untersucht und daraus Anforderungen an eine Architektur zur Integration von Netzmanagementanwendungen abgeleitet.

Eines der Ziele, das bei der Spezifizikation von ISO 7498-4 verfolgt wurde, war es, aus einem allgemeinen Managementfunktionsmodell Aufgaben und Lösungen für Managementarchitekturen herauszudestillieren. Es klassifiziert fünf Funktionsbereiche (*Systems Management Functional Areas*, SMFAs[11]), die den Aufgabenkomplex Management aufgliedern:

- Fehlermanagement (Fault)
- Konfigurationsmanagement (Configuration)
- Abrechnungsmanagement (Accounting)
- Leistungsmanagement (Performance)
- Sicherheitsmanagement (Security)

Die Bereiche werden im Folgenden anlehnend an [Garb 91] dargestellt und jeweils auf ihre Bedeutung für eine dreidimensionale Visualisierung untersucht.

[10] Sie geht von einer generalisierten Betrachtungsweise des Managements aus und eignet sich deshalb gut für diese Aufgabe.

[11] Anlehnend an die Anfangsbuchstaben der Bereiche wird dieses Schema auch als "FCAPS" bezeichnet

2.1.1 Fehlermanagement

Das Fehlermanagement umfasst Konzepte und Hilfsmittel zur Erkennung, Untersuchung und (ggf. teilweisen) Beseitigung von Störungen. In Form von *Alarmen* werden dabei akute Störungszustände gemeldet. Sie gehen aus von:

- einer fehlerhaften/überlasteten Komponente direkt,
- einer Komponente, die eine fehlerbehaftete/überlastete Komponente benutzt,
- oder einem Überwachungsmechanismus, der auch als Komponente repräsentiert sein kann

Wichtig erscheint, dass die Störungszustände angemessen visualisiert werden, um bei der Betrachtung der Darstellung Fehler schneller erkennen zu können. Alarme und Störungszustände sind logische Ressourcen eines Netzes. Es entstehen hier keine neuen Anforderungen, da eine Visualisierung logischer Ressourcen bereits in VANF 1 gefordert wurde.

Weiter sieht das Modell *Fehlerberichte (Event Reports*) vor. Das sind periodische oder aperiodische Informationssammlungen über das Störungsverhalten einer Komponente oder einer Menge von Komponenten. Störungen können aber auch durch die Auswertung von *Log- oder Statistikdaten* erkannt werden. Ebenso können *externe Ereignisse* Störungen signalisieren, z.B. ausgelöst von Temperaturdetektoren. Logdaten und Event-Reports sind für Managementplattformen fraglos notwendig. Obwohl eine Diskussion einer dreidimensionalen Visualisierung von Logeinträgen sowie deren Repräsentation in einer dreidimensionalen Metapher denkbar wäre, würde sie hier aber zu weit führen. Deshalb wird vorgeschlagen, zunächst lediglich eine klassische textbasierte Visualisierung von Logdaten zu verwenden. Sowohl eine textbasierte, als auch eine grafische Darstellung wird durch Anforderung VANF 1 bereits abgedeckt.

Die Diagnose von Fehlern ist darauf ausgerichtet, die Ursachen einer Störung zu erkennen. Es werden zwei Formen unterschieden:

- die *Symptomgesteuerte Diagnose* – sie knüpft an Fehlerberichte an.
- und die *Testgesteuerte Diagnose* – sie testet systematisch (alle) Systemkomponenten auf die Ausführung ihrer normalen Funktionen und Leistungen.

Im Rahmen des Netzmanagements dienen Tests der vorbeugenden Prüfung der Leistungsbereitschaft (*Konfidenztest*s) oder der näheren Bestimmung von Störungen und ihren Ursachen (*Diagnosetest*s). Außerdem sind Tests auch im Leistungsmanagement sinnvoll, wobei dort die Auslotung des quantitativen Leistungsspektrums im Vordergrund steht. Der Begriff *Test* kann in vier Kategorien gegliedert werden:

- *Interner Ressourcentest* – auf ein einzelnes System oder eine seiner Komponenten gerichtet,
- *Transferintegritätstest* – Test, der die generelle Konnektivität zwischen Systemen oder die Transferfunktionalität mit unterschiedlichen Anforderungen betrifft,
- *Protokollintegritätstest* – Test, der die in der Protokollspezifikation enthaltenen Prozeduren auf ordnungsgemäßen Ablauf testet,

- *Kapazitätstest* – Prüfung des Verhaltens von Systemen oder deren Komponenten in Hochlast- oder Grenzlastbereichen, oft verbunden mit der Erzeugung künstlicher Arbeitslasten.

Das allgemeine Konzept für Konfidenz- und Diagnosetests als Bestandteil des OSI-Managements geht stets davon aus, dass von einem offenen System aus Testfunktionen ausgelöst bzw. gesteuert werden, die in einem oder mehreren anderen offenen Systemen wirksam werden. Hierbei ist im Rahmen der Architektur weniger die interne Testprozedur bedeutsam, als vielmehr die Art und Weise der Initialisierung und die Rückübertragung der Testergebnisse. Dies kann von einer einheitlichen Schnittstelle zu Managementanwendungen erreicht werden, die dann für die Durchführung der Tests zuständig sind. Eine solche Schnittstelle sollte deshalb über Operationen zur Initialisierung und Steuerung von Prozeduren verfügen (AANF 4).

Weiter notwendig ist eine adäquate Visualisierung der Testergebnisse. Diese soll als Veränderung des grafischen Verhaltens der beteiligten Ressourcen abhängig von ihren Parametern und Attributen erfolgen, z.B. bei Kapazitätstests (VANF 2). Je nach Ressource soll aber auch eine einfache Anzeige des quantitativen Zahlenwertes des Attributes möglich sein.

2.1.2 Konfigurationsmanagement

Das Konfigurationsmanagement umfasst die wichtigsten Mittel zur Steuerung und Überwachung eines Netzes im normalen, störungsfreien Betrieb und hat folgende Aufgaben:

- Einschluss und Entfernung physischer und logischer Ressourcen in das Netz. Damit ist die Erzeugung und Löschung von Managementobjekten verbunden. Ressourcen können logische Ressourcen (wie z.B. Instanzen, Dienstzugangspunkte, Verbindungen) sein, aber auch Hardwarekomponenten (Endsysteme, Transitsysteme, Übertragungsmedien). Analog muss für eine Visualisierung eine Erzeugung und Löschung grafischer Repräsentationen für Managementobjekte (VANF 1) erfolgen.
- Anpassung der Konfiguration an die Nutzungs- und Betriebserfordernisse. Dazu gehört die Pflege aller durch Eingriff von außen veränderbaren Attribute von Managementobjekten, insbesondere solche, die Relationen zwischen Managementobjekten oder deren Status betreffen. Dazu gehören auch beispielsweise Routinginformationen. Ein Visualisierungssystem muss die Möglichkeit bieten, die grafische Darstellung während der Laufzeit zu verändern, da sich die Parameter und Attribute sowie die Beziehungen der visualisierten Ressourcen zueinander verändern können (IANF 3).
- Erhalt von Informationen über die Netzkonfiguration, ihre Komponenten, und deren Eigenschaften (MANF 1) sowie über die Wirkung der unter den vorgenannten Punkten genannten Aktionen.
- Explizite Anforderung einzelner Informationen (MANF 2) oder Empfang von Ergebnissen einer unaufgeforderten, sporadischen oder regelmäßigen Ereignisberichterstattung (MANF 3).

2.1.3 Abrechnungsmanagement

Das Abrechnungsmanagement hat zum Ziel, die vom Rechnernetz bereitgestellten Dienste nutzerbezogen zu erfassen und die Grundlagen dafür zu schaffen, eine verursachungsgerechte Weiterbelastung der damit verbundenen Kosten zu ermöglichen. Ob und nach welchem Algorithmus eine solche Weiterbelastung erfolgt, ist Gegenstand der *Abrechnungspolitik*. In manchen Umgebungen existieren Limitierungen für die mengenmäßige Inanspruchnahme von Diensten des Rechnernetzes. Es ist eine weitere Aufgabe des Abrechnungsmanagements, die Einhaltung dieser Limite zu überwachen.

Bei der Anmeldung eines Nutzers wird die Limitkontrolle durchgeführt. Dies geschieht auch bei der Nutzung normaler Dienste, soweit diese abrechnungspflichtig sind. Ist das Limit erreicht, wird die Nutzeranforderung zurückgewiesen. Allgemein zeigt dies eine Notwendigkeit der Prüfung von Benutzeraktionen auf ihre Gültigkeit hinsichtlich der Abrechnungspolitik (IANF 1).

Während der normalen Dienstnutzung erfolgt dabei eine Erfassung, wobei bei verschiedenen Anlässen, spätestens aber bei der Abmeldung des Nutzers, ein Satz in eine Abrechnungsdatei geschrieben wird. Zu gewissen Zeitpunkten wird die Abrechnungsdatei nach den Algorithmen der Abrechungspolitik aufbereitet. Dadurch entstehen nutzerbezogene Belastungsdateien sowie Benutzungsstatistiken. Soll auch die Auswertung der Abrechnungen über die VR-Managementplattform erfolgen, so ist es also notwendig, Belastungsdateien und Statistiken in eine Visualisierung mit einzubeziehen. Hierzu müssen geeignete grafische Repräsentationen gefunden werden, es kann aber auch zunächst wie bei den Logdateien im Fehlermanagement lediglich eine klassische textbasierte Visualisierung der Belastungsdateien verwendet werden. Die Anzeige von Statistiken bietet ebenfalls Raum für Diskussionen. Es wäre beispielsweise eine grafische Animation des Geschehens zu bestimmten Zeitpunkten denkbar. Zunächst aber soll hier – um den Umfang der Arbeit in Grenzen zu halten – auf eine weitere Diskussion verzichtet und die grafische Umsetzung dieser Problematik einer späteren ausführlichen Implementierung der Architektur überlassen werden.

2.1.4 Leistungsmanagement

Im Leistungsmanagement werden die Leistungsparameter miteinander kommunizierender offener Systeme überwacht, diesbezügliche statistische Daten definiert und gesammelt, der Überwachungsvorgang gesteuert, die gewonnenen Werte analysiert und daraus korrektive Aktionen abgeleitet. Diese können unmittelbar ausgeführt werden oder aber langfristigen Charakter haben (z.B. Konfigurationsänderungen). Hauptsächliches Zielgebiet des OSI-Leistungsmanagements sind Beziehungen zwischen Systemen und die Möglichkeiten, solche Beziehungen von entfernten Systemen aus zu überwachen und zu steuern. Insbesondere werden dabei überwacht:

- Arbeitslast (workload)
- Durchsatz durch eine Ressource (throughput)
- Wartezeit auf die Nutzung einer Ressource (waiting time)

- Fortpflanzungszeit eines Datenelements (propagation time)
- Reaktionszeit einer Transaktion (response time)
- Dienstqualität (quality of service) zwischen Verbindungsendpunkten

Die Überwachung und Messwertgewinnung kann erfolgen durch:

- das Abfragen von Einzelinformationen an den Komponenten (wie MANF 2)
- den Empfang unaufgeforderter Ereignismeldungen, sog. "event reports" (wie MANF 3)
- den Empfang ereignis- oder zeitpunktgebundener aufgeforderter Berichte (Kombination von MANF 3 und MANF 2)

Die Anzeige der erhaltenen Parameter muss dabei entsprechend ihrer Komponentenzugehörigkeit und der Abbildungsweise der Komponenten erfolgen (VANF 2).

2.1.5 Sicherheitsmanagement

Um Datensicherheit zu gewährleisten, ist ein breites Spektrum aufeinander abgestimmter Maßnahmen notwendig. Diese können von außen nach innen hierarchisch in mehreren Schichten angeordnet werden:

- Bau- und versorgungstechnische Maßnahmen
- Organisatorische Maßnahmen
- Technologische Maßnahmen
- Programm- und gerätetechnische Maßnahmen

Eine softwareseitige Integration von Sicherheitsmechanismen in das geplante System dürfte zunächst vor allem den letzten Punkt berühren. *Programm- und gerätetechnische Maßnahmen* beinhalten beispielsweise die Privilegierung der Befehlsausführung, Begrenzung der Speicherzugriffsrechte, das Prinzip der Virtualisierung von Ressourcen, Autorisierungs- und Zugriffskontrollmechanismen, kryptographische Verfahren oder Authentisierung.

Das Management der Sicherheitsmechanismen umfasst spezifische, auf einzelne Sicherheitsmechanismen bezogene Funktionen, z.B. das Einstellen von Parametern und Verteilen bzw. Auswählen von Schlüsseln bei kryptographischen Verfahren. Es erfordert eine Prüfung von Benutzeraktionen auf ihre Gültigkeit hinsichtlich der Sicherheitspolitik (erweitert IANF 1). Dadurch erfolgt eine Ausführung von Aktionen nur, wenn sicherheitspolitisch gesetzte Limits nicht überschritten werden. Außerdem müssen geeignete grafische Repräsentationen für Sicherheitsmechanismen gefunden werden. Das Management von Sicherheitsdiensten und -mechanismen kann durch eine Visualisierung der sicherheitsrelevanten Ressourcen (Dämonen, Firewalls,....) sowie deren Konfigurationsmöglichkeit durch Zugriff auf relevante Parameter geschehen.

2.2 Anforderungen an die Visualisierung

Wie in Kapitel 1.2 gezeigt, kann man VR durch Dreidimensionalität, Simulation, Interaktion, Immersion und Künstlichkeit charakterisieren. *Dreidimensionalität* fordert eine dreidimensionale Darstellung (VANF 4) der Ressourcen und deren Beziehungen zueinander. *Simulation* entsteht durch das ,,Simulationsprinzip": die Verarbeitung und Darstellung eines Modells eines Rechnernetzes als Abbild des wirklichen Netzes (VANF 5). Dieses Modell ist die Basis zur Speicherung der Repräsentation der Netzressourcen (VANF 6). Es soll von verschiedenen Managementanwendungen gespeist werden können (VANF 7).

Bestandteile des Datenmodells sind Objekte, die Informationen über ihnen zugeordnete Ressourcen tragen. *Interaktion* bedeutet, dass mit diesen Objekten interagiert werden kann. Dies kann auf verschiedene Arten erfolgen:

- Interaktion mit Ressourcen (IANF 2),
- Interaktion mit Prozessen, die sich auf mehr als ein Objekt beziehen (IANF 3),
- Interaktion mit der ganzen dargestellten Szene im Sinne einer Navigation durch das Bild (IANF 6), sowie
- Interaktion mit Operationen zur Steuerung der Managementanwendungen (IANF 4) und letzlich
- Möglichkeiten einer oberflächenintegrierten Übertragung von Eingabedaten an die Managementanwendungen (IANF 5).

Immersion besteht, wenn es Möglichkeiten gibt, immersive Hardware zur Visualisierung zu verwenden. So verfügt beispielsweise das Leibniz-Rechenzentrum (LRZ) der Bayerischen Akademie der Wissenschaften, wo der Einsatz des Systems erprobt werden soll, über eine großformatige Stereo-Projektionsanlage, eine sog. Holobench. Aus praktischen Gründen soll deshalb die Möglichkeit vorgesehen werden, die dreidimensionale Darstellung des Rechnernetzes auf verschiedenen Ausgabemedien darzustellen und verschiedenste Eingabegeräte zu verwenden (VANF 8) – also sowohl auf einer (immersiven) Holobench als auch auf einem (nicht-immersiven) PC. Hierbei ist zu beachten, dass die Holobench von einer SGI Onyx2 Workstation aus angesteuert wird, die ein anderes Betriebssystem hat, als die PCs. Deshalb ist es erforderlich, eine weitgehende Plattformunabhängigkeit der Implementierung der grafikerzeugenden Programmteile (AANF 3) zu schaffen. Die *Künstlichkeit* der VR fordert außerdem eine automatische Generierung der dargestellten Szene (VANF 9) aus den Daten, die über das Netz bekannt sind.

Die automatische Generierung der Darstellung soll sich an den in Kapitel 1.3 beschriebenen Konzepten zur Veranschaulichung orientieren. Die *abbildungstreue Veranschaulichung* kommt nur für die Ressourcen in Frage, die auch tatsächlich physisch in Erscheinung treten. Dies könnte z.B. ein dreidimensionales abbildungstreues Modell eines Routers sein. Abbildungstreue Darstellungen eignen sich zwar als Grundlage konkretisierender Darstellungen, da aber die Lage funktionaler Einheiten bei einer abbildungstreuen Veranschaulichung räumlich anders liegen kann, als bei einer konkretisierenden Veranschaulichung, könnte eine zu detaillierte grafische

Abbildungstreue zu Inkompatibilitäten der Veranschaulichungsmodelle führen. Dennoch ist es vorteilhaft, grafische Repräsentationen realistisch aussehen zu lassen, um beispielsweise bei Bedarf das entsprechende Gerät in einem Rack (Netzregal) zu finden. Deshalb soll eine grafische Abbildungstreue als Stilmittel zur näheren Ausgestaltung der grafischen Repräsentation von Ressourcen in Betracht gezogen werden. Dies bedeutet, dass Texturen und 3D-Modelle der Ressourcen möglich sein sollen (VANF 10).

Die *schematisierende Veranschaulichung* fordert die bewusste Ein- und Ausblendung oder Vergrößerung oder Verkleinerung irrelevanter Details der Abbildung. Eine einfache Art und Weise dies zu realisieren, ist die bereits geforderte Navigation in der dreidimensionalen Darstellung, bei der der Benutzer des Systems selber entscheidet, welche Objekte für ihn in der aktuellen Situation relevant sind. Er kann sich dann auf Ressourcenobjekte zu- oder von ihnen wegbewegen und dementsprechend werden diese größer oder kleiner dargestellt. Des weiteren soll eine Möglichkeit geschaffen werden, verschiedene Sichten auf Ressourcen je nach den relevanten Kennzahlen zu bieten. Dies schließt ein Ein- und Ausblenden irrelevanter Details oder der ganzen Ressource ein (VANF 20).

Die *konkretisierende Veranschaulichung* nach [ScBu 01] fordert die Verdinglichung logischer Abstraktion durch Überführung logischer Modelle in Objektform. Unter anderem tritt dadurch eine Sinnesskalierung von für den Menschen nicht direkt wahrnehmbaren Sachverhalten auf. Beim Netzmanagement eignen sich hierfür vor allem Kommunikationsarchitekturen (MANF 4) wie beispielsweise ISO/OSI oder Internet. A priori soll aber keine Wahl einer bestimmten Netztechnologie oder gar einer bestimmten Darstellungsweise erfolgen. Vielmehr sollte eine allgemeine Möglichkeit der grafischen Darstellung logischer Konzepte und der Organisation ihrer Objekte (VANF 11) gewählt werden. Abbildung 1 gibt einen Überblick über den Vorgang der Visualisierung, wie er sich mit den bisher gesammelten Anforderungen darstellt. Zunächst muss die Sammlung der zu visualisierenden Daten erfolgen. So können diese Daten zu einem Netzmodell verdichtet werden. Dieses Modell ist architektonisch in einem dreidimensionalen Raum zu organisieren. Schließlich soll die so erzeugte Sicht auf verschiedenen Medien ausgegeben werden können.

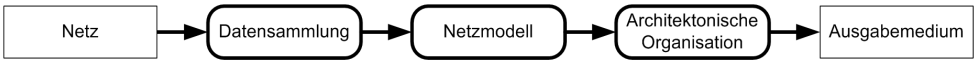


Abbildung 1: geforderte Visualisierungsphasen

2.2.1 Szenario: Visualisierung der Netztopologie

Eine Architektur zur Integration von Netzmanagement- und Netzplanungswerkzeugen muss notwendigerweise eine Visualisierung der Netztopologie unterstützen. Deshalb soll im Folgenden ein Szenario vorgestellt werden, in dem eine Visualisierung der Netztopologie diskutiert wird. Dabei wird die Sammlung der Anforderungen um zusätzliche Punkte erweitert.

Zunächst werden einführend die Begriffe Topographie und Topologie geklärt. Betrachtet man ein Rechnernetz von außen, so erschließt sich einem lediglich ein "Knäuel von Kabeln und

Geräten". Diese sind auf einer physischen Ebene der sog. *Topographie* eines Netzes angesiedelt. Dieser Begriff kennzeichnet die physikalische Struktur eines Netzes in Gestalt der Kabelführung und der eindeutigen räumlichen Plazierung seiner Komponenten. Auf dieser Ebene ist feststellbar, aus welchen Netzkomponenten ein Netz besteht und die Vernetzung der Komponenten beispielsweise mit Hilfe von Kabeln oder Glasfaserleitern ist sichtbar.

Ein großer Teil der Funktionalität eines Rechnernetzes findet sich allerdings in dem materiell weniger greifbaren Bereich logischer Abläufe. Betrachtet man nur die Hardware, so läßt sich nur sehr schwer auf die eigentlichen Vorgänge in der Software, geschweige denn auf die vollständige logische Verschaltung der Netzkomponenten schließen. Dies liegt daran, dass der Großteil des Geschehens in einem Rechnernetz auf einer logischen Ebene stattfindet, die aus den physischen Bauteilen emergiert. Dort erfolgt ein großer Teil der Strukturierung der Signale. Die Beschreibung der Struktur, die die Ressourcen des Netzes auf einer logischen Ebene zueinander einnehmen, bezeichnet man als *Topologie* des Netzes. Topographie und Topologie eines Netzes sind in den meisten Fällen nicht identisch, bestehen aber aus denselben Grundformen. Es gibt Punkt-zu-Punkt-, Bus-, Ring-, Stern-, Baum- und teil- oder vollvermaschte Maschenstrukturen. Größere Netze sind aus Kombinationen dieser Grundformen aufgebaut. Abbildung 2 stellt diese Grundformen dar.

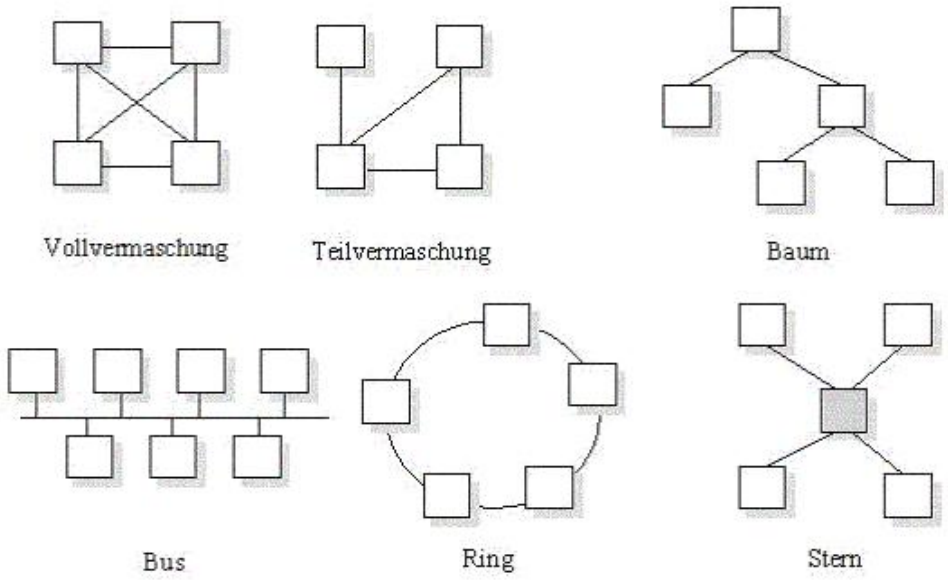


Abbildung 2: Grundstrukturen von Topologie und Topographie

Allgemein betrachtet kann man topologische Daten als Teil eines Informationsraumes auffassen. Eine Visualisierung dieses Informationsraumes erfordert eine architektonische Organisation der in ihm enthaltenen Elemente und deren Darstellung auf ein Ausgabemedium (VANF 12). Jedes Programm und jeder Prozess in einem System, jedes einzelne verarbeitete Bit, auch wenn

es zwischen Netzkomponenten ausgetauscht wird, wird von den beteiligten Rechnern in seinen Kontext eingeordnet und dementsprechend verarbeitet. Dieser Kontext ist durch die Architektur des jeweils betrachteten Systems festgelegt.

Rechnernetze betreffend liegen einer Netztopologie immer Kommunikationsarchitekturen zugrunde[12]. Das für die meisten Kommunikationsarchitekturen angewandte Strukturierungsprinzip ist das der Schnittbildung. Will man Rechnernetze grafisch darstellen, so liegt deshalb eine Visualisierung der durch die Schnittbildung entstandenen Instanzen nahe. Beispielsweise sieht das ISO/OSI-Referenzmodell sieben Schichten (Layers) zur Beschreibung von Verbindungen vor, die zwar alle über eine physische Leitung laufen, aber in ihrem logischen Kontext streng voneinander getrennt (disjunkt) sind und somit auch getrennt visualisiert werden können. In der Regel wird in Kommunikationsarchitekturen der Protokollschnitt, der Dienstschnitt und der Systemschnitt angewandt. Dienstschnitte definieren virtuelle Kommunikationssysteme; Protokollschnitte berücksichtigen deren Realisierung durch getrennte autonome Systeme (vgl. [HAN 99]:17).

Protokollschnitte definieren die Schichtprotokolle, die grafisch als Verbindungen zwischen ihren jeweiligen Endsystemen darstellbar sind (VANF 13). Die Lage dieser Verbindungen ist in der vertikalen Ebene (Höhe) durch den Dienstschnitt festgelegt und in der horizontalen Ebene durch die gegenseitige Lage der Systeme zueinander, die über den Systemschnitt spezifiziert werden kann.

Der Dienstschnitt führt zu einer funktionalen Zerlegung von Kommunikationsvorgängen. Dies führt zu einer Bildung von Funktionsschichten, dem klassischen Schichtenmodell. Zwischen und in den Schichten werden in der Regel Dienstzugangspunkte und Dienstprimitive definiert, welche die vertikale Kommunikation zwischen den Schichten regeln. Diese vertikale Kommunikation kann auf der vertikalen Koordinatenachse (Höhe) dargestellt werden. Deshalb soll folgende Festlegung erfolgen: Der Dienstschnitt definiert die Lage der grafischen Repräsentation der Dienste und Protokolle in der vertikalen, d.h. die Höhenkoordinaten (VANF 14).

Der Systemschnitt definiert die Begriffe Endsystem, Transitsystem und Übertragungsmedium. Er bestimmt die diskreten Systeme, zwischen denen kommuniziert wird. Auch die Festlegung der gegenseitigen Lage der Systeme zueinander gehört in den Bereich des Systemschnittes. Da in der Praxis beispielsweise Interfaces (beispielsweise Netzkarten) mit dem Übertragungsmedium verbunden sind, die kommunizierenden Systeme aber jeweils ihre Netzkarte enthalten, sind Relationen notwendig, um den Sachverhalt zu beschreiben.

Es lassen sich für Sachverhalte aus dem Netzmanagement zwei Arten von Relationen identifizieren: Die eine beschreibt, welches System über ein Übertragungsmedium mit welchem anderen System verknüpft ist. Sie wird Adjazenzrelation geannt (VANF 17). Eine andere Art Relation beschreibt, welche Systeme welche anderen Systeme enthalten. Sie wird Enthaltenseinsrelation genannt (VANF 18).

Zwei miteinander verbundene Router müssten also folgendermaßen beschrieben sein:

- Router R_1 -enthält- Interface I_1

[12] siehe dazu auch MANF 4

- Router R_2 -enthält- Interface I_2
- Interface I_1 -ist verbunden mit- Interface I_2

So wird es möglich, die Lage einer Ressource in der Ebene zu definieren (VANF 15).

Wie die eben beschriebenen Mechanismen zeigen, ist ein Layoutverfahren möglich, das die Verteilung der Systeme im Raum mittels Dienstschnitt, Systemschnitt und Protokollschnitt einer Kommunikationsarchitektur vornimmt.

Damit sind die Anforderungen an die Darstellung der Topologie eines Rechnernetzes abgedeckt. Mit der algemeinen Forderung einer Visualisierung beliebiger Architekturmodelle könnten darüber hinaus auch Objekte aus dem Systemmanagement wie beispielsweise Anwendungen, Prozesse und Speicher darstellbar sein. Je nach gewünschter Anwendung und Granularität ist dazu allerdings, wie auch für die Darstellung verschiedener Kommunikationsarchitekturen, die Spezifikation einer eigenen Layoutpolitik (VANF 16) notwendig.

2.2.2 Szenario: Administration und Planung von VLAN-Konfigurationen

Ein weiteres Szenario für den Einsatz eines Visualisierungssystems für Netze ist die Administration und Planung von VLAN-Konfigurationen.

Ein *VLAN* ist eine Gruppe von PCs, Servern oder anderen Netzkomponenten, die in physikalisch getrennten Segmenten positioniert sind, aber so kommunizieren, als befänden sie sich an der selben Leitung bzw. auf einem gemeinsamen physischen LAN.

Dies erleichert eine flexible Netzsegmentierung, die dessen Effizienz erhöhen und überschüssigen Verkehr verhindern kann. Beispielsweise können Benutzer und Ressourcen, die viel miteinander kommunizieren, in gemeinsame VLANs gruppiert werden. Außerdem steigt die allgemeine Performanz des Netzes durch die Limitierung von Broadcastverkehr auf einzelne VLANs. Zwischen VLANs können routerbasierte Sicherheitsmechanismen wie z.B. Firewalls darüber hinaus für eine verbesserte Sicherheit von Teilnetzen sorgen.

Da die Funktionsweise von VLAN-Switches erst 1998 mit 802.1Q von IEEE standardisiert wurde [IEEE 802.1Q], gibt es vereinzelt noch proprietäre Lösungen. Verschiedene Firmen haben schon vor der Standardisierung jeweils andere Verfahren entwickelt. Darunter sind:

- Port-basiertes VLAN (Schicht eins): Jeder physikalische Port eines Switches wird mit einer Zugriffsliste konfiguriert, die dessen Mitgliedschaft in eine Menge von VLANs spezifiziert.
- MAC-basiertes VLAN (Schicht zwei): Die Konfiguration basiert auf einer Zugriffsliste, die MAC-Adressen einzelnen VLANs zuordnet.
- Protokoll-basiertes VLAN (Schicht zwei): Die Konfiguration basiert auf dem Protokolltyp des Schicht-2 Protokollheaders.
- IP-basierendes VLAN (Schicht drei): Die Mitgliedschaft zu einem VLAN basiert auf der IP-Subnetzadresse.

- Höhere Schichten: Es ist auch möglich, VLANs auf Anwendungen oder Diensten basieren zu lassen, beispielsweise können FTP-Verkehrsflüsse dem einen und Telnet-Daten einem anderen VLAN zugeordnet werden.

IEEE 802.1Q definiert VLANs auf den Schichten eins und zwei.

Allen Arten von VLANs gemeinsam ist eine VLAN-Identifikationsnummer, die einzelne Netzkomponenten eindeutig zu VLANs zuweißt. An diesem Punkt wird eine Visualisierung von VLAN-Topologien machbar, wenn die VLAN-Identifikationsnummern der darzustellenden Netzkomponenten bekannt sind. Sie können später als Grundlage dienen, um einzelne VLANs und deren Komponenten darstellen, ausblenden oder markieren zu können. Die Zuordnung der VLAN-Identifikationsnummern als Parameter der Ressourcen kann durch Managementanwendungen erfolgen. Die bereits definierten Anforderungen VANF 1 und VANF 2 dürften dafür ausreichen.

Welche Anforderungen stellt das VLAN-Management an die Interaktion mit dem Benutzer? Nach [Poin 97]:82ff sollte ein VLAN Managementsystem u.a. folgende Aufgaben erfüllen können:

- Kreieren eines VLANs
- Modifizieren eines VLANs
- Löschen eines VLANs
- Hinzufügen von Mitgliedern eines VLANs
- Herausnehmen von Mitgliedern eines VLANs

Alle diese Aufgaben können mit Hilfe eines Benutzer-Interfaces durchgeführt werden, welche die Durchführung der beiden letztgenannten Aufgaben ermöglicht. Zum Kreieren eines bestimmten VLANs muss eine Komponente nur einem bisher nicht vorhandenen VLAN hinzugefügt werden. Das bedeutet, man stellt an der Repräsentation des Switches das gewünschte VLAN ein, und ordnet diesem eine Endstation zu. Zum Modifizieren genügt das Hinzufügen oder Herausnehmen einer Endstation oder die Änderung der Subnetznummer am Switch, und gelöscht werden kann ein VLAN, indem die letzte Endstation aus dem VLAN entfernt wird. Letztendlich können mit einer einfachen Änderung der Zugehörigkeit der betrachteten Switchrepräsentation zu einem VLAN alle Aufgaben erfüllt werden.

Die Konfigurationsdaten der VLANs sollten deshalb mit Hilfe einer direkten Interaktion mit der Repräsentation der Ressource (IANF 2) verändert werden können: Die Änderung der VLAN-Zugehörigkeit der Switchrepräsentation wird dann zu einer Änderung der Attribute des Managementobjekts[13] und damit zu einer Änderung der Repräsentation führen.

Für die Darstellung der VLANs sind ferner verschiedene Sichten möglich:

- (Gesamt-) Sicht: Zeigt alle im System vorhandenen VLANs. Es werden alle logischen, VLAN betreffenden Ressourcen angezeigt. Alle Ressourcen, die VLAN-unabhängig sind, sollen ausgeblendet werden können (wie VANF 20).

[13] zu einer Definition von Managementobjekten siehe Kapitel 3.4

- VLAN-spezifische Sicht: ein einzelnes VLAN wird betrachtet. Seine aktuellen Mitglieder werden farbig markiert (Endstationen, Server). Zur Auswahl des betrachteten VLANs bietet sich die Interaktion mit einem in diesem VLAN enthaltenen Switch oder einem extra für diese Aufgabe erzeugten Objektes, das eine globale Auswahl zur Verfügung stellt, an (wie IANF 2). Die Markierung kann durch Veränderung der visuellen Parameter der Darstellung der betroffenen Ressourcen erreicht werden (wie VANF 2).

Bei [Poin 97] wird weiter in physische und logische Komponenten unterteilt. Eine solche Unterteilung wurde im vorliegenden Fall bereits im Abschnitt 2.2.1 zur Topologie betrachtet, es müssen deshalb diesbezüglich keine weiteren Anforderungen definiert werden.

2.2.3 Szenario: Visualisierung von Verkehrsflüssen

Eine weitere Anforderung an das System ist die Visualisierung von Verkehrsflüssen.

Verkehrsflüsse sind Flüsse von Protokolldateneinheiten (*Protocol Data Units*, PDUs) von Endsystem zu Endsystem (durch Transitsysteme oder auch direkt).

Dies ist in allen Schichten des ISO/OSI Modells möglich, so gibt es beispielsweise SMTP-, FTP-, HTTP-, IP-, UDP-, oder TCP-PDUs.

Wie in Kapitel 2.2.1 bereits diskutiert und gefordert, soll mit Hilfe des Dienstschnittes und des Systemschnittes die räumliche Lage eines Schichtprotokolls und damit eines Verkehrsflusses im Projektionsraum der Visualisierung festgelegt werden. Die vertikale Lage der Protokolle ist dabei durch den Dienstschnitt festgelegt, die horizontale Lage der End- und Transitsysteme durch den Systemschnitt. Bei der Darstellung der Protokolle als Verbindungen zwischen Dienstinstanzen sollen Unicast (eine definierte Quelle - ein definierter Empfänger) und Multicast (eine definierte Quelle - viele definierte Empfänger) möglich sein (VANF 19). Die Darstellung von Broadcasts (eine definierte Quelle - viele undefinierte Empfänger) stellt sich als schwieriger dar, da bei Broadcasts keine definierten Endpunkte identifiziert werden können. Es wird vorgeschlagen, dass eine Anwendung, die einen Broadcast visualisieren möchte, diesen mit Hilfe zusätzlicher Informationen über das Netz zunächst auf einzelne Unicasts aufteilt.

Liefern die Managementobjekte zusätzlich Fakten aus dem Leistungsmanagement, wie Durchsatz, Bandbreite, etc. so kann die grafische Darstellung des Verkehrsflusses abhängig von den erhaltenen Parametern und Kennzahlen geändert werden. Ist der Verkehrsfluss beispielsweise durch eine Linie festgelegt, so kann die Dicke oder die Farbe der Linie geändert werden. Es ist auch vorstellbar, als Repräsentation eines Verkehrsflusses beispielsweise einen Scatterplot des Kennzahlenverlaufs oder eine Animation, bei der ein Paket vom Ausgangs- zum Zielsystem wandert, zu verwenden. Letztlich ist immer eine Abbildung der Parameter und Kennzahlen auf die grafische Darstellung der Verkehrsflüsse notwendig, die entsprechende Anforderung VANF 2 wurde bereits definiert.

Weiterhin bleibt zu klären, wie Interaktionen mit der Visualisierung der Verkehrsflüsse stattfinden sollen. Dies kann prinzipiell mit Hilfe von zwei Mechanismen erfolgen, deren Anforderungen bereits in Kapitel II.1 spezifiziert wurden:

- Selektion an den Komponenten

  Damit kann für einen Verkehrsfluss die Visualisierung des durchgehenden Verkehrs an- oder abgeschaltet und die Art des visualisierten Verkehrs ausgewählt werden (wie IANF 2).

- Komponentenübergreifende Selektion

  Über Funktionen kann die Verkehrsflussvisualisierung für mehrere Komponenten ausgewählt werden. Das ermöglicht es, nach bestimmten Kriterien zu visualisieren, z.B. können damit die Top 5 der Komponenten mit dem größten Verkehrsaufkommen angezeigt werden usw.(wie IANF 4 und IANF 3)

Die letzten beiden Szenarien konnten bestätigen, dass die gesammelten Anforderungen aus dem Netzmanagement, der Visualisierung und dem ersten Szenario bereits so umfassend sind, dass sie die Anforderungen der letzten beiden Szenarien mit abdecken.

2.3 Anforderungskatalog

Zusammengefasst ergibt sich folgender Anforderungskatalog:

- Architekturanforderungen (AANF):
  1. Es wird eine modulare, offene und datenintegrierte Managementplattform als Trägersystem für die Managementanwendungen benötigt.
  2. Die mit dem geforderten Visualisierungssystem arbeitenden Managementwerkzeuge sollen oberflächenintegriert sein. Dies bedeutet, dass die Präsentation der ausgegebenen Information homogen erfolgen und die Bedienung der Werkzeuge und Bedienung einheitlich sein soll.
  3. Die Implementierung des Benutzer-Interfaces soll plattformunabhängig erfolgen.
  4. Es wird eine einheitliche Schnittstelle von den zu integrierenden Managementanwendungen zum Visualisierungssystem benötigt. Diese soll die Initialisierung und Steuerung der Anwendungen ermöglichen.
- Managementanforderungen (MANF):
  1. Das Visualisierungssystem muss Informationen über die Netzkonfiguration, ihre Komponenten, und deren Eigenschaften erhalten können.
  2. Es müssen Einzelinformationen zu den Ressourcen angefordert werden können.
  3. Es müssen unaufgeforderte Ereignismeldungen von den Ressourcen empfangen werden können.
  4. Als grundlegendes Konzept des Aufbaus von Rechnernetzen sollen Kommunikationsarchitekturen berücksichtigt werden.
- Visualisierungsanforderungen (VANF):
  1. Physische und logische Ressourcen eines Netzes sollen durch die Erzeugung und Löschung ihrer grafischer Repräsentationen visualisiert werden.
  2. Das grafische Verhalten der Repräsentation beteiligter Ressourcen soll sich abhängig von deren Parametern und Attributen ändern.
  3. Parameter, Attribute und Beziehungen der visualisierten Ressourcen sollen sich während der Laufzeit ändern können. Es ist deshalb eine stets aktuelle Visualisierung notwendig.
  4. Im Allgemeinen soll die Darstellung aller Komponenten dreidimensional erfolgen.
  5. Die Verarbeitung und Darstellung soll auf einem Modell eines Rechnernetzes als Abbild des wirklichen Netzes erfolgen.
  6. Das Datenmodell ist eine Repräsentation des tatsächlichen Netzes und die Basis zur Speicherung der Repräsentation der Netzressourcen.

7. Das Datenmodell soll von verschiedenen Managementanwendungen gespeist werden können.
8. Medienunabhängigkeit: Es soll eine Möglichkeit der Ein/Ausgabe auf verschiedenste Medien bestehen.
9. Die dargestellte Szene soll automatisch generiert werden.
10. Die Repräsentationen der Ressourcen sollen Texturen und 3D-Modelle ermöglichen.
11. Es soll eine allgemeine Möglichkeit der grafischen Darstellung logischer Konzepte und der Organisation ihrer Objekte bestehen.
12. Eine Visualisierung des Informationsraumes erfordert seine architektonische Organisation.
13. Die Schichtprotokolle sollen grafisch als Verbindungen zwischen ihren jeweiligen Endsystemen dargestellt werden können.
14. Der Dienstschnitt definiert hierbei die Lage der grafischen Repräsentation der Dienste und Protokolle in der vertikalen, d.h. deren Höhenkoordinaten.
15. Über den Systemschnitt soll die Lage einer Ressource in der horizontalen Ebene bestimmt werden.
16. Es soll möglich sein, auch andere Layoutverfahren, als in VANF 13 bis VANF 15 beschrieben, zu spezifizieren.
17. Eine Menge von Relationen soll beschreiben, welches System über ein Übertragungsmedium mit welchem anderen System verknüpft ist (Adjazenzrelation).
18. Eine Menge von Relationen soll beschreiben, welches System in einem anderen enthalten ist (Enthaltenseinsrelation).
19. Bei Verkehrsflüssen soll eine Darstellung von Unicast (eine Quelle-ein Empfänger) und Multicast (eine Quelle-viele Empfänger) möglich sein.
20. Ganze Repräsentationen und einzelne grafische Details sollen ein- und ausgeblendet werden können.

- Interaktionsanforderungen (IANF):
  1. Es soll eine Prüfung von Benutzeraktionen auf ihre Gültigkeit hinsichtlich der Abrechnungs- und Sicherheitspolitiken erfolgen.
  2. Es soll eine Interaktion mit einzelnen Ressourcen möglich sein. Diese soll durch die Interaktion mit deren Rohdaten und grafischen Repräsentationen und erfolgen.
  3. Ebenso soll eine Interaktion mit Prozessen, die sich auf mehr als ein Managementobjekt beziehen stattfinden können. Solche Prozesse sind entweder Managemantwendungen oder einzelne Repräsentationen von Ressourcen, die andere Repräsentationen verändern können.

4. Zur Steuerung der Managementanwendungen sollen deren Funktionen von der Benutzeroberfläche aus gestartet werden können.
5. Zur Steuerung der Managementanwendungen benötigen diese ferner verschiedene Möglichkeiten zum Erhalt von Benutzereingaben. Die Eingabe soll deshalb ebenfalls oberflächenintegriert erfolgen können.
6. Eine Navigation durch das Bild im Sinne einer Interaktion mit der ganzen dargestellten Szene ist notwendig.

3 Architekturdesign des Rahmenwerks

In diesem Abschnitt wird ein Rahmenwerk vorgestellt, dass VR-Visualisierungen in Managementplattformen integriert und es ermöglicht, beliebige Managementanwendungen auf einfache Art und Weise in VR-Umgebungen einzubinden. Zielvorgabe ist hierbei die Erfüllung der Anforderungen aus Kapitel 2.3.

Kapitel 3.1 ordnet das Thema kurz in bestehende Zusammenhänge ein und gibt Hinweise auf den aktuellen Stand der Forschung.

Um die beiden Themenkreise Visualisierung und Management vereinen zu können, wird dann zunächst in Kapitel 3.2 ein allgemeines Visualisierungsmodell vorgestellt und es werden die Grundlagen der Informationsvisualisierung besprochen.

Kapitel 3.3 integriert diese grundlegenden Mechanismen schließlich in ein allgemeines Modell von Managementplattformen. Hier wird der Begriff der Visualisierungsanwendung eingeführt, und es wird der für eine Visualisierung notwendige Datenfluss innerhalb der Plattform aufgezeigt.

Die darauffolgenden Abschnitte 3.4 bis 3.6 zeigen detailiert auf, wie die einzelnen Komponenten des Systems beschaffen sein müssen, und welche Zusammenhänge zwischen ihnen bestehen. In mehreren Schichten werden Objekte und grundlegende Dienste definiert, deren Zusammenspiel schließlich eine dynamische dreidimensionale Visualisierung der Managementinformation erlaubt.

Daraufhin wird in Kapitel 3.7 diskutiert, wie mit der Visualisierung, den zugrundeliegenden Daten und den Managementanwendungen interagiert werden kann, und wie dies oberflächenintegriert geschehen kann.

Abschließend gibt 3.8 einen Überblick über die Umsetzung der Anforderungen aus Kapitel 2 und diskutiert die Vor- und Nachteile des Designs.

3.1 Bestehende Ansätze

Erste Ansätze zur Visualisierung von Netzstrukturen gehen zurück bis 1981. So legte Bertin in [Bert 81] wichtige Grundlagen, indem er grafische Repräsentationen für Knoten-, Kanten- und Matrizen verwendete. 1988 wurden mit SEMNET erste Erfolge mit einer dreidimensionalen Darstellung von großen Wissensbasen eines semantischen Netzes erzielt [FPF 88].

Erst ab etwa 1992 wurde in ersten Experimenten die Anwendungsmöglichkeit auf Kommunikationsnetze untersucht. In [CLFZ 93] konnten zu dieser Zeit dreidimensionale Darstellungen zur Visualisierung von großen Breitbandnetzwerken eingesetzt werden. Es fand zunächst nur eine Visualisierung der Struktur eines Netzes statt, erst 1995 konzentrierten sich Becker, Eick und Wilks in [BEW 95] mit dem SeeNet System auf die Visualisierung von Datenflüssen durch Netze.

Neuere Arbeiten konzentrieren sich auf die Visualisierung großer dynamischer Datensätze aus der Telekommunikation [KNK 99] und die Verwendung zum Netzmanagement. Wichtige Vertreter sind die Systeme VENoM [CuEg 98] und Cybernet [AGL$^+$ 00], die aktuelle Daten aus dem Netz dyamisch verarbeiten können. Eine allgemeine Integration aus der Sicht des Netzmanagements fand allerdings bisher nicht statt.

3.2 Visualisierungsmodell

Die Visualisierung von Managementressourcen erfordert einen Prozess der schrittweisen Umsetzung der Rohdaten in ein sichtbares Bild. Dieser Prozess ist unabhängig von der Art der Rohdaten. Ob es sich dabei um den Inhalt einer Datenbank, eine Sammlung von Wetterdaten oder um ein Rechnernetz handelt, der grundlegende Prozess der Visualisierung bleibt immer derselbe.

Mit diesem Themenkreis beschäftigt sich das Fachgebiet der *Informationsvisualisierung*, deren fachlicher Gegenstand von [CMS 98] als

> "the use of computer-supported, interactive, visual representations of abstract data to amplify cognition".

definiert wird.

Um einen klaren Rahmen für die weitere Diskussion und ein modulares Softwaredesign zu schaffen, werden nun grundlegende Mechanismen zur Informationsvisualisierung und eine Terminologie und eine mathematische Formalisierung vorgestellt.

3.2.1 Teilschritte der Visualisierung

Die Rohdaten werden durch einen Prozess der schrittweisen Umsetzung in ein sichtbares Bild umgewandelt. Hierzu kann eine sogenannte ,,Visualisierungspipeline" verwendet werden.

Definition 3.1 *Eine Visualisierungspipeline unterteilt den Prozess der Visualisierung in konzeptionelle Teilbereiche. Das Ziel ist dabei, die Information aus den Rohdaten in ein Format umzusetzen, das vom menschlichen Wahrnehmungssystem verstanden werden kann. Gleichzeitig soll aber die Integrität der ursprünglichen Information erhalten bleiben.*

[HaMc 90] beschreibt das sogenannte filter-map-render Referenzmodell, das eine Visualisierungspipeline mit drei Transformationen vorsieht. Diese Transformationen treten in den meisten Visualisierungsprozessen auf.

- Die erste Transformation ist eine Datenanreicherung bzw. Datenverbesserung (data enrichment bzw. data enhancement). Sie operiert direkt auf den Rohdaten. Aus den Rohdaten werden die für die anschließenden Visualisierungsoperationen notwendigen Sachverhalte abgeleitet.
- Die nächste Transformation ist die Visualisierungsabbildung (visualisation mapping). Sie konstruiert ein imaginäres Objekt, das als *abstraktes Visualisierungsobjekt* (abstract visualisation object, AVO) bezeichnet wird. Seine Konstruktion erfolgt aus den von der ersten Transformation abgeleiteten Daten. Ein AVO ist ein imaginäres Objekt mit einer Ausdehnung in Raum und Zeit. Es besteht aus einem Attributfeld, in das die Simulationsdaten abgebildet werden. Diese Attribute können beispielsweise die Geometrie, die Uhrzeit, die Farbe, die Transparenz und die Oberflächentextur enthalten. Um die Abbildung zwischen den Daten und den AVO-Attributen durchführen zu können, existieren

Transferfunktionen, die einfache Abbildungen zwischen Daten und AVO-Attributen definieren.

- Die letzte Transformation ist das Rendering, d.h. die Generierung des fertigen Bildes aus den bekannten Eckpunkten für die darzustellenden Objekte. Sie erfolgt durch die mathematische Berechnung imaginärer Lichtstrahlen, die auf die Formen fallen. Das Rendering operiert dabei auf den AVOs, die bereits alle für das Rendering notwendigen Daten enthalten. Typische Renderingoperationen enthalten außerdem aus der Computergrafik bekannte Sichttransformationen wie Rotation oder Translation (Verschiebung) und Algorithmen wie Clipping (Abschneiden von Kanten) oder Anti-Aliasing (Weichzeichnen der Linien).

Neben dem filter-map-render Referenzmodell gibt es verschiedene weitere Modelle für solche Pipelines. Das umfassendste ist derzeit wohl das Data State Reference Model, das in [Chi 00] beschrieben wird. Es basiert auf einer ähnlichen funktionellen Unterteilung, nimmt allerdings einen generalisierteren Blickwinkel als das filter-map-render-Modell ein. Die Pipeline, die ein Objekt bis zu seiner Darstellung durchläuft, wird dabei in vier Stufen eingeteilt:

- Value - Die Rohdaten
- Analytische Abstraktion (analytical abstraction) - Daten über die Daten, Metadaten
- Darstellungsabstraktion (Visualization abstraction) - mit Hilfe von Visualisierungstechniken präsentierbare Information
- View - Das Endprodukt der Visualisierungsabbildung, vom Benutzer sichtbar.

Die Transformation von Daten von einer Stufe zur nächsten erfolgt durch drei Operatoren:

- Datentransformation (Data Transformation) - generiert eine analytische Abstraktion vom Eingabewert.
- Visualisierungstransformation (Visualization Transformation) - reduziert eine analytische Abstraktion in eine Visualisierungsabstraktion (erzeugt praktisch ein AVO), die darstellbaren Inhalt repräsentiert.
- Visuelle Abbildung/Präsentation (Visual Mapping Transformation) - stellt die in darstellbarem Format vorliegende Information dar.

Außerdem kann es in jeder Stufe selbst auch Operatoren geben.

Abbildung 3 zeigt die hier verwendete Pipeline, die Bezeichnungen sind dem Data-State Referenzmodell und der in den nächsten beiden Kapiteln beschriebenen Terminologie nach [Wüns 98] entnommen. MO bezeichnet ein Managementobjekt[14] und IO ein Informationsobjekt[15] mit verbesserten, für das System lesbar formatierten Daten. PO bezeichnet ein Präsentationsobjekt[16], das Pendant zum AVO im filter-map-render Modell. Die Wolken bezeichnen Mengen und die Pfeile einen sequentiellen Ablauf.

[14] siehe Kapitel 3.4
[15] siehe Kapitel 3.2.2
[16] siehe Kapitel 3.2.3

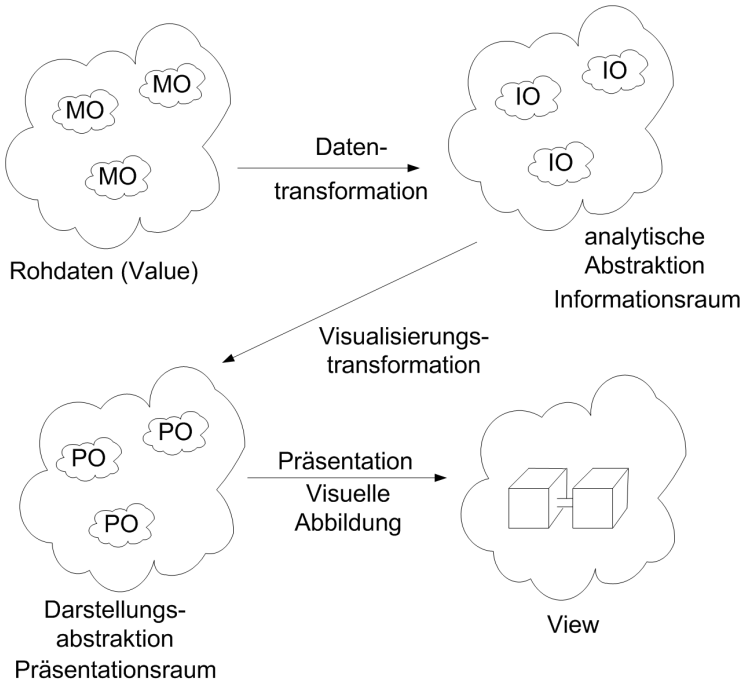


Abbildung 3: Die Visualisierungspipeline nach [Chi 00] und [Wüns 98]

3.2.2 Informationsraum

In [Wüns 98] werden einige Definitionen vorgeschlagen, die eine einheitliche Terminologie sowie eine mathematische Formalisierung von Visualisierungsvorgängen ermöglichen. Zunächst wird der Begriff des Informationsraumes eingeführt, der die in den Rohdaten enthaltene Information greifbar macht. Hierzu ist die Unterteilung des Raumes in eine diskrete Menge aus informationstragenden Objekten und die Festlegung deren Struktur notwendig. Diese Einordnung und Kapselung kann aus der Sicht des Data-State-Reference-Modells als Datentransformation angesehen werden. Auf diese Art und Weise wird die analytische Abstraktion von den Rohdaten erreicht. Die informationstragenden Objekte werden Informationsobjekte genannt:

Definition 3.2 *Unter einem Informationsobjekt IO versteht man die durch die Struktur eines informationsverarbeitenden Systems notwendige Kapselung von informationsrepräsentierenden Daten. Ein Informationsobjekt ist ein konkretes Objekt, das in einem anderen Informationsobjekt enthalten sein kann.*

Alle im Informationsraum enthaltenen informationstragenden Objekte bilden die Informationsmenge:

Definition 3.3 *Die Informationsmenge $\mathbb{IM}$ ist eine diskrete Menge von Informationsobjekten:*

$$\mathbb{IM} = \{IO_1, IO_2, \ldots, IO_n\} \quad \forall i \nexists j : IO_i = IO_j \quad mit\, i \neq j, \quad n, i, j \in \mathbb{N} \tag{1}$$

Die informationstragenden Objekte im Informationsraum stehen zueinander in Beziehung. Die Art der Beziehung und die beteiligten Objekte werden von der Informationsstruktur beschrieben:

Definition 3.4 *Eine Informationsstruktur $\mathbb{IS}$ ist eine Relation, die die Beziehung zwischen den Informationsobjekten IO beschreibt:*

$$\mathbb{IS} \subseteq \mathbb{IM} \times \mathbb{IM} \tag{2}$$

Informationsobjekte und Informationsstruktur bilden gemeinsam den Informationsraum:

Definition 3.5 *Ein Informationsraum $\mathbb{IR}$ wird durch eine Informationsmenge $\mathbb{IM}$ und eine Informationsstruktur $\mathbb{IS}$ definiert:*

$$\mathbb{IR} = \{\mathbb{IM}, \mathbb{IS}\} \quad mit |\mathbb{IM}| \geq 1 \quad und \quad |\mathbb{IS}| \geq 0 \tag{3}$$

Jedes Informationsobjekt verfügt außerdem über eine Menge von Attributen, die seine Information enthalten.

Definition 3.6 *Die Funktion attr liefert alle Attribute der ihr übergebenen Objekte.*

$$\begin{aligned} attr(\{Objekt_1, Objekt_2, \ldots, Objekt_k\}) = \{A_1, A_2, \ldots, A_n\} \\ \forall i \nexists j : A_i = A_j \quad mit \quad i \neq j, \quad i, j, k, n \in \mathbb{N} \end{aligned} \tag{4}$$

Die Attribute aller Informationsobjekte einer Menge bilden deren Attributmenge, die die enthaltene Information in der Menge darstellt.

Definition 3.7 *Eine Attributmenge $\mathbb{AM}$ ist die Menge aller Attribute A_i einer Menge von Objekten.*

$$\mathbb{AM} = attr(\{Objekt_1, Objekt_2, \ldots, Objekt_k\}) \quad k \in \mathbb{N} \tag{5}$$

Entsprechend ist die Attributmenge einer Informationsmenge die Menge aller Attribute aller Objekte der Informationsmenge.

Definition 3.8 *Eine Attributmenge $\mathbb{AM}_{\mathbb{IM}}$ ist die Menge aller Attribute A_i einer Informationsmenge $\mathbb{IM}$.*

$$\mathbb{AM}_{\mathbb{IM}} = attr(\mathbb{IM}) \tag{6}$$

Da die in einem Informationsraum enthaltene Information in den Attributen seiner Objekte enthalten ist, bestimmen diese auch seine Dimension.

Definition 3.9 *Die Dimension eines Informationsraumes* $\mathbb{IR}$ *ist die Mächtigkeit der Attributmenge* $\mathbb{AM}_{\mathbb{IM}}$.

$$dim(\mathbb{IR}) = |\mathbb{AM}_{\mathbb{IM}}| \tag{7}$$

Mit dem vorgestellten formalen Gerüst konnte ein Raum definiert werden, der die darzustellende Information enthält. Durch die Kapselung in Objekte wurde ferner eine Vorsortierung der Information in überschaubare Einheiten erreicht. Dies realisiert die Anforderungen VANF 5 und VANF 6.

Wie stellt man nun diese Information dar?

3.2.3 Präsentationsraum

Wie die im Kapitel 3.2.1 dargestellten Modelle aufzeigen, muss jedes Informationsobjekt im Informationsraum der Rohdaten über ein korrespondierendes abstraktes Visualisierungsobjekt bzw. Daten in der Stufe der Darstellungsabstraktion verfügen. Für diese Art von Objekten wird im Folgenden entsprechend der Terminologie in [Wüns 98] der Begriff des Präsentationsobjekts (PO) verwendet. Präsentationsobjekte stellen aus der Sicht des Data-State-Reference-Modells die Darstellungsabstraktion dar. Sie bilden den sogenannten Präsentationsraum und beschreiben, wie die in den Informationsobjekten enthaltene Information grafisch dargestellt werden soll. Die Präsentationsobjekte sind folgendermaßen definiert:

Definition 3.10 *Ein Präsentationssobjekt PO ist eine logische grafische Einheit im Prozess der Präsentation.*

Definition 3.11 *Die Präsentationsmenge* $\mathbb{PM}$ *ist eine diskrete Menge von Präsentationssobjekten:*

$$\mathbb{PM} = \{PO_1, PO_2, \ldots, PO_n\} \quad \forall i \nexists j : PO_i = PO_j \quad mit\, i \neq j, \quad n, i, j \in \mathbb{N} \tag{8}$$

Mit diesen Definitionen kann nun auch der resultierende Präsentationsraum definiert werden:

Definition 3.12 *Ein Präsentationsraum* $\mathbb{PR}$ *wird durch eine Präsentationsmenge* $\mathbb{PM}$ *und eine Menge* $\mathbb{F}$ *von Funktionen gebildet.*

$$\mathbb{PR} = \{\mathbb{PM}, \mathbb{F}\} \quad mit |\mathbb{PM}| \geq 1 \quad und \quad |\mathbb{F}| \geq 0, \quad \mathbb{F} = \{F_1, F_2, ..., F_n\}, \quad n \in \mathbb{N} \tag{9}$$

Auch Präsentationsobjekte verfügen über Attribute, die allerdings nicht die in den korrespondierenden Daten enthaltene Information beschreiben, sondern vielmehr deren grafische Repräsentation.

Definition 3.13 *Eine Attributmenge $\mathbb{AM}_{\mathbb{PM}}$ ist die Menge aller Attribute A_i einer Präsentationsmenge $\mathbb{PM}$.*

$$\mathbb{AM}_{\mathbb{PM}} = attr(\mathbb{PM}) \tag{10}$$

Analog zur Dimension des Informationsraumes ist die des Präsentationsraumes definiert:

Definition 3.14 *Die Dimension eines Präsentationsraumes $\mathbb{PR}$ ist die Mächtigkeit der Attributmenge $\mathbb{AM}_{\mathbb{PM}}$.*

$$dim(\mathbb{PR}) = |\mathbb{AM}_{\mathbb{PM}}| \tag{11}$$

Die Semantik im Präsentationsraum wird durch eine Menge von Funktionen $\mathbb{F}$ geregelt, die durch Programme realisiert werden können. Damit liegt eine recht allgemeingültige Definition vor, da mit den Funktionen und Attributen der PO_i ganz verschiedene Anwendungsfälle realisiert werden können. Es ist ferner mit Hilfe der Funktionen möglich, die gewünschte grafische Struktur der Darstellung durch Anpassung der Attribute der Präsentationsmenge $\mathbb{PM}$ zu verändern. Dadurch können beispielsweise die Lage, die Ausrichtung im Raum sowie eine eventuell gewünschte Überlappung der PO_i abhängig von der Struktur des Informationsraumes $\mathbb{IR}$ verändert werden. Auch eine Interaktion mit der grafischen Darstellung kann über diese Funktionen durchgeführt werden (siehe Kapitel 3.7).

3.2.4 Abbildungsfunktion

Die in diesem Kapitel beschriebenen Grundlagen ermöglichen es nach [Wüns 98] nun, die Funktionalität einer Visualisierungspipeline von den Rohdaten zur Darstellungsabstraktion durch eine Abbildung zu formalisieren. Sind die Parameter bekannt, kann eine automatische Generierung der Abbildung erfolgen (VANF 9).

Definition 3.15 *Die Informationsvisualisierung IVis ist eine Funktion, die einen Informationsraum $\mathbb{IR}$ in einen Präsentationsraum $\mathbb{PR}$ abbildet. Die der Funktion IVis übergebenen Parameter sind:*
$\mathbb{TIM}$*: ausgewählte Informationsobjekte*
$\mathbb{IS}$*: Informationsstruktur*
$\mathbb{TAM}_{\mathbb{TIM}}$*: ausgewählte Attributmenge von* $\mathbb{TIM}$
$Control$*: Steuergrößen*

$$
\begin{aligned}
IVis(\mathbb{TIM}, \mathbb{IS}, \mathbb{TAM}_{\mathbb{TIM}}, Control) = & \mathbb{PR} \\
& \mathbb{TIM} \in \wp\mathbb{IM} \\
& \mathbb{AM}_{\mathbb{TIM}} = attr(\mathbb{TIM}) \\
& \mathbb{TAM}_{\mathbb{TIM}} \in \wp\mathbb{AM}_{\mathbb{TIM}} \\
& \mathbb{IR} = \{\mathbb{TIM}, \mathbb{IS}\}
\end{aligned} \tag{12}
$$

(12) ist nach [Wüns 98] eine Eins-zu-Eins-Abbildung auf $\mathbb{PR}$. Diese Abbildung ist linksvollständig, rechtsvollständig und umkehrbar eindeutig.

Zur Verdeutlichung dieses Prozesses kann man sich das folgende Szenario vorstellen:

Gegeben seien drei Endsysteme $(IO_1, \ldots, IO_3)$, die über Kabel mit einem Switch IO_4 verbunden sind.
Ohne detailiert auf den mathematischen Formalismus einzugehen, kann man sich den Prozess der Informationsvisualisierung folgendermaßen veranschaulichen:

1. Datentransformation: Es erfolgt die Bildung eines empirischen Modelles in Form des Informationsraumes $\mathbb{IR}$. Dieser besteht im vorliegenden Fall aus den Informationsobjekten $(IO_1, \ldots, IO_4)$, d.h. aus den Datenobjekten, die die Endsysteme und Switches beschreiben, und aus der Informationsstruktur $\mathbb{IS} = \{(IO_4, IO_1), (IO_4, IO_2), (IO_4, IO_3)\}$, um die Verbindungskonstellation zwischen den Objekten zu beschreiben. Der Switch IO_4 steht also mit allen anderen Objekten aus der Menge, den Endsystemen, in Beziehung. Er ist mit ihnen verbunden.
2. Visualisierungstransformation: Durch die Anwendung der Funktion IVis erfolgt die Bildung eines Präsentationsraumes $\mathbb{PR}$. Dies bedeutet in der Praxis, dass ein Mapping vom Informationsobjekt (als Repräsentation der Daten) auf ein bestimmtes Präsentationsobjekt (das die gewünschte grafische Metapher darstellt) stattfindet. Die Funktionen $\mathbb{F}$ im Präsentationsraum sorgen ferner für eine Anpassung der Präsentationsobjekte an die gewünschte Darstellungsweise (beispielsweise durch Layout). So wird der Präsentationsraum $\mathbb{PR}$ generiert, der nun bereits die notwendige grafische Beschreibung der einzelnen Objekte sowie deren Verteilung im Raum enthält.
3. Visuelle Abbildung: $\mathbb{PR}$ wird nun auf dem Medium ausgegeben, eine Darstellung des Netzes wird sichtbar.

Um diese Prinzipien für das Netzmanagement anwenden zu können, ist es nun notwendig, dass diese in allgemeine Konzepte des Netzmanagement integriert werden. Dadurch wird sichergestellt, dass das resultierende System hohen Anforderungen an Erweiterbarkeit und Anpassbarkeit genügt und für eine Vielzahl von Fragestellungen aus dem Netzmanagements einsetzbar ist.

3.3 Integration in Netzmanagementplattformen

Die in Kapitel 3.2 gesammelten Grundlagen über Visualisierungsvorgänge können nun zur Visualisierung von Rechnernetzen angewandt werden. Zunächst erfolgt eine Einordnung der Problematik in den Rahmen bestehender Managementkonzepte.

Wie in AANF 1 gefordert, soll die Architektur auf eine Managementplattform aufsetzen. In der Regel folgen Managementplattformen einem funktionalen Aufbau, wie er in Abbildung 4 nach [HAN 99] dargestellt ist. Zentraler Baustein ist die Infrastruktur der Plattform, die ein Kernsystem zur Verfügung stellt. Dieses besteht aus einer Informationsverwaltung mit Datenbank und einem Kommunikationsbaustein. Der Kommunikationsbaustein stellt Methoden zum Zugriff auf Netz- und Systemressourcen zur Verfügung. Die Managementanwendungen auf der Plattform lassen sich aufteilen in Managementapplikationen, Basisanwendungen und Entwicklungswerkzeuge. Alle Managementanwendungen werden über ein API an das Kernsystem angebunden. Externe Werkzeuge, Benutzer und Entwickler können über einen Oberflächenbaustein bzw. ein weiteres API auf die Anwendungen zugreifen. Da alle Managementanwendungen und die für das Netzmanagement wichtigen Daten über die Infrastruktur erreichbar sind, sind nur der Oberflächenbaustein und der Bereich der Managementapplikationen für das Visualisierungssystem relevant. Sie sind im Bild schraffiert dargestellt.

Der Oberflächenbaustein realisiert die Oberflächenintegration (AANF 2), indem er eine allgemeine Schnittstelle zwischen dem Oberflächenprozess und den Managementanwendungen bereitstellt (vgl. [HAN 99]).

Hier kann der letzte Abschnitt einer Visualisierungspipeline positioniert werden, also das Rendering bzw. die visuelle Abbildung von $\mathbb{PR}$ auf die eingesetzten Medien. Eine solche Einordnung realisiert die Forderungen nach der Medienunabhängigkeit (VANF 8) und ermöglicht eine Plattformunabhängigkeit der grafikerzeugenden Programmteile (AANF 3).

Die Positionierung des von der visuellen Abbildung erzeugten Views im Oberflächenprozess resultiert in einer benutzerspezifischen Sicht auf den Präsentationsraum vor allem im Hinblick auf:

- seine Navigation durch den Präsentationsraum. Diese erfolgt im Benutzer-Interface und ändert die dem Raum zugrundeliegenden Daten selbst nicht (realisiert IANF 6). Dadurch werden außerdem mehrere synchron auf demselben Präsentationsraum $\mathbb{PR}$ arbeitende Benutzer möglich.
- seine Sicht auf den Raum. Durch die Trennung von Oberflächenbaustein und Visualisierungspipeline können außerdem bestimmten Benutzern nur bestimmte Sichten auf einen Präsentationsraum angeboten werden, beispielsweise um eine Darstellung von Sicherheitsmechanismen wie Firewalls auszublenden (dies betrifft auch IANF 1).

Die anderen Teilvorgänge in der Visualisierungspipeline können in mehreren kooperierenden Komponenten als einzelne Managementanwendungen realisiert werden, die ihrerseits über die Infrastruktur der Managementplattform miteinander verbunden sind. Im Folgenden werden diese als *Visualisierungsanwendungen* bezeichnet.

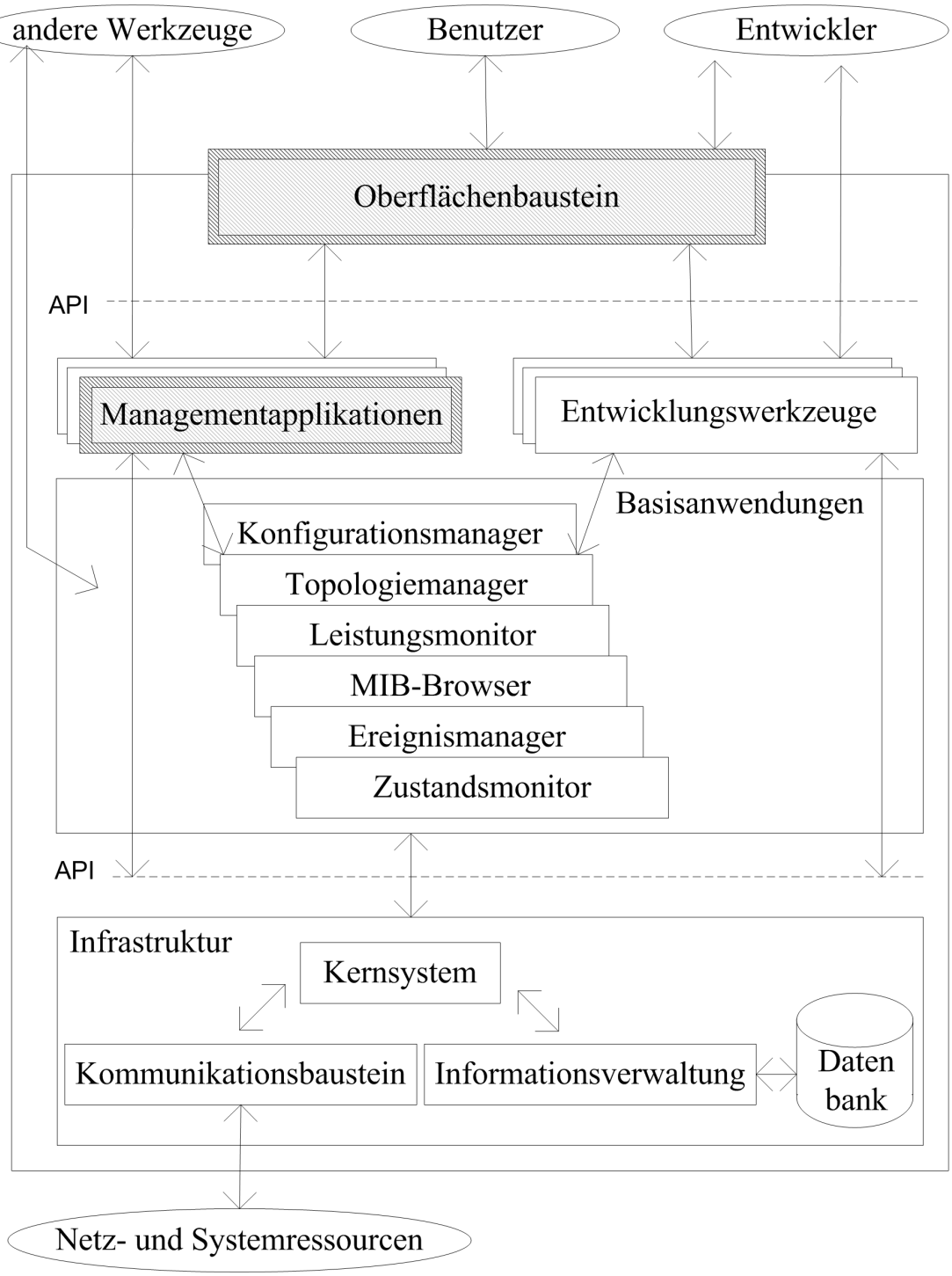


Abbildung 4: Aufbau von Managementplattformen

Definition 3.16 *Unter einer Visualisierungsanwendung versteht man eine in eine Managementplattform integrierte Managementanwendung, die einen Teil einer Visualisierungspipeline realisiert und damit einen Teil der für eine Visualisierung notwendigen Funktionalität übernimmt. Visualisierungsanwendungen kooperieren untereinander über die Infrastruktur einer Managementplattform und tauschen mit deren Hilfe ihre Zwischenergebnisse aus.*

Die erste dieser Visualisierungsanwendungen bezieht die Rohdaten des Netzes über die Infrastruktur der Managementplattform (MANF 1) von den Managementanwendungen und liefert ihre Ergebnisse an diese zurück. Die weiteren Teile der Pipeline fordern die Zwischenergebnisse von der Plattform an und geben ihrerseits die neuen Zwischenergebnisse an sie zurück. Schließlich stellt die letzte der Anwendungen das von den anderen Teilen der Pipeline berechnete Ergebnis dem Oberflächenprozess zur Verfügung.

Eine solche Struktur stellt die modulare Ausbaufähigkeit um zusätzliche Funktionalitäten sicher. Auf einfache Art und Weise können so auch neue Funktionalitäten $\mathbb{F}$ zur Manipulation von $\mathbb{PR}$ hinzugefügt werden. Außerdem kann so gegebenenfalls durch eine von der Infrastruktur einer geeigneten Plattform gewährleistete transparente Datenhaltung eine einfache Verteilung der gesamten Visualisierungspipeline bzw. einzelner Teile des Informations- und Präsentationsraumes ermöglicht werden.

Alle zu integrierenden Managementwerkzeuge sind, wie in Abbildung 4 ersichtlich, mit der Infrastruktur der Managementplattform verbunden. Das heißt, dass diese entweder auf der Plattform selbst laufen, oder ein Kommunikationskanal über das Rechnernetz existiert. In beiden Fällen können sie über die Infrastruktur der Plattform die Visualisierungsanwendung erreichen und von ihr erreicht werden. Sie sind hier nicht Teil der Spezifikation, werden aber unweigerlich Teil der Anwendung sein.

Um die einzelnen Transformationen durchführen zu können, sind die folgenden Visualisierungsanwendungen notwendig. Sie werden hier kurz vorgestellt und in den nächsten Kapiteln genau spezifiziert. Da es sich hier strenggenommen um Visualisierungsdienste (engl. visualisation service) handelt, werden sie als "Services" bezeichnet.

- Anwendungsservice - übernimmt die Verwaltung der IOs und der Relationen.
- Mappingservice - bildet die IOs auf POs nach bestimmten in seinem Mappingmodell definierten Regeln ab.
- Visualisierungsservice - bildet die vorhandenen POs in einen Szenegraphen ab und steuert die Übertragung des Ergebnisses an die Clients. Außerdem empfängt er Interaktionsanweisungen und leitet diese entsprechend weiter.

Ferner wird eine grundlegende Visualisierungsfunktion definiert, der Layoutservice. Dieser übernimmt die Positionierung der POs im Raum und hält sich dabei an eine definierte Layoutpolicy (VANF 16).

Abbildung 5 zeigt die Einordnung der soeben beschriebenen Visualisierungsanwendungen und des Benutzer-Interfaces in eine Managementplattform sowie die notwendigen Schnittstellen.

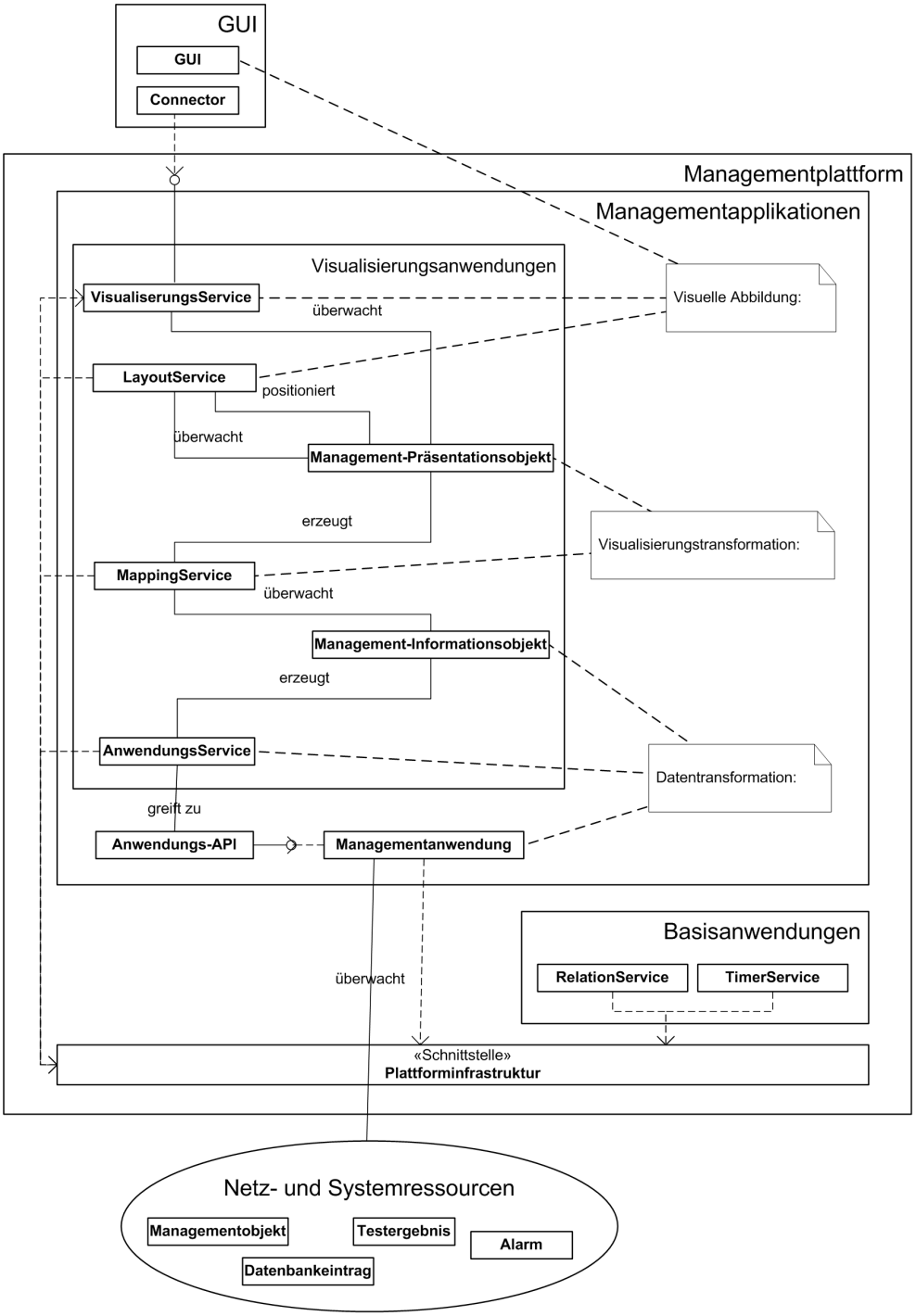


Abbildung 5: Integration der Visualisierungspipeline in eine Managementplattform

Die Abstraktionsebenen der Daten und die Aufteilung der Funktionalitäten resultieren in einer hierarchischen Schichtung der Architektur mit zunehmender Abstraktion von unten nach oben. Ganz unten befindet sich die Schicht der Managementanwendungen und der Managementobjekte sowie der Anwendungsservice. Darüber befinden sich die Daten in der Stufe der analytischen Abstraktion (Informationsobjekte), die konkrete Hinweise auf den Inhalt der Managementobjekte sowie auf die mögliche Visualisierung derselben enthalten, und als Visualisierungsanwendung der Mappingservice. Zwischen den Schichten erfolgt die Datentransformation. Oberhalb der analytischen Abstraktion befindet sich die Schicht der Darstellungsabstraktion (Präsentationsobjekte). Sie enthält Daten, die die grafische Repräsentation betreffen, Visualisierungsfunktionen (hier beispielhaft der Layoutservice) und als Visualisierungsanwendung den Visualisierungsservice. Von der analytischen zur Darstellungsabstraktion erfolgt die Visualisierungstransformation, die u.a. ein Mapping zwischen den Objekten vornimmt. D.h. sie weißt den analytischen Abstraktionen konkrete grafische Repräsentationen zu. Zuletzt erfolgt die visuelle Abbildung auf den View, der dann im Benutzer-Interface angezeigt wird.

Zur Anbindung von Managementanwendungen an den Anwendungsservice wird außerdem ein API zur Verfügung gestellt, das "Anwendungs-API". Es stellt alle Funktionen bereit, über die eine Managementanwendung mit dem Visualisierungssystem kommunizieren kann. Auf der anderen Seite des Visualisierungssystems beim Benutzer-Interface, befindet sich ein Programmteil, der die Kommunikation mit dem Visualisierungsservice steuert. Dieser wird "Connector" genannt.

Während die Integration in das allgemeine Modell von Managementplattformen prinzipiell in jedes System so wie beschrieben stattfinden kann, müssen für die konkrete Ausarbeitung eines Designs einige grundlegende Vereinbarungen zur Umgebung getroffen werden, wenn man nicht in allen Details "das Rad neu erfinden" will. Im Folgenden wird deshalb eine Managementumgebung angenommen, die die folgenden Eigenschaften aufweist:

- Komponentenarchitektur mit einem Server, der die Komponenten und Objekte in einer Datenbank hält und sie für die Anwendungen sichtbar macht,
- dynamische Erweiterbarkeit zur Laufzeit.
- Alle verwendeten Objekte und Komponenten implementieren eine Managementschnittstelle, d.h. sie besitzen gesetzte Attribute, auf die zugegriffen werden kann, Operationen, die aufgerufen werden können und sie sind in der Lage, Notifikationen zu senden und zu empfangen.
- Vorhandensein eines Relationservices,
- Vorhandensein eines Timerservices.

Der Relationservice und der Timerservice werden zentral zur Verfügung gestellt. Der Relationservice verwaltet die Relationen zwischen den Managementobjekten in der Plattform. In der vorliegenden Architektur werden die Relationen vom Anwendungsservice erzeugt und vom Visualisierungs- und Layoutservice angefragt. Der Timerservice sendet zu bestimmten, vom anwendenden Service bestimmbaren Zeitabständen Notifikationen an diesen. Dies löst beispiels-

weise beim Layoutservice regelmäßig einen Layoutvorgang aus.

3.4 Informationsschicht: Datentransformation und Informationsraum

Der in Kapitel 3.2.2 definierte Informationsraum kann nun auf den konkreten Fall des Netzmanagements angewendet werden. Dort gibt es den Begriff des *Managementobjektes (MO)*. Darunter versteht man die Abstraktion der Charakteristika der Ressourcen eines Rechnernetzes. Ein Managementobjekt kapselt als konsistentes Abbild die managementrelevanten Eigenschaften der zu steuernden, zu überwachenden oder zu verwaltenden Betriebsmittel und deren managementrelevanten Beziehungen bzw. Relationen (vgl. [Proe 02] und [HAN 99]).

Managementobjekte stellen also in der Regel die zu visualisierenden Rohdaten dar. Um eine Weiterverarbeitbarkeit der Daten zu erreichen, müssen sie ein festgelegtes Format haben. Da nicht vorhergesagt werden kann, welches Format von Managementobjekten[17] im konkreten Anwendungsfall vorliegt, werden sog. *Managementinformationsobjekte* eingeführt.

Definition 3.17 *Unter einem Managementinformationsobjekt MIO versteht man ein in einem spezifizierten Format, das den Visualisierungsanwendungen bekannt ist, vorliegendes Managementobjekt. Es trägt den für eine Visualisierung benötigten Teil der Informationen über die ihm zugeordnete Ressource.*

Eine softwaretechnische Spezifikation der MIOs folgt in Kapitel 3.4.1. Mit der Konstruktion eines Managementinformationsobjektes kann nun die nötige Informationsmenge definiert werden:

Definition 3.18 *Die Managementinformationsmenge $\mathbb{IM}_{\mathbb{M}}$ ist eine diskrete Menge definierter Managementobjekte, Managementinformationsobjekte (MIO) genannt:*

$$\mathbb{IM}_{\mathbb{M}} = \{MIO_1, MIO_2, \ldots, MIO_n\} \quad \forall i \nexists j : MIO_i = MIO_j \quad mit\, i \neq j, \quad n, i, j \in \mathbb{N} \tag{13}$$

Die Managementinformationsmenge bildet die Definitionsmenge für die Festlegung der Informationsstruktur, die die Beziehungen zwischen den Objekten beschreibt. Hierfür werden zunächst zwei Relationen vorgesehen:

- Die Enthaltenseinsrelation, sie realisiert VANF 18 und spiegelt die Tatsache wieder, dass die Ressourcen im Netzmanagement oft zusammengesetzt sein können; beispielsweise hat eine Netzkomponente vom Typ Switch meist mehrere Schnittstellenkarten, diese wiederum mehrere Ports.

  Definition 3.19 *Die Informationsstruktur ,,Enthaltensein" $\mathbb{IS}_{\mathbb{E}}$ sei definiert als:*

  $$\mathbb{IS}_{\mathbb{E}} = \{(a, b) \mid b\, ist\, in\, a\, enthalten\} \quad a, b \in \mathbb{IM}_{\mathbb{M}} \tag{14}$$

- Die Adjazenzrelation (Nachbarschaftsrelation), sie realisiert VANF 17. Außerdem stellt sie ein Datenmodell dar, das den Austausch von Informationen zwischen jeweils zwei

[17] beispielsweise SNMP-MIB, MOC, Klasse, Datenbankeintrag, o.a.

Systemen oder Protokollinstanzen berücksichtigt und ermöglicht damit die Erfüllung der Anforderung VANF 13, die fordert, dass Schichtprotokolle als Verbindungen zwischen ihren Endsystemen dargestellt werden können.

Definition 3.20 *Die Informationsstruktur „Adjazenz" $\mathbb{IS}_{\mathbb{A}}$ sei definiert als:*

$$\begin{aligned}\mathbb{IS}_{\mathbb{A}} = \{&(a,c) \mid a\ ist\ mit\ c\ verbunden \vee \\ &(a,b,c) \mid a\ ist\ mit\ Hilfe\ von\ b\ mit\ c\ verbunden\} \\ &a,b,c \in \mathbb{IM}_{\mathbb{M}}\end{aligned} \tag{15}$$

Abbildung 6 zeigt die Rollen und Kardinalitäten dieser Relationen. Letztere berücksichtigen die Forderung nach Uni- und Multicast aus VANF 19.

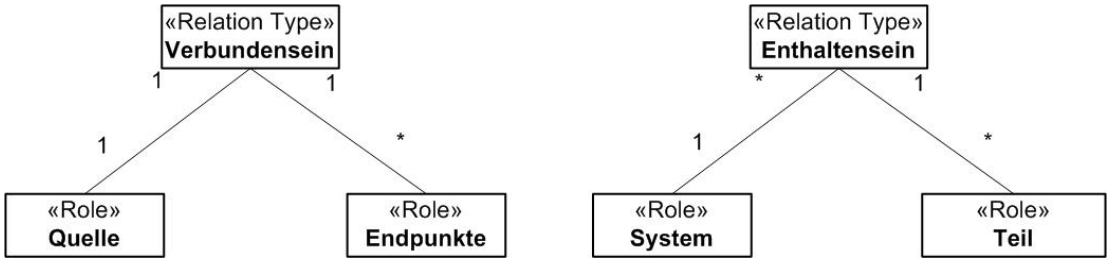


Abbildung 6: Relationen der Informationsstruktur

Nun läßt sich der entsprechende Informationsraum festlegen.

Definition 3.21 *Der Managementinformationsraum $\mathbb{IR}_{\mathbb{M}}$ sei durch die Informationsmenge $\mathbb{IM}_{\mathbb{M}}$ und die Informationsstrukturen $\mathbb{IS}_{\mathbb{E}}$ und $\mathbb{IS}_{\mathbb{A}}$ definiert:*

$$\mathbb{IR}_{\mathbb{M}} = \{\mathbb{IM}_{\mathbb{M}}, \{\mathbb{IS}_{\mathbb{E}}, \mathbb{IS}_{\mathbb{A}}\}\} \quad mit |\mathbb{IM}_{\mathbb{M}}| \geq 1 \quad und\ |\mathbb{IS}_{\mathbb{E}}| \geq 0, \quad |\mathbb{IS}_{\mathbb{A}}| \geq 0 \tag{16}$$

3.4.1 Objekte

Es folgt nun die Spezifikation der notwendigen Attribute und Methoden, über die MIO verfügen müssen. Hierzu wird eine Grundklasse definiert. Im einzelnen Anwendungsfall müssen von dieser abgeleitete Klassen implementiert werden, die über die Attribute, die zur Visualisierung der dem jeweiligen Objekt zugeordneten Ressource erforderlich sind, verfügen.

Jedes MIO muss mit folgenden Attributen ausgestattet sein:

- Eine Typenangabe, die beschreibt, von welcher Art das MIO ist (praktisch der Klassenname). Entsprechend erfolgt später das Mapping auf das Präsentationsobjekt (`String Type`).

- Eine Angabe, die festlegt, welche von (eventuell) mehreren vorhandenen Repräsentationen für diesen Typ bevorzugt verwendet werden soll. Erst wenn diese nicht verfügbar ist, wird eine Standardrepräsentation verwendet (`String Domain`). Siehe dazu auch Kapitel 3.5.
- Eine für den lokalen $\mathbb{IR}$ eindeutige ID (`String MIOID`),
- eine Referenz auf die Managementanwendung, die es angemeldet hat (String Source),
- eine Beschreibung der referenzierten Ressource in natürlicher Sprache (`String Description`).

Daraus resultiert die in Abbildung 7 dargestellte Oberklasse `MIO`. Sie dient als Template (Vorlage) für alle verwendeten Informationsobjekte, deren Struktur je nach Bedarf erweitert werden kann.

Die anderen in der Abbildung dargestellten Klassen realisieren einige der Punkte aus dem Anforderungskatalog: `System` und `Architecture` dienen als Wurzeln für die Repräsentationen der beteiligten Systeme sowie verschiedener Kommunikationsarchitekturen und anderer logischer Konzepte (realisiert VANF 11). Die Attribute der Objekte unterhalb `System` und `Internet` sind Beispiele und können je nach Anwendungsfall erweitert werden, sie berücksichtigen unterhalb der Klasse `Internet` die geforderten Funktionalitäten aus den Anforderungen VANF 15, VANF 14 und VANF 13. `Internet` enthält das Attribut `Layer`, das die Schicht angibt, in der sich das betrachtete Objekt befindet. Das erleichtert es später dem Layoutservice, es entsprechend seinem Kontext innerhalb der Kommunikationsarchitektur einzuordnen (MANF 4). Darunter wird dann zwischen den Klassen `ProtocolInstance` (Protokollinstanz) und `PDU` (Protocol Data Unit) unterschieden. Dies reflektiert den Sachverhalt, dass eine PDU immer eine Quelle und ein oder mehrere Ziele besitzt, die sog. Protokollinstanzen. `PDU` stellt dabei neben einem Verkehrsfluss auch eine Relation dar. Je nach verwendeter Plattform müssen hier evtl. Operationen und Attribute zur Relationsverwaltung definiert werden. Die übrigen Klassen dienen als Beispiele für Systeme, Protokolle und Protokollinstanzen. Einige davon sind auch für die Realisierung der in Kapitel 2 vorgestellten Szenarien nützlich.

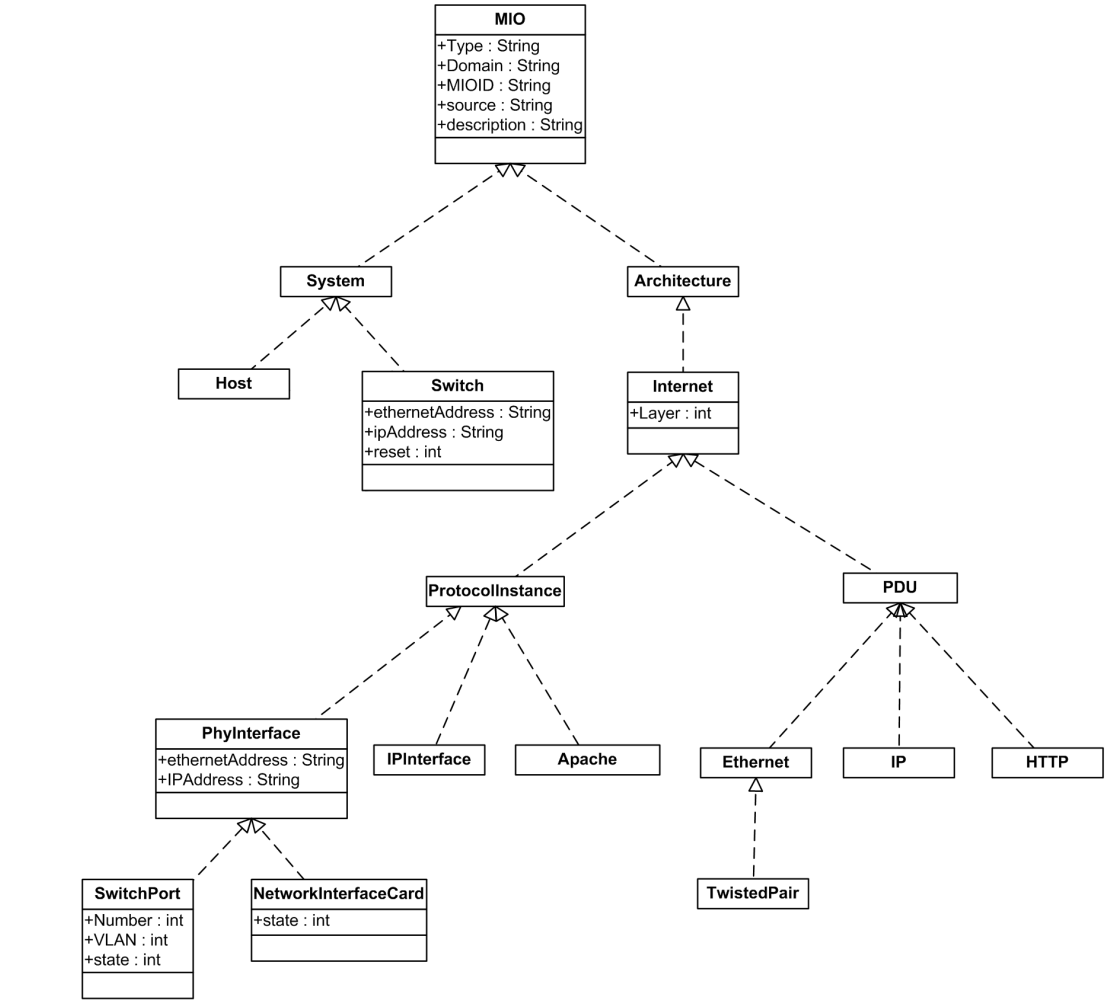
Zur Dynamik der MIOs wird folgendes festgelegt: Wird ein Attribut eines MIO verändert, so erhält der Anwendungsservice und alle anderen angemeldeten Komponenten[18] eine Nachricht, die den Namen des entsprechenden Attributes sowie die MIOID des MIOs oder das MIO selbst enthält.

3.4.2 Anwendungsservice

Der Anwendungsservice ist für die Transformation der Rohdaten aus den Managementanwendungen in MIOs zuständig. Er realisiert das Anwendungsinterface, das den Managementanwendungen eine Schnittstelle für seine Ansteuerung zur Verfügung stellt.

Die Sammlung der Rohdaten wird den Managementanwendungen überlassen. Diese sammeln

[18] beispielsweise POs, siehe Kapitel 3.5.1



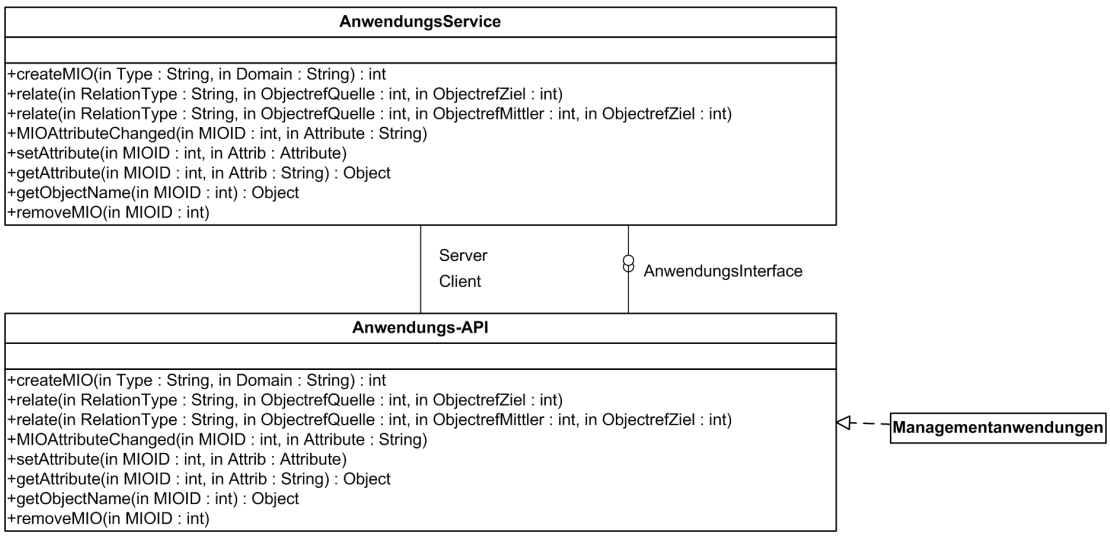


Abbildung 7: Visualisierungsanwendungen und -objekte der Informationsschicht

und halten ihre Objekte in einem beliebigen Format und können mit Hilfe des Anwendungsservices die entsprechenden MIOs erzeugen. Das Anwendungsinterface berücksichtigt auch die Bildung von Relationen zwischen den Rohdaten, soweit diese als MIOs registriert wurden.

Ein solcherart formalisierter Informationsraum ist das Ergebnis der Datentransformation und die Voraussetzung für die Transformationen späterer Teile der Visualisierungspipeline. Durch das beschriebene Vorgehen wird die Forderung nach einem Datenmodell erfüllt, das aus unterschiedlichsten Quellen gespeist werden kann (VANF 6 und VANF 7). Eine solche Quelle kann entweder die in eine Visualisierung zu integrierende Managementanwendung selbst sein, die mit Hilfe des Anwendungsservices die MIOs erzeugt (gegebenenfalls als ganze oder teilweise Kopie von Managementobjekten) oder aber eine *Proxyanwendung*[19], die stellvertretend für die zu integrierende Managementanwendung die Verwaltung und Erzeugung der MIOs vornimmt. Proxyanwendungen werden beispielsweise dann notwendig, wenn es sich bei der zu integrierenden Anwendung um ein Fremdprodukt handelt, dessen Sourcecode nicht zugreifbar ist oder wenn eine indirekte Anbindung an das Visualisierungssystem erwünscht ist bzw. aus anderen Gründen notwendig wird.

Das Anwendungsinterface stellt die in Abbildung 7 dargestellten abstrakten Operationen bereit, um MOs von Managementanwendungen in MIOs überzuführen. Bei einem Aufruf von `createMIO(String type, String domain)` erzeugt der Anwendungsservice das mit Hilfe des Parameters `Type` gewählte MIO und setzt dementsprechend dessen Attribut `Type` und `Domain`. Das Attribut `Source` wird ebenfalls gesetzt und enthält einen Identifikator auf die aufrufende Managementanwendung. Für `MIOID` wird eine laufende lokal eindeutige Nummer vergeben, die mit Hilfe etwa einer Hashmap mit den Daten, die die Infrastruktur der Managementplattform zum Zugriff auf das erzeugte Objekt benötigt, verknüpft wird. Zum Zugriff auf dieses Mapping von anderen Visualisierungsanwendungen aus steht die Funktion `getObjectName(int MIOID)` zur Verfügung. Desweiteren erfolgt eine interne Assoziation des MIOs mit der oder den Managementanwendungen, die das Objekt angemeldet haben oder darauf zugreifen. Die Operation `createMIO(String type, String domain)` liefert den Identifikator an die aufrufende Managementanwendung zurück, die diesen fortan als Referenz verwenden kann.

Mit Hilfe der Operation `removeMIO(int id)` können erzeugte MIOs wieder aus dem System entnommen werden. Abhängige MPOs und Relationen werden dabei ebenfalls entfernt. Mit den Operationen `getAttribute(int MIOID,String Attrib)` und `setAttribute(int MIOID, Attribute Attrib)` können Managementanwendungen und Visualisierungsanwendungen auf die Attribute des MIOs zugreifen. Hierbei liefert ersteres bei Nennung des MIOs und des Attributes dessen Wert zurück, während letzeres den Wert eines Attributes setzt.

Ändert sich ein Attribut eines MIO, so erhält der Anwendungsservice von diesem eine entsprechende Nachricht. Bei deren Empfang wird die Funktion `MIOAttributeChanged(int MIOID, String Attrib)` aufgerufen, die die Managementanwendungen, die das MIO

[19] engl.: Stellvertreter

verwalten bzw. angemeldet haben (sie werden über das Attribut Source des MIO bestimmt), darüber informiert, indem sie dort ebenfalls `MIOAttributeChanged(int MOID, String Attrib)` aufruft. Diese können dann auf die Datenquelle zugreifen und Änderungen an den zugrundeliegenden Managementobjekten durchführen bzw. die Daten des MIO mit den Rohdaten abgleichen. Darüber hinaus sendet das MIO diese Nachricht auch an das dem MIO entsprechende Präsentationsobjekt, damit dort die grafische Repräsentation evtl. neu aufgebaut werden kann.

Die Methoden `relate(String RelationType, int ObjectrefQuelle, int ObjectrefZiel)` und `relate(String RelationType, int ObjectrefQuelle, int ObjectrefMittler, int ObjectrefZiel)` dienen zum Anmelden von Beziehungen zwischen bereits registrierten MIOs. Die Managementanwendung ruft beim Anwendungsservice diese Funktionen auf, und dieser registriert die Relationen beim Relationservice der Managementplattform. Die letztgenannte relate-Operation dient dazu, zwei MIOs mit Hilfe eines dritten zu verknüpfen. Beispielsweise wird ein Switchport mit Hilfe eines Twisted-Pair-Kabels mit einem anderen Switchport verknüpft. Das Kabel nimmt hier die Rolle des Mittlers ein. Je nach Plattform ist für diesen Mechanismus eine Erweiterung der Mittler-MIOs um einige Methoden notwendig (beispielsweise bei JMX die Implementierung des Interfaces `javax.management.relation.Relation`). Dies gilt vor allem für das MIO `PDU`.

3.4.3 Anwendungs-API

Das Anwendungs-API unterstützt die Managementanwendungen bei der Herstellung der Verbindung mit dem Anwendungsservice. Je nach Implementierung und verwendeter Managementplattform kann dies unterschiedlich erfolgen. das API kann von allen Managementanwendungen verwendet werden, die das Visualisierungssystem nutzen wollen. Mit Hilfe von Proxyanwendungen mit Anwendungs-API ist es möglich, durch Zugriff auf exportierte Datenbasen auch Managementanwendungen zu integrieren, die das Anwendungs-API nicht verwenden. Die Proxyanwendungen können die von zu integrierenden Anwendungen erzeugten Daten und Managementobjekte und die notwendigen strukturellen Informationen sammeln und Informationsmenge und -strukturen an das Anwendungs-API weitergeben. Dies setzt allerdings vorraus, dass diese Proxyanwendungen mit der Struktur und der Semantik der Rohdaten für den jeweiligen Anwendungsfall vertraut sind.

Das Anwendungs-API bietet alle Funktionen des Anwendungsservice an und stellt alle Funktionalitäten bereit, die notwendig sind, um auf den Anwendungsservice zugreifen zu können. Jede Anwendung kann, soweit sie Informationen über die Änderung der von ihr angemeldeten Objekte empfangen will, für diesen Fall eine Schnittstelle bereitstellen. Dies geschieht durch Überschreiben der Funktion `MIOAttributeChanged(int MIOID,String Attrib)`. Es gehört zu den Aufgaben jeder Managementanwendung, die das Anwendungs-API verwendet, bei Bedarf für die Konsistenz von MIO und zugrundeliegenden Rohdaten zu sorgen.

Sollen Ein- und Ausgaben direkt von der Managementanwendung erfolgen, so erfolgt dies über

die von der Managementplattform zur Verfügung gestellte Managementschnittstelle. Die Ein- und Ausgabe erfolgt dabei in der Regel über die Änderung von Attributen. Wie im Kapitel 3.7 (Interaktion) beschrieben, kann so auf einfache und modellkonsistente Weise eine Oberflächenintegration der Ein- und Ausgabemechanismen von Managementanwendungen erreicht werden.

3.5 Präsentationsschicht: Visualisierungstransformation und Präsentationsraum

So wie der Informationsraum für das Management festgelegt wurde, erfolgt nun auch eine Festlegung des Präsentationsraumes für das Management. Dazu wird zunächst die Art der notwendigen Präsentationsobjekte für die gewünschte Anwendung festgelegt.

Definition 3.22 *Ein Managementpräsentationssobjekt MPO ist eine logische grafische Einheit im Prozess der Präsentation. Es greift auf keines, eines oder mehrere der Attribute seines zugehörigen MIOs zu und generiert daraus in Abhängigkeit von seinen eigenen Parametern eine grafische Repräsentation, d.h. Shapes (grafische Form, Polygonbeschreibungen) und beispielsweise Farbinformationen oder ähnliche Darstellungsbeschreibungen. Ferner enthält es aktuelle Simulationsdaten wie seine Lage und Sichtbarkeit und generiert daraus die notwendigen Translationen (Verschiebungen) und Rotationen (Drehungen) für seine Position im Raum.*

Eine Menge aus MPOs bildet dann die Managementpräsentationsmenge.

Definition 3.23 *Die Managementpräsentationsmenge $\mathbb{PM}_{\mathbb{M}}$ ist eine diskrete Menge von Managementpräsentationsobjekten*

$$\mathbb{PM}_{\mathbb{M}} = \{MPO_1, MPO_2, \ldots, MPO_n\} \quad \forall i \nexists j : MPO_i = MPO_j \quad mit\, i \neq j, \quad n, i, j \in \mathbb{N} \tag{17}$$

Die Präsentationsinformationsbasis (PIB) enthält Template-Klassen für MPOs, die vom Mappingservice bei Bedarf instantiiert werden.

Weiter existieren Visualisierungsfunktionen $\mathbb{F}_{\mathbb{P}}$, die auf die Managementpräsentationsmenge $\mathbb{PM}_{\mathbb{M}}$ einwirken, um das Aussehen der Szene gemäß ihren Parametern zu verändern.

Definition 3.24 *Eine Visualisierungsfunktion $\mathbb{F}$ wirkt auf die Attribute der Präsentationsobjekte ein, um Teile der Präsentation zu verändern. Jede Funktion überwacht die Präsentationsmenge und verändert gemäß ihrer Aufgabe die Attribute der Präsentationsobjekte. Dabei hat sie auch Zugriff auf die Daten des Informationsraumes.*

Der Layoutservice oder eine Markierungsfunktion für VLANs sind solche Funktionen. Gemeinsam mit der Managementpräsentationsmenge ergeben sie den Managementpräsentationsraum $\mathbb{PR}_{\mathbb{M}}$.

Mit Funktionen kann beispielsweise die Sichtbarkeit einiger Objekte ausgeschaltet und die anderer eingeschaltet werden. Es können auch Farbmarkierungen verändert und beispielsweise Verbindungen in einem bestimmten VLAN (also abhängig von den im Informationsraum gespeicherten Daten) mit anderen Farben markiert werden. Die Methoden der Funktionen werden entweder durch Benutzerinteraktionen oder durch das Eintreten von Ereignissen, beispielsweise einer Notifikation vom Timerservice, gestartet.

Definition 3.25 *Ein Managementpräsentationsraum $\mathbb{PR}_{\mathbb{M}}$ wird durch eine Präsentationsmenge $\mathbb{PM}_{\mathbb{M}}$ und eine Menge $\mathbb{F}_{\mathbb{P}}$ von Funktionen gebildet.*

$$\mathbb{PR}_{\mathbb{M}} = \{\mathbb{PM}_{\mathbb{M}}, \mathbb{F}_{\mathbb{P}}\} \quad mit \quad |\mathbb{PM}_{\mathbb{M}}| \geq 1 \quad und \quad |\mathbb{F}_{\mathbb{P}}| \geq 0, \quad \mathbb{F}_{\mathbb{P}} = \{F_1, F_2, ..., Fn\}, \quad n \in \mathbb{N} \tag{18}$$

Die Attributmenge und die Dimension des Managementpräsentationsraumes kann analog zu den Formeln 10 und 11 definiert werden. Auch Präsentationsobjekte verfügen über Attribute, die allerdings nicht die bereits in den korrespondierenden Informationsobjekten enthaltene Information beschreiben, sondern vielmehr deren grafische Repräsentation.

3.5.1 Objekte

Wie in der Definition von abstrakten Visualisierungsobjekten in Kapitel 3.2 und der Definition der MPOs bereits erwähnt, haben Präsentationsobjekte eine imaginäre Ausdehnung in Raum und Zeit. Ihre Attribute beschreiben aktuelle Simulationsdaten (VANF 5).

Es sind deshalb mehrere grundlegende Informationen notwendig:

- Die Sichtbarkeit des Objektes (VANF 20)
- Die Position im dreidimensionalen Raum (VANF 4).
- die Geometrie und Erscheinung des Objektes (VANF 1 und VANF 10)

Das Attribut `Visibility` bezeichnet die Sichtbarkeit des gesamten Objektes und wird als ein Attribut mit dem Typ `Boolean` dargestellt. Die Position im dreidimensionalen Raum wird durch die Attribute `x`, `y` und `z` mit dem Typ `Float` bestimmt. Diese sind dabei immer die relativen Koordinaten zu dem in der Enthaltenseinsrelation übergeordneten Objekt. Sie werden in der Regel vom Layoutservice festgelegt.

Die Erscheinung des Objektes schließt beispielsweise die Farbe, die Transparenz und die Oberflächentextur (VANF 10) ein. Für die Geometrie und die Erscheinung des Objektes sind zwei Attribute notwendig, die abhängig von den anderen gegebenenfalls zusätzlich definierten Attributen des MPO eine Beschreibung der grafischen Repräsentation des Objekts und deren Lage im Koordinatensystem liefern (VANF 3). Dies sind die Attribute `Shape` und `Transform`. Sie dienen dem Visualisierungsservice während des Aufbaus der gesamten Szene und stellen Teile des Szene-Graphen[20] dar. Das Atribut `Shape` definiert die notwendigen Beschreibungen der Form und des Aussehens des dargestellen 3D-Modells abhängig von seinen grafischen Attributen. Hier werden Polygone oder geometrische Grundformen definiert, aber auch Beleuchtung, Texturen, Material und Farben des Objektes. Das Attribut `Transform` enthält in Form von Translationen (Verschiebungen), Rotationen (Drehungen) Anweisungen für die Positionierung im Raum. Für die Werte dieser Attribute können Standards wie VRML oder Java3D verwendet werden. Die hier definierte Grundklasse MPO in Abbildung 8 verwendet die Java3D-Klassen `BranchGroup` und `TransformGroup`.

Außerdem verfügen MPOs über verschiedene Funktionalitäten $\mathbb{F}_{\mathbb{P}}$. Dies bedeutet, dass sie nicht statisch sind, sondern ihre Attribute selbst verändern können. Ein MPO muss außerdem aktiv die Attribute seines zugehörigen MIOs überwachen können und bei Bedarf seine eigenen grafischen Attribute dementsprechend verändern (VANF 2). Dies erfordert eine Methode `start()`, die nach Registrierung des MPOs Grundeinstellungen vornimmt und den gewünschten Er-

[20] eine Definition und Erläuterungen finden sich in Kapitel 3.6

halt von Notifikationen beim zugrundeliegenden MIO anmeldet[21]. Ferner wird eine Operation `MIOAttributeChange(String Attribute)` benötigt, um die notwendigen Änderungen an den grafischen Attributen vornehmen zu können. Sie wird initiiert, sobald das MPO direkt vom zugehörigen MIO[22] eine entsprechende Nachricht erhält, wenn sich die Attribute eines MIO geändert haben.

Ähnliches gilt, wenn ein Attribut des MPO selbst verändert wird. So kann beispielsweise durch eine Änderung des Attributes `Sichtbarkeit` eine entsprechende Änderung anderer Attribute des MPO initiiert werden. Die ensprechende Methode `POAttributeChange(String Attribute)` wird üblicherweise von der Setterfunktion der Attribute aufgerufen. Ist das Attribut `sendChangeNotifications` gesetzt, so erfolgt das Senden einer AttributeChange-Notifikation, sobald sich ein Attribut des MPO verändert hat. Damit bei einem hintereinander erfolgenden Setzen aller drei Koordinaten (z.B. nach dem Layout) des MPO nicht drei Notifikation abgesendet werden, wurde zusätzlich die Operation `setCoords()` eingeführt.

Manche MPOs, die Verbindungen zwischen zwei anderen darstellen, benötigen eine Möglichkeit, Änderungen an den Relationen feststellen zu können. Hierzu wird die Methode `RelationChange()` spezifiziert. Sie wird aufgerufen, sobald der Relationservice eine Änderung an einer Relation feststellt. Für jede der Klassen können je nach gewünschter Komplexität und Funktionalität der Visualisierung weitere Attribute festgelegt werden. Die Struktur und die Namen der von der Grundklasse MPO abgeleiteten Klassen sind dieselben, wie sie bereits im Informationsmodell für MIOs definiert wurden.

Mit den hier definierten Mitteln können komplexe MPOs mit Simulationsfunktionalitäten erstellt werden. Für eine weitgehende Simulation der Netzplanung und Logdatenüberwachung wäre ferner beispielsweise die Attributierung detailierter Simulationsdaten wie der Uhrzeit des angezeigten MPO-Zustandes interessant. Im vorliegenden Entwurf soll auf solche Möglichkeiten aus Gründen der Komplexität aber nicht eingegangen werden.

Abbildung 8 zeigt die im vorstehenden definierte Oberklasse "MPO" sowie einige weitere Präsentationsobjekte. Die Struktur entspricht dem Informationsmodell der zugehörigen MIOs in Abbildung 7.

Die Klasse `Internet` enthält ein Attribut `Height` als Abstandskonstante, die den gewünschten Abstand zwischen den Visualisierungen einzelner Schichten der visualisierten Kommunikationsarchitektur definiert. Mit Hilfe dieser Konstante und der Schichtzugehörigkeit (Attribut `Layer`) des dem MPO zugehörigen MIO wird ein Offset der Höhenkoordinaten für das MPO errechnet. Objekte, die in einer höheren Schicht in der Kommunikationsarchitektur angesiedelt sind, werden so entsprechend "geographisch" höher visualisiert. Dieser Mechanismus berücksichtigt die Anforderung VANF 13 und erleichtert es, die Objekte auch grafisch entsprechend der Kommunikationsarchitektur einzuordnen. Die tatsächliche Darstellung des Objektes auf der Y-Koordinate resultiert dann aus der einfachen Formel `y=y+Height*Layer`. Liegt ein Objekt

[21] wird der im Ausblick vorgeschlagene Mechanismus zur Verteilung des Systems eingesetzt, so ist dies nicht mehr möglich, die Notifikationen müssen dann vom Anwendungsservice vermittelt werden

[22] oder indirekt vom Anwendungsservice

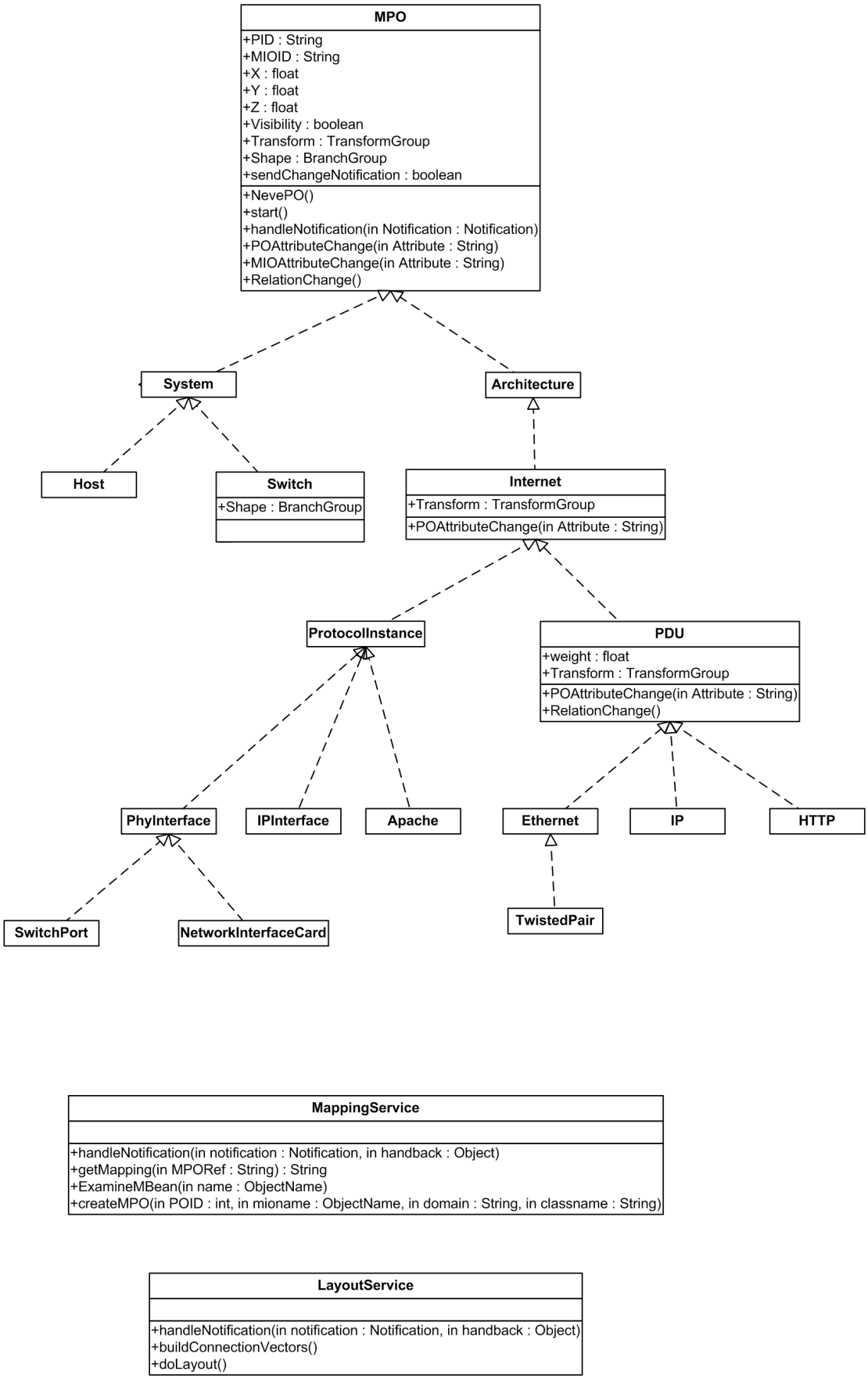


Abbildung 8: Visualisierungsanwendungen, -objekte und -funktionen der Präsentationsschicht

beispielsweise in Schicht 3 und die Abstandskonstante beträgt 10, so erfolgt die y-Positionierung des Objektes auf `y=y+30`.

Entsprechend den in den Anforderungen festgelegten Visualisierungsparadigmen sind `Twisted-Pair` und `Protocol"` Unterklassen der Klasse `PDU`, die eine Verbindung zwischen zwei physikalischen oder logischen Systemen darstellt, die von der Klasse classProtocolInstance repräsentiert werden.

3.5.2 Mappingservice

Der Mappingservice führt die Visualisierungstransformation durch. Seine Aufgabe ist es, den Präsentationsraum konsistent zu halten, so dass für jedes MIO ein entsprechendes MPO vorhanden ist. Zu diesem Zweck überwacht der Mappingservice mit Hilfe der Infrastruktur der Managementplattform den Informationsraum, indem seine Operation `handleNotification(Notification notification, Object handback)` von ihr eine Nachricht für jedes neu erzeugte oder veränderte Objekt erhält. Die Operation `ExamineMBean()` untersucht das Objekt darauf, ob es sich um ein MIO handelt. Im positiven Fall, wird die Methode `createMPO(int POID, ObjectName mioname, String domain, String classname)` aufgerufen, die entsprechend den Attributen `Type` und `Domain` ein neues MPO erzeugt oder ein existierendes ersetzt (VANF 1). Durch einmaligen Aufruf der Methode `start()` des betreffenden MPO wird dort außerdem dessen Initialisierung vorgenommen. Weiter wird ein Eintrag in einer für den Mappingservice global definierten Mappingtabelle gespeichert, der notiert, welches MIO auf welches MPO abgebildet wurde. Sie wird von der später beschriebenen Funktion `getMapping(String MPORef)` verwendet.

Die Entscheidung, welches MPO zu einem bestimmten MIO erzeugt werden soll, wird anhand der in den MIOs festgelegten Informationen über deren Typ und die bevorzugte grafische Repräsentation, dem Inhalt des Attributes `Domain`, getroffen. Bereits der Name weißt auf die hier verwendete Domänenstruktur hin, die zum Auffinden verschiedener Repräsentationen für Objekte desselben Typs in der Präsentations-Informationsbasis (PIB) dient. Ein Beispiel gibt Abbildung 9.

Verfügt eine PIB über alle dort abgebildeten Klassen, so kann eine Managementanwendung beispielsweise je nach Art oder Position der Ressource aus folgenden Repräsentationen für die grafische Darstellung eines Rechners (Host) zu wählen:

- `PIB.LRZ.Host` - Allgemeine Repräsentation für alle Rechner des Leibniz-Rechenzentrums (LRZ)
- `PIB.LRZ.Mail.Host` - Mailserver am LRZ
- `PIB.Standard.Host` - Standardrepräsentation, wenn keine andere Repräsentation gewählt wurde, oder eine ausgewählte nicht gefunden werden konnte.

Innerhalb jeder Domäne implementieren diese Klassen den gewünschten Ausschnitt aus der in

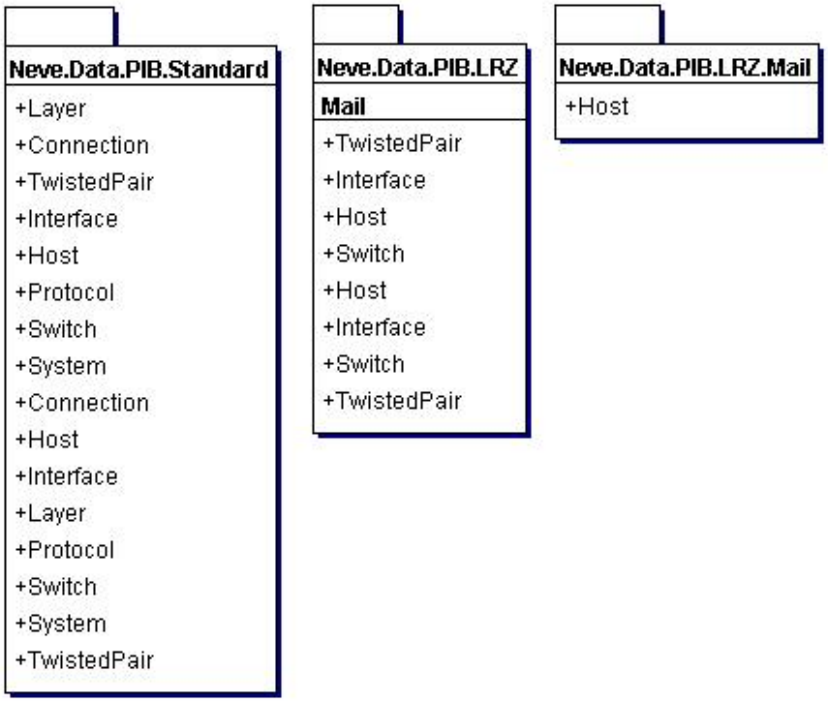


Abbildung 9: Paketdomänen zur Repräsentationswahl

Abbildung 8 abgebildeten Standard-Klassenstruktur. Sowohl die Domänenstruktur der PIB, als auch die Klassenstruktur sind durch Hinzufügen neuer Klassen leicht erweiterbar.

Eine weitere Aufgabe des Mappingservice ist die Bereitstellung von Zugriffsmöglichkeiten auf die Inhalte des dem Präsentationsraum zugrundeliegenden Informationsraumes. Hierzu stellt er die Funktion `getMapping(String MPORef)` zur Verfügung. Sie dient dazu, den Visualisierungsanwendungen in der Präsentationsschicht und den aktiven Methoden der MPOs den Zugriff auf die zugrundeliegenden MIOs zu ermöglichen, indem sie über das vorher erfolgte Mapping Aufschluss gibt und eine Referenz auf das einem MPO zugrundeliegende MIO zurückgibt.

3.5.3 Visualisierungsfunktionen

Visualisierungsfunktionen stellen an ihrer Managementschnittstelle Operationen zur Veränderung der grafischen Darstellung bereit. Hierfür benötigen sie Informationen über Ereignisse, die den Informations- oder den Präsentationsraum verändern. Sie melden dazu den Empfang von Notifikationen beim Server an. Dies ermöglicht es einer Visualisierungsfunktion, über die Neuregistrierung, Löschung und Veränderung von Objekten informiert zu sein oder in regelmäßigen Abständen aufgerufen zu werden. Alle Visualisierungsfunktionen können die Attribute aller MIOs und MPOs sowie die Relationen auslesen und auch Anfragen auf den Server nach Teilmengen

dieser Objekte absetzen. Sie verändern nur Objekte im Managementpräsentationsraum, greifen also nicht schreibend auf MIOs oder Relationen zu.

3.5.4 Layoutservice

Der Layoutservice trägt den Namen eines Dienstes (engl. service), da er eine so grundlegende Visualisierungsfunktion ist, dass er für jede Visualisierung notwendig ist. Er nimmt eine architektonische Organisation des Präsentationsraumes (VANF 12) vor, die entsprechend einer für seinen Aufgabenbereich definierten Layoutpolicy erfolgt (VANF 16). D.h. es wird definiert, welche Art von MPOs er bearbeiten soll, wann er tätig werden soll und wie und wo die MPOs positioniert werden sollen. Bestehen Abhängigkeiten oder Zusammenhänge zwischen mehreren Policies, so ist es empfehlenswert, alle in einem Layoutservice zu realisieren. Sollen mehrere unterschiedliche, voneinander unabhängige Layoutpolicies angewendet werden, so ist es auch möglich, mehrere verschiedene Layoutservices zu implementieren.

Ein Layoutservice hat über die Infrastruktur der Managementplattform Zugriff auf alle MPOs und deren Attribute sowie auf die Informationsstruktur. Wird diese verändert, so erhält er vom Relationservice eine Notifikation. Werden neue MPOs registriert oder alte gelöscht, so erhält er eine solche vom Server. Der Layoutservice ruft dann seine Operation `buildConnectionVectors()` auf, die das Layout vorbereitet, indem es eine Adjazenzmatrix erzeugt. Für jedes der in der Präsentationsmenge vorhandenen MPOs erfolgt in dieser Methode eine Anfrage an den Mappingservice nach dem zugrundeliegenden MIO. Am Relationservice werden die Relationen abgefragt, die es verknüpfen. Hierbei werden nur die betrachtet, die für die vorher definierte Layoutpolicy relevant sind. Mit dieser Information werden für jede betrachtete Relation zwei interne Vektoren aufgebaut. Der erste enthält Referenzen auf die entsprechenden MPOs, die die Knoten repräsentieren (beispielsweise Verbindungsendpunkte), der zweite Referenzen auf die entsprechenden MPOs, die die Kanten repräsentieren (beispielsweise Verbindungen).

Die Operation `doLayout()` berechnet auf der Grundlage der Vektoren und der im Einzelfall implementierten Layoutstrategie die neuen Positionen aller betroffenen MPOs. Anschließend werden die Positions-Attribute der MPOs auf die neuen Werte gesetzt. Der Layoutservice stellt nach außen die Operation `doLayout(int iterations)` bereit, die zunächst so viele Layout-Vorgänge per `doLayout()` durchführt, wie im Parameter angegeben sind. Erst dann erfolgt das Setzen der neu berechneten Koordinaten bei den Präsentationsobjekten. Dies verhindert, dass Änderungen von wenigen Pixeln propagiert werden und führt bei genügend hoher Iterationszahl zu ausgeglichenen Darstellungen, ohne dass der Anwender das Layout im Einzelnen verfolgen muss. So wird eine hohe Prozessorlast vermieden, die in der Folge durch die Berechnung der Visualisierungen für die Vielzahl an Änderungen entstehen würde.

Jeder Layoutservice hat aber darüberhinaus die Möglichkeit, sich beim Timerservice anzumelden. In regelmäßigen, am Layoutservice per Attributwert einstellbaren, Zeitabständen sendet dieser dann eine Notifikation, woraufhin die Operation `doLayout()` und damit der eigentliche Layoutvorgang ausgelöst wird. Diese Methode erlaubt ein ständiges Layout im Hintergrund,

führt aber nur bei kleinen Netzen zum Erfolg, da sie sehr viel Rechenzeit benötigt und oft auch ein unschönes Wackeln bei der Darstellung hervorruft.

Für die Durchführung des Layouts in der Methode `doLayout()` sind eine Vielzahl von Algorithmen und Strategien denkbar. Eine gute Auswahl findet sich beispielsweise bei [TBET 99] und mit [KuFo 96] liegt ein guter Algorithmus für dreidimensionales Layout vor. Welcher Layoutalgorithmus vom jeweiligen Layoutmanager in der Implementierung tatsächlich eingesetzt wird, entscheidet die jeweilige Layoutpolicy.

In der Anforderungsanalyse wurde für die vorliegenden Anwendungsfälle mit VANF 15 eine für das Szenario der Visualisierung der Netztopologie auf der Basis von Kommunikationsarchitekturen grundlegende Policy festgestellt. Sie gilt für alle MPOs und schreibt vor, dass die geographische Lage der Präsentationsobjekte anhand ihrer Verbindungen bestimmt werden soll. Mit Hilfe der Vektoren soll deshalb festgelegt werden, welches Präsentationsobjekt mit welchem verbunden ist. Mit diesen Daten als Basis und Anfragen an den Relationservice nach Verbindungen mit dem Typ "verbunden-mit" können so beispielsweise mit einem Spring-Embedding-Algorithmus die Positionen der Objekte innerhalb der horizontalen Ebene bestimmt werden.

Zu beachten ist, dass die in den Attributen der MPOs vorgesehenen Koordinaten immer als relative Koordinaten zu eventuell in der Enthaltenseinshierarchie höherliegenden MPOs zu betrachten sind. Dies ist auf die Struktur des Szenegraphen zurückzuführen, die in Kapitel 3.6 näher betrachtet wird. Aus diesem Grund dürfen nur die in der Hierarchie der Enthaltenseinsrelation ganz oben gelegenen MPOs vom Layoutservice positioniert werden, während die anderen MPOs später vom Visualisierungsservice in den Szenegraphen unterhalb der so berechneten Knoten eingefügt werden und die für die oberen Knoten berechnete Position erben.

Weiter gilt es den Fall zu betrachten, dass ein System aus mehreren Teilen bestehen kann, die alle in einem einzigen System enthalten sind[23]. D.h. beispielsweise verfügt ein Router über mehrere Interfaces. Es muss hier unter Umständen eine geographische Positionierung der einzelnen Teile des Systems in der horizontalen Ebene erfolgen, damit sich deren grafische Darstellungen nicht überschneiden. Dies kann als eigenständige Layoutpolicy betrachtet und durch die Implementierung eines weiteren Layoutservice realisiert werden.

[23] vgl. die Definition der Enthaltenseinsrelation in Abbildung 6

3.6 Visuelle Abbildung

Die letzte der durchzuführenden Transformationen ist die der visuellen Abbildung. Sie wird vom Visualisierungsservice vorgenommen, der aus den MPOs ein von 3D-Renderern lesbares Modell erzeugt, indem es Einzelteile eines sog. Szenegraphen generiert und den Benutzer-Interfaces zum Abruf bereit stellt. Ferner informiert der Visualisierungsservice mit Hilfe von Notifikationen alle registrierten Benutzer-Interfaces über Status und Änderungen am Szenegraphen.

Definition 3.26 *Ein Szenegraph ist ein gerichteter, azyklischer Graph, der die Struktur einer darzustellenden Szene definiert. Der logische Aufbau der darzustellenden Objekte wird auf eine gleichartig aufgebaute, baumähnliche Struktur, die im wesentlichen aus Definitionen von Transformationen und Geometriedaten besteht, abgebildet. Diese Struktur wird von den 3D-Routinen für die grafische Ausgabe verwendet.*

Der Szenegraph ist das abstrakte Modell für die präsentierte Szene. Oberflächenprozesse visualisieren schließlich die Szene dreidimensional auf den entsprechenden Medien wie beispielsweise einem Webbrowser oder immersiver VR-Hardware.

Liegt der Szenegraph in einem definierten, von allen beteiligten Komponenten lesbaren Format vor, so kann er auch über das Abbildungs-Interface vom Visualisierungsservice an die Clients übertragen werden. Es gibt verschiedene Möglichkeiten, einen solche Graphen syntaktisch zu beschreiben. 3DAPIs, wie die Virtual Reality Markup Language (VRML), stellen solche Strukturen (im Sinne von Beschreibungssprachen) bereit. Leider ist VRML sehr von Browsern abhängig und bietet nur eingeschränkte Möglichkeiten zur Interaktion und zur Weiterverarbeitung durch Software außerhalb eines VRML-Browsers, so dass es prinzipiell zwar genügend Funktionalität für den Einsatz in einem Abbildungs-Interface bietet, in der Praxis aber etwas umständlich zu handhaben ist. Außerdem bietet es nur wenig Kontrolle über das Rendering und eine Umsetzung der beschriebenen Szenen in tatsächliche Visualisierungen muss über einen Parser erfolgen und ist deshalb teilweise langsam. Eine bessere Performanz und ein größerer Beschreibungsumfang kann durch den Einsatz von Java3D erreicht werden. Dessen Geschwindigkeit ist mittlerweile höher geworden und man kann davon ausgehen, dass es weiterhin gepflegt und optimiert wird. Es entfällt dabei auch eine Übersetzung der Szenebeschreibung in eine Textform, da alle eingesetzten Objekte von Anfang an vom Renderer lesbar sind. Die Programmiersprache bietet darüberhinaus die Möglichkeit, Teile von Szenegraphen byteweise zu übertragen, neuere Erweiterungen erlauben sogar eine Reduzierung deren Größe durch Komprimierung, was sich vor allem auf die Performanz bei einer Übertragung über ein Rechnernetz positiv auswirkt. Aus diesen Gründen wird im Folgenden von einem Java3D-System ausgegangen. Prinzipell dürfte das Gesagte aber auch mit VRML oder mit anderen Beschreibungsmethoden für 3D-Systeme anwendbar sein.

3.6.1 Visualisierungsservice

Der Visualisierungsservice generiert aus dem Präsentationsraum das Modell, das der Ansicht im GUI (Graphical User Interface - grafisches Benutzer-Interface) zugrundeliegt. Er hat die

folgenden Aufgaben:

- Generierung und Aktualisierung der Teile einer Szene bzw. eines Szenegraphen
- Registrierung der Oberflächenprozesse
- Bereitstellung der fertigen Szenedaten für die Oberflächenprozesse.

In dieser Komponente wird das Renderering vorbereitet, durchgeführt wird es aber auf dem Benutzer-Interface als Client. Auch die Navigation durch die dargestellte Szene wird dort ausgeführt. Einerseits soll möglichst viel Funktionalität auf den Client verlagert werden, um den Server zu entlasten, andererseits gilt es aber, aus Sicherheitserwägungen datenkritische Berechnungen auf dem Server zu belassen. Abbildung 10 stellt die Operationen für den Austausch der Grafikdaten und die Ansteuerung der Clients dar.

Der Visualisierungsservice implementiert das Visualisierungs-Interface, über das die Oberflächenprozesse ihre Daten erhalten. Seine Hauptfunktionalität ist es, die von den MPOs definierten Subgraphen entsprechend der Enthaltenseinsrelation zu strukturieren und damit einen Szenegraphen zu definieren. Dazu wird für die transitive Hülle jeder Enthaltenseinsrelation und für alle Objekte, die nicht an einer Enthaltenseinsrelation beteiligt sind, ein Teilbaum des Szenegraphen generiert. Werden alle diese Teilbäume an eine Wurzel gehängt, so bilden sie die gesamte Beschreibung der Szene, den Szenegraphen, der von einem Renderer dargestellt werden kann. Im Folgenden soll ein solcher Teilbaum Cluster genannt werden.

Definition 3.27 *Ein Cluster sei ein Teilbaum eines Szenegraphen unterhalb dessen Wurzel.*

Um den Szenegraphen mit seinen Clustern zu generieren, kann folgende Vorgehensweise angewendet werden:

- `handleNotification(Notification notification, Object handback)` erhält vom Server Notifikationen, sobald neue Objekte registriert bzw. deregistriert wurden. Ferner erhält diese Routine Notifikationen von MPOs, sobald sich deren Attribute verändert haben und Informationen über Änderungen an den Relationen.
- Wurde ein MPO neu registriert, so hänge dessen Graph zunächst oben in den Baum als neues Cluster ein.
- Wird ein MPO gelöscht, so entferne es aus dem Baum. Lösche das gesamte Cluster, wenn es der letzte Knoten darin war.
- Reorganisiere den Baum, sobald sich Enthaltenseinsrelationen verändern. Hänge die Knoten so um, dass der Baum immer die Struktur der Enthaltenseinsrelationen wiederspiegelt und lösche dabei eventuell leere Cluster.
- Verändern sich die Attribute eines MPOs, so tausche den entsprechenden Knoten im Szenegraphen gegen die aktuellen Daten aus.

MPOs, die direkt in anderen enthalten sind, hängen innerhalb eines Clusters immer als Blätter unter diesen. Die MPOs, die in keinem anderen enthalten sind, bilden jeweils ein eigenes Cluster.

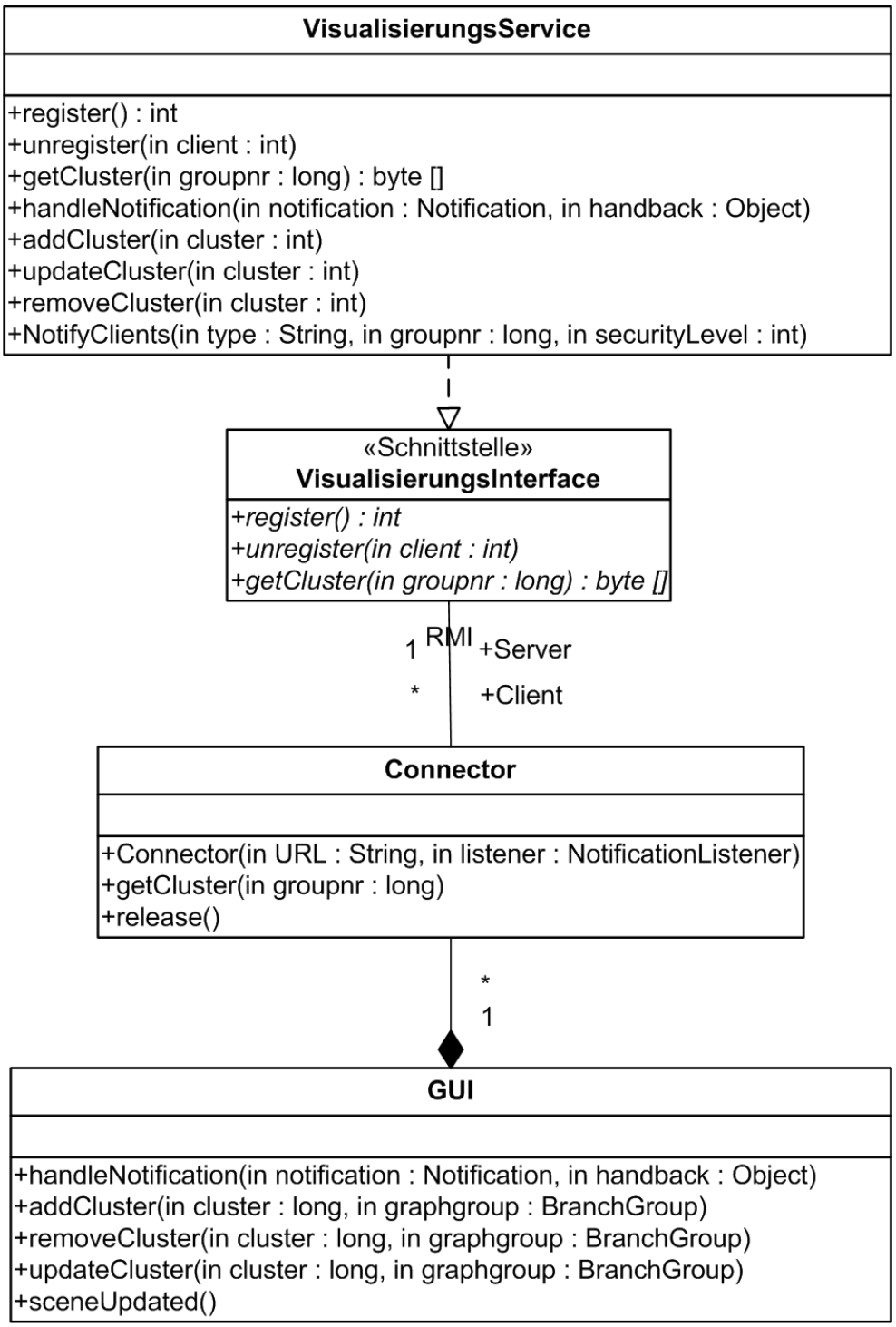


Abbildung 10: Die Dienste zur Visuellen Abbildung

Jeder Knoten in einem Cluster besteht dabei aus den in den Attributen der ihm entsprechenden MPOs beschriebenen Transformationen und Shapes. Entsprechend werden die Transformationen (Drehungen), Translationen (Verschiebungen), Farben und Shapes (grafische Form, Polygonbeschreibungen) in der Szene-Beschreibungssprache des Renderers generiert.

Mit Hilfe von Notifikationen werden die Clients über Änderungen an der Clustermenge informiert. Der Visualisierungsservice sendet vier Arten von Notifikationen, die jeweils aus einem Notifikationstyp bestehen und einem Referenzidentifikator, der das Cluster identifiziert. Folgende Notifikationstypen gibt es:

- "OLD" - Dieses Cluster ist bekannt, die Nachricht dient neu registrierten Clients zum Aufholen der Daten.
- "UPD" - Das Cluster wurde verändert.
- "KLL" - Das Cluster wurde gelöscht.
- "ADD" - Das Cluster wurde neu angelegt.

Wird ein Client neu beim Visualisierungsservice registriert, so werden alle IDs der ihm bekannten Cluster mit der Nachricht "OLD" an die Clients verschickt. Alle Clients, die das Cluster mit der betreffenden ID noch nicht kennen, holen sich mit Hilfe der Connectorfunktion `getCluster(int groupnr)` das entsprechende Cluster und fügen es sowohl ihrer Datenbasis als auch der aktuell gerenderten Szene hinzu. Diese Funktion wird auch bei der Aktualisierung der von der Notifikation mit den Typen "UPD" oder "ADD" referenzierter Cluster aufgerufen.

Es kann vorkommen, dass in sehr kurzer Zeit Änderungen an mehreren MPOs durchgeführt werden, beispielsweise während des Layouts. Dies kann dazu führen, dass die Änderungen nicht schnell genug über die Leitung übertragen werden und die Clients mit der Aktualisierung der Darstellung nicht nachkommen. Deshalb wird vor dem Ändern des Baumes und der anschließenden Client-Notifikation ein Ringbuffer zwischengeschaltet. Trifft eine Notifikation ein, nachdem ein MPO verändert wurde, wird zunächst eine Referenz auf das MPO in einem Ringbuffer zwischengespeichert. Aus diesem werden dann in einem einstellbaren Intervall diese Referenzen wieder ausgelesen, erst dann erfolgt die Aktualisierung des Szenegraphen und die Notifikation an die registrierten Benutzer-Interfaces. Mit diesem Mechanismus ist es möglich, unabhängig von der gegenwärtigen Serverauslastung einen konstanten Datenfluss an die angeschlossenen Clients aufrecht zu erhalten.

Was die Sicherheit angeht (IANF 1), so kann der Visualisierungsservice dahingehend erweitert werden, dass er bestimmte Cluster nur an Clients mit entsprechenden Rechten weitergibt. Somit kann beispielsweise verhindert werden, dass die Infrastruktur um eine Firewall nur von den Administratoren, nicht aber von externen Benutzern betrachtet werden kann. Hierzu könnte den MPOs ein Attribut "Sicherheitsstufe" mitgegeben werden. So kann der Visualisierungsservice später entscheiden, welche Clients über das Cluster informiert werden und welche nicht. Ist die Sicherheitsstufe des Client höher als die des MPOs, so erhält er Nachrichten über das Cluster, ansonsten nicht.

3.6.2 Graphisches Benutzer-Interface

Das Model-View-Controller (MVC)-Paradigma aus [KrPo 88] hat sich für die Programmierung grafischer Benutzer-Interfaces gut etabliert. Es teilt eine Präsentation auf in die drei Teile "Model" (Datenmodell), "View" (Ansicht des Datenmodells) und "Control" (Kontrolleinheit).

[PaPh 01] empfiehlt eine Verlagerung des Models auf den Server, wenn Umgebungen mit mehreren Views auf dasselbe Modell gewünscht sind. Im hier vorgestellten Modell ist das Benutzer-Interface für den View und den Control-Teil zuständig, das Modell liefert der Visualisierungsservice auf dem Server. So werden mehrere unabhängige Oberflächenprozesse ermöglicht. Jedes GUI instantiiert auf Clientseite einen Connector, der sich zum Server verbindet und für die Übertragung der Model-Daten vom Server zum Client zuständig ist. Außerdem bietet das GUI für den Erhalt von Notifikationen eine Operation `handleNotification(Notification n, handback h)` an, und übergibt diese dem Connector als Parameter. Sie erhält dann, sobald sich die Szene verändert hat, Notifikationen vom Visualisierungsservice und kann über den Connector die betroffenen Cluster anfordern. Es ist notwendig, dass die vom GUI empfangenen Cluster gemeinsam mit ihren IDs gespeichert werden, um später die Identifikationsnummern in den Notifikationen mit den richtigen Clustern assoziieren zu können. Die einzelnen gespeicherten Cluster werden letztlich an das vom Renderer verwaltete "Universum" angehängt und der Renderer zeigt die gesamte Szene an.

3.6.3 Connector

Der dem Oberflächenprozess vorgelagerte Connector als Schnittstelle zwischen dem Model- und dem View-Controller-Teil macht die geforderte Medienunabhängigkeit (VANF 8) möglich. Er entkoppelt das Visualisierungssystem von den Benutzer-Interfaces und erlaubt dadurch die simultane Verwendung mehrerer verschiedener Anzeigeprogramme auf verschiedenen Systemen (AANF 3). Dieser Zugriffspunkt dient außerdem dazu, dass der View-Controller-Teil weitgehend unabhängig vom Visualiser implementiert werden kann. Der Connector stellt die Verbindung zwischen dem Programmteil für den Verbindungsaufbau sowie dem NotificationListener (der Programmteil, der Notifikationen in Empfang nimmt) auf der Client-Seite und den Funktionen des Visualisierungsinterface auf der Server-Seite her. Dies kann beispielsweise über RMI (Remote Method Invocation) geschehen.

3.7 Interaktion

Schon in den Anforderungen wurde auf Möglichkeiten zur Interaktion Wert gelegt. Speziell wurden dabei die folgenden Interaktionsobjekte identifiziert:

- IANF 2: Interaktion mit der grafischen Repräsentation (dem MPO) und den Daten (dem MIO)
- IANF 3: Interaktion mit Managementanwendungen
- IANF 3: Interaktion mit Repräsentationen von Ressourcen, die andere Repräsentationen verändern können (MPOs und Funktionen)
- IANF 6: Interaktion mit der dargestellten Szene als Ganzem, Navigation (View im GUI)

Wie man sieht, muss also auf alle Objekte in der Pipeline zugegriffen werden können. So kann jeweils mit einer anderen Ebene der Visualisierung interagiert werden. In [CMS 98] wird ein Interaktionsmodell für Informationsvisualisierung vorgestellt, das sich auf den hier diskutierten Fall anwenden läßt. Abbildung 11 zeigt, sinngemäss daran anlehnend, die einzelnen Ebenen der Visualisierungspipeline und die Interaktionsmöglichkeiten, die sich dort ergeben. So kann der Benutzer auf die Managementanwendungen, die Daten, die Art und Weise der Darstellung, die Funktionen und seine Sicht auf den Raum einwirken.

Soll beispielsweise die Zugehörigkeit eines Switchports zu einem VLAN geändert werden, so wird auf das MIO zugegriffen und dessen Attribut VLAN geändert. Der Anwendungsservice stellt sicher, dass die Anwendung, die das MIO erzeugt hat, über die Änderung informiert wird. Diese verändert dann per SNMP oder ein herstellerabhängiges Konfigurationsprotokoll die Konfiguration am Switch.

Wenn dagegen eine Veränderung der Visualisierung gewünscht ist, beispielsweise eine andere Kennzahl die Breite der Darstellung eines Verkehrsflusses bestimmen soll, so erfolgt die Interaktion mit den Attributen des entsprechenden MPO. Dieses berechnet dann seine Grafikdaten anders und liefert die Änderungen an den Visualisierungsservice.

Soll eine Managementanwendung beispielsweise Teile eines Netzes untersuchen und diese an das Visualisierungssystem weitergeben, so können die Adressen der gewünschten Komponenten durch direkte Interaktion mit der Managementschnittstelle der Anwendung übergeben werden.

Alle Interaktionen finden also über die Managementschnittstellen der Komponenten statt. Das in der Zeichnung als InteraktionsInterface bezeichnete Objekt ist dabei eine Schnittstelle zum GUI. Es unterstützt einen entfernten Zugriff auf die Managementschnittstellen der auf dem Server gespeicherten Objekte. So wird es möglich, auf deren Attribute und Operationen lesend und schreibend bzw. auslösend einzuwirken. Es wird vorgeschlagen, hierzu einen XML-Mechanismus (Extended Markup Language, dient zum einheitlichen Datenaustausch) zu verwenden, der dann vom GUI per HTTP (Hypertext Transfer Protocol) oder RMI angesteuert wird. Diese Schnittstelle kann vollständig in den Visualisierungsservice und den Connector integriert werden, oder als separater Teil realisiert werden.

Einige Managementplattformen stellen solche Schnittstellen nach außen bereits zur Verfügung, weshalb hier keine detailiertere allgemeine Spezifikation angegeben werden kann. Ein Beispiel, wie die Integration in eine JMX-Umgebung stattfinden kann, findet sich im Kapitel 4.3.6.

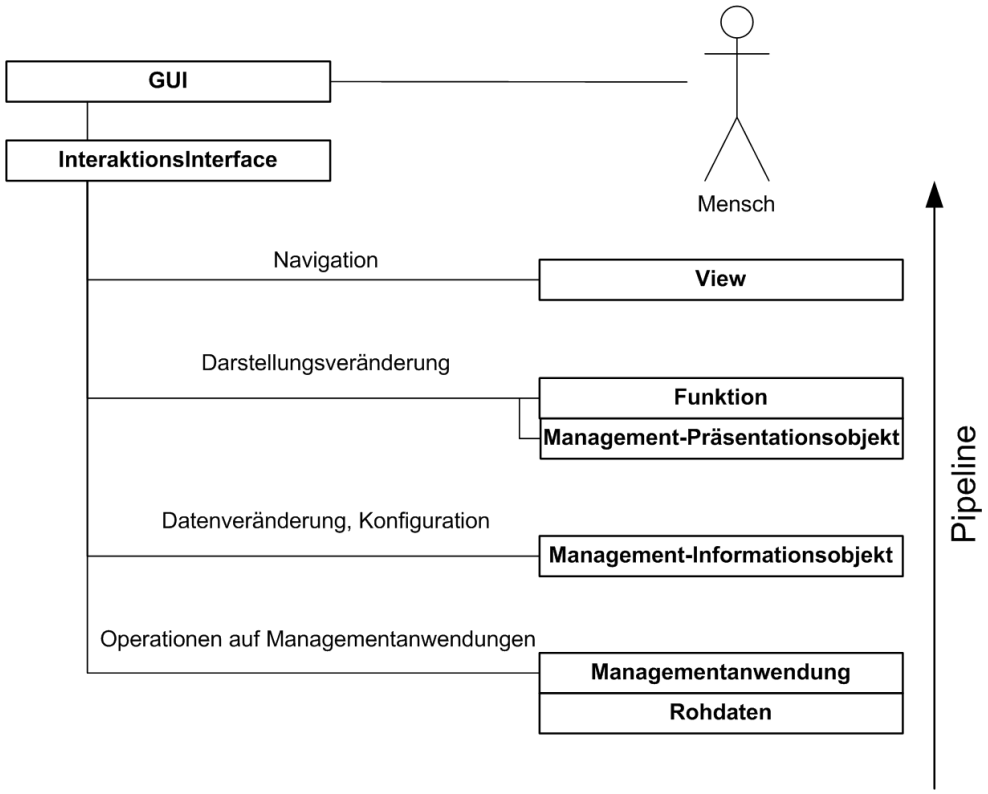


Abbildung 11: Interaktionsmodell

Mit der Definition der Klasse `InteraktionsInterface` sind nun auch die Forderungen nach einer oberflächenintegrierten Interaktion erfüllt:

- AANF 2: Oberflächenintegration der Managementanwendungen, homogene Bedienung
- IANF 5: Homogene Eingabemöglichkeit an die Managementanwendungen
- IANF 4: Homogenes Starten von Operationen der Managementanwendungen

Das Interaktions-Interface ist auch der geeignete Platz für Sicherheitsmechanismen (IANF 1). Hier müssen Benutzeranfragen auf ihre Zulässigkeit hinsichtlich der Sicherheitspolicy untersucht und im Einzelfall bearbeitet oder abgewiesen werden.

3.8 Bewertung des Ansatzes

3.8.1 Erfüllte Anforderungen

Der hier vorgeschlagene Ansatz konnte alle der in Kapitel 2.3 gesammelten Anforderungen erfüllen. Dabei konnte ein Großteil der zur Realisierung der Anforderungen notwendigen Mechanismen in das Rahmenwerk integriert werden. So wurden alle im Anforderungskatalog genannten Anforderungen an die Architektur (AANF) berücksichtigt: Die Funktionalität des Visualisierungssystems wird in eine Managementplattform mit den genannten Voraussetzungen integriert (AANF 1). Dabei findet eine Oberflächenintegration (AANF 2) der mit dem Visualisierungssystem arbeitenden Managementanwendungen statt. Sie erhalten die Möglichkeit, über eine einheitliche Schnittstelle auf das Visualisierungssystem zuzugreifen (AANF 4). Alle Operationen und Strukturen, die vom Design verwendet werden, können darüberhinaus plattformunabhängig implementiert werden (AANF 3).

Alle im Katalog genannten Anforderungen aus dem Management (MANF) wurden ebenfalls umgesetzt. Die eben erwähnte Schnittstelle wurde flexibel und sehr allgemein definiert, um von den Managementanwendungen alle notwendigen Informationen über die allgemeine Netzkonfiguration, die Netzkomponenten und deren Eigenschaften zu erhalten (MANF 1). Mit demselben Mechanismus können auch unaufgeforderte Ereignismeldungen wie z.B. Alarme oder Ereignisberichte von den Ressourcen des Netzes empfangen werden (MANF 3). Die Anforderung von Einzelinformationen zu den Ressourcen erfolgt dabei allerdings nicht, wie Anforderung MANF 2 auch verstanden werden könnte, aus der Visualisierung heraus, sondern aus der Managementanwendung heraus. Dies schafft eine klare Trennung der Visualisierung von den Daten des Netzes und ermöglicht die Berücksichtigung spezieller Sicherheitsmechanismen. Die Daten werden dabei in der Art und Weise eines "push"-Verfahrens über die Anwendungsschnittstelle (das `AnwendungsServiceInterface`) an die Visualisierung weitergegeben. In der internen Datenhaltung für die Visualisierung (Abbildung 7 und Abbildung 8) wurden außerdem Kommunikationsarchitekturen als grundlegendes Konzept des Aufbaus von Rechnernetzen berücksichtigt (MANF 4).

Die umfangreiche Sammlung von Anforderungen zur Visualisierung (VANF) wurde teils direkt in das Rahmenwerk integriert, teils wurden Mechanismen geschaffen, um deren einfache Implementierung zu ermöglichen und zu unterstützen. So wurde der Vorgang zur Erzeugung und Löschung grafischer Repräsentationen formalisiert und die dazu notwendigen Objekte definiert. Dazu gehört auch ein Mapping-Dienst, der diese Aufgabe automatisch wahrnimmt. Für physische und logische Ressourcen wurden in Form von Managementinformations- und Managementpräsentationsobjekten grundlegende Datenstrukturen geschaffen, die nach Bedarf verwendet und erweitert werden können (VANF 1). Die MPOs sind so konzipiert, dass sich ihr grafisches Verhalten in Abhängigkeit der Parameter des zugehörigen Managementobjektes ändern kann, sofern es für eine Visualisierung notwendig erscheint (VANF 2). Darunter fällt auch ein Ein- und Ausblenden von Ressourcen (VANF 20). Alle Objekte des Systems sind darüber hinaus für eine dynamische Bearbeitung ausgelegt, so dass während der Laufzeit eine stets aktuelle Visualisierung erfolgen kann (VANF 3). Werden also die Struktur oder Teile der Daten

verändert, so ändert sich auch die Visualisierung.

Durch Mechanismen zur Generation von Szenegraphen und die Integration der zur Beschreibung dreidimensionaler Darstellungen notwendiger Objekte in das Datenmodell des Systems (VANF 5) wurde eine dreidimensionale Darstellung ermöglicht (VANF 4). Durch geeigneten Einsatz der bereitgestellten Möglichkeiten ist auch die Darstellung von Texturen und 3D-Modellen machbar (VANF 10). Das verwendete Datenmodell dient dabei sowohl als Basis zur Speicherung der Netzressourcen (VANF 6) als auch als Basis zur Speicherung und Vorbereitung der aktuellen Visualisierung. Es kann von verschiedenen Managementanwendungen gespeist werden (VANF 7) und die Ausgabe der erzeugten Visualisierung kann auf verschiedene Medien geschehen (VANF 8). Dabei wird die dargestellte Szene automatisch generiert (VANF 9) und es erfolgt eine architektonische Organisation des darzustellenden Informationsraumes (VANF 12).

Durch die Trennung in Visualisierungsobjekte und -funktionen werden Dienste für verschiedene Darstellungen ermöglicht, die den Anwendungsgebieten angepasst werden können. So kann beispielsweise der Layoutservice ausgetauscht werden oder verschiedene Layouter gleichzeitig eingesetzt werden, die verschiedene Layoutverfahren und Heuristiken anwenden (VANF 16). Gemeinsam mit der Flexibilität des verwendeten Informationsmodells und dessen Objekten ermöglicht das die Darstellung verschiedenster logischer Konzepte (VANF 11). Beispielhaft wurde ein Informationsmodell entworfen, das die Prinzipien der OSI-Kommunikationsarchitektur berücksichtigt sowie Netzvisualisierungen anhand von Systemschnitt, Dienstschnitt und Protokollschnitt (VANF 13 bis 15) durchführt. Beim Layout und der Visualisierung werden dabei die zwischen Managementobjekten vorhandenen Relationen Adjazenz (VANF 17) und Enthaltensein (VANF 18) berücksichtigt. Die definierte Adjazenzrelation unterstützt Unicast- und Multicast-Verkehrsflüsse (VANF 19), Broadcasts können durch ,,intelligente" PDU-Repräsentationen implementiert werden.

Die Anforderungen zur Interaktion wurden ebenfalls vollständig umgesetzt. Mit Hilfe des spezifizierten Interaktionsmodells konnte eine Oberflächenintegration der Interaktionen mit allen verwendeten Diensten und Daten erreicht werden. Dies ermöglicht einem Benutzer, abhängig von seinen Zugriffsrechten (IANF 1) mit den Rohdaten und Visualisierungen der Ressourcen (IANF 2) aber auch mit den Visualisierungsdiensten (IANF 3) und Managementanwendungen (IANF 4) zu interagieren. Die Managementanwendungen beziehen dabei ihre Eingaben ebenfalls wie die Visualisierungsobjekte und Dienste über ihre Attribute, die von außen gesetzt werden können (IANF 5). Der Oberflächenprozess stellt darüber hinaus Möglichkeiten zur Navigation durch die erzeugte Darstellung bereit (IANF 6).

3.8.2 Nachteile

In der vorliegenden Fassung des Entwurfes ist die Performanz noch verbesserungsfähig. Die vom Visualisierungssevice und dem Benutzer-Interface verwendeten Szenegraphen-Cluster sind zwar für erste Ergebnisse zweckmäßig, aber nicht optimal. Die Übertragung des gesamten Clusters nach einer Änderung eines einzelnen im Cluster positionierten Knotens kann in vielen Fällen redundant sein, da sich oft nicht das gesamte Cluster ändert, sondern nur ein einzelner Baumknoten.

Dies führt zu einem Overhead, der bei komplizierteren grafischen Darstellungen (vor allem bei Modellen mit Texturen) und größeren Netzen die Geschwindigkeit stark herabsetzen kann. Es ist deshalb für einen über einen Prototypen hinausgehenden Einsatz des Systems notwendig, das Übertragungsprotokoll zwischen Visualisierungsservice und Benutzer-Interface dahingehend zu optimieren. Dies sollte so erfolgen, dass nur die tatsächlich geänderten Teile des Szenegraphen neu übertragen werden. Bei einem Großteil der während der Laufzeit veränderten Werte handelt es sich um Positionsinformationen, d.h. dass sich lediglich die TransformGroups, jeweils repräsentiert vom Attribut Transform in den MPOs, ändern. In diesem Fall sollten deshalb auch nur die betroffenen TransformGroups und nicht etwa die komplette Repräsentation des MPO in der Beschreibungssprache des Renderers übertragen werden. Eine Einschränkung der Definition der Cluster auf die Attribute Shape und Transform der MPOs als Bestandteil der Knoten im Szenegraphen sowie eine andere Adressierung veränderter Cluster sollte hier Abhilfe schaffen.

Ebenfalls als Nachteil könnte man die Tatsache verstehen, dass das Design für Verkehrsflüsse keine Broadcasts vorsieht. Hierzu sei allerdings bemerkt, dass eine Implementierung von Broadcasts mit dem vorliegenden Design durchaus möglich ist, aber eine interne Darstellung des Broadcasts als Multicast erfordert. Die hierfür notwendige Festlegung der Empfänger des Broadcastes könnte dabei dynamisch durch eine Visualisierungsfunktion erfolgen.

3.8.3 Vorteile

Neben der Berücksichtigung aller Anforderungen bietet das Design einige weitere Vorteile. Der Ansatz geht von einem allgemeinen Visualisierungsmodell aus, was eine einfache Erweiterbarkeit und Anpassung auf verschiedenste Problemstellungen ermöglicht. Das Data State Reference Model setzt einen Rahmen, in den je nach Anwendungsfall neue Visualisierungstechniken hinzugefügt, oder bereits integrierte angepasst und verändert werden können. In [Chi 00] konnten beispielsweise Visualisierungstechniken aus 36 Projekten in dieses Visualisierungsmodell eingeordnet werden.

Das verwendete Datenmodell für die Beschreibung der Vorgänge im Rechnernetz, die Managementinformationsbasis, ist ebenfalls sehr allgemein gehalten. Dies kommt der Anpassbarkeit und Erweiterbarkeit des Systems zugute. Durch die Präzisierung von Basisklassen sind die zur Weiterverarbeitung notwendigen Attribute definiert. Die einzelnen Objekte selbst können im Einzelfall um weitere Attribute erweitert werden, je nachdem, um welche Art von Ressource es sich handelt und welche Daten notwendig sind. Auch ist es möglich, bereits aus anderen Managementkontexten vorhandene MIBs in dieses Format zu konvertieren. Durch die Verwendung von Relationen ergibt sich ein hoher Freiheitsgrad zur Beschreibung der Verknüpfungen zwischen den Objekten in der MIB, die für verschiedenartige Visualisierungen, aber auch für die durchzuführenden Managementaufgaben, herangezogen werden können.

Im Vergleich zu anderen Ansätzen (vgl. Cybernet) beschreibt das Datenmodell dadurch nicht von vornherein Bäume, sondern läßt auch andere Arten von Beziehungen zu. Dies führt in der Folge zu einem flexibleren Mapping. Es muss nicht von vornherein ein statisches Mappingmodell gewählt werden, sondern es können durch Auswahl aus verschiedenen MPOs mehrere

Mappingtechniken gleichzeitig verwendet werden. Durch dieses allgemeine Daten- und Visualisierungsmodell ist eine Vielzahl von Anwendungsmöglichkeiten denkbar. So liefert es neben Mechanismen zur Visualisierung der geforderten Szenarien auch die Möglichkeiten zur Visualisierung der Zustände anderer Modelle und Entitäten. Es ist außerdem denkbar, nicht nur Kommunikationsarchitekturen darzustellen, sondern auch andere logische Modelle wie beispielsweise Agentensysteme, Petrinetze, Automaten, neuronale Netze o.ä. zu visualisieren.

Die grafischen Repräsentationen für die Netzkomponenten können außerdem personalisiert werden. Beispielsweise kann ein Router im MWN anders aussehen, als der Router eines kommerziellen Providers.

Das Design sieht zum jetzigen Zeitpunkt kein konkretes Sicherheitskonzept vor, definiert aber Schnittstellen, an denen ein solches vorgesehen ist. Außerdem wird sichergestellt, dass sensible Daten nicht extern verfügbar sind. Alle Daten bleiben innerhalb des Servers, eine Änderung von Daten im Benutzer-Interface führt zu keiner Änderung von Daten auf dem Server, solange dies nicht auf den vorgesehenen Wegen zur Interaktion geschieht. Dies stellt bereits zu einem frühen Zeitpunkt eine spätere Integration konkret definierter Sicherheitsmechanismen an der Interaktionsschnittstelle sicher, was spätere Probleme vermeiden soll.

Die Verteilung des Model-View-Control Design Patterns auf Client und Server ermöglicht den gleichzeitigen Zugriff mehrerer Benutzer auf denselben View. Dies ermöglicht ein gemeinsames Arbeiten mehrerer Fachleute an demselben Problem. Der Ansatz liefert damit eine Grundlage für einen eventuellen späteren Einsatz des Systems als Groupware.

Durch die klare Trennung der Objekte in einzelne funktionelle Schichten ist außerdem eine Verteilung des gesamten Systems denkbar. Beispielsweise könnten mehrere Server untereinander ihre Präsentations- und Informationsbasen austauschen. Dies würde dazu führen, dass nicht nur das Netz innerhalb der vom Server betrachteten Domäne dargestellt werden kann, sondern auch externe Netze. Bei Zugriffen dorthin könnte dann eine Darstellung der vom externen Visualisierungsserver betrachteten Infrastruktur erfolgen. Notwendig für die Erweiterung des Designs um solche Funktionalitäten wäre aber die Vergabe globaler Adressen für MIOs und MPOs und die Spezifikation von Übertragungsprotokollen zwischen den Servern.

4 Tragfähigkeitsnachweis

In diesem Abschnitt wird über die Implementierung berichtet und die Tragfähigkeit des Rahmenwerkes gezeigt. Schon vor Beginn der Arbeit wurde, auf Erfahrungen mit dem Systementwicklungsprojekt 3DNet [Baur 00] aufbauend, ein erster Prototyp entwickelt. Dieser diente u.a. dazu, die Möglichkeiten von Visualisierungspipelines zu erproben. Auf diese Vorarbeiten wird in Abschnitt 4.1 eingegangen.

Danach begann die Anforderungsanalyse und das anschließende Architekturdesign. So konnte schließlich ein Rahmenwerk bereitgestellt werden, um eine allgemeine Integration von Netzmanagementmechanismen in VR-Visualisierungen zu gewährleisten.

Zum Nachweis der Tragfähigkeit der entworfenen Architektur und zur Einschätzung ihrer Leistungsfähigkeit wurde, an den Entwurf anschließend, ein zweiter Prototyp realisiert. Der Sourcecode und das ausführbare Programm wurden unter einer Open-Source Lizenz freigegeben und finden sich im WWW unter http://neve.sourceforge.net.

Als Managementumgebung und Grundlage für die Implementierung wurde dabei JMX (Java Management Extensions) gewählt. Es erfüllt die vom Design vorausgesetzten Annahmen in Kapitel 3.3 und bietet viel Raum für die Entwicklung eigener Managementanwendungen. Einen Überblick über die grundlegenden Mechanismen von JMX gibt Kapitel 4.2 und die über das Rahmenwerk hinausgehenden Besonderheiten der Implementierung des zweiten Prototypen werden in Abschnitt 4.3 beschrieben.

Wie Versuche und Bewertungen in Kapitel 4.4 abschließend aufzeigen, kann der Prototyp die in der Anforderungsanalyse betrachteten Szenarien realisieren und alle im Anforderungskatalog in Abschnitt 2.3 verzeichneten Anforderungen erfüllen. Damit weist er die Tragfähigkeit des Architekturentwurfes nach und stellt gleichzeitig ein brauchbares Instrument für ein VR-Netzmanagement dar.

4.1 Vorarbeiten

Zum besseren Verständnis der Problematik und für die Erstellung erster Visualisierungsstudien wurde noch vor der Aufnahme der Anforderungsanalyse ein erster Prototyp in Java durch "Rapid Prototyping" erstellt. Der Prototyp erhielt den Namen NEVE (NEtwork Visualisation Environment). Als 3D-Engine wurde das frei erhältliche Anfy3D [anfy] verwendet, welches zunächst eine höhere Rendergeschwindigkeit als Java3D versprach. Später sollte sich aber zeigen, das Anfy3D über einige Features nicht verfügte, die für die vorliegenden Anforderungen dringend notwendig waren. So stellt es beispielsweise keine Grafikprimitive zur Verfügung, was die Programmierung der Metapherobjekte erschwerte. Ferner stand für Anfy3D kein fertiges Benutzer-Interface zur Verfügung, so dass alles selbst programmiert werden musste, dies hätte zur Konsequenz gehabt, dass fortschrittlichere Navigationsmöglichkeiten nur mit immensem Zeitaufwand realisierbar gewesen wären. Außerdem fehlten einfache Möglichkeiten der Übertragung der Grafikdaten über das Netz sowie eine unkomplizierte Möglichkeit des Zugriffs auf die Grafikbuffer einer SGI Workstation für eventuelle VR-Visualisierungen auf einer Holobench.

Der Prototyp war von seiner Grundkonzeption her monolithisch angelegt, d.h. die gesamte Visualisierungpipeline wurde durch mehrere Klassen realisiert, die direkt miteinander verwoben waren. Er basierte aber bereits auf einer Trennung in Informations- und Präsentationsobjekte, welche dort noch als Daten- und Metapherobjekte bezeichnet wurden. Im Unterschied zum zweiten Prototyp fand aber noch keine differenzierte Interaktion mit den Objekten statt, da eine Integration mit Netzmanagementumgebungen fehlte. Außerdem waren Managementanwendungen schwieriger einzubinden. Es gab auch keine Trennung in Benutzeroberfläche und Visualisierungsanwendungen, so dass der Prototyp für eine Mehrbenutzerumgebung ungeeignet war. Die Grenzen zwischen den funktionalen Schichten (Informationsraum, Präsentationsraum, View) waren nicht sehr scharf gezogen, so dass teilweise nicht ersichtlich wurde, welcher Teil der Software für das Rendering und welcher für die Assemblierung der Szene zuständig war. Schließlich wurde die Definition des zu visualisierenden Netzes durch Relationen nur begrenzt unterstützt. Dennoch lieferte der Prototyp bereits erste brauchbare Ergebnisse und war in der Lage, in Testanwendungen definierte Netze sauber anzuzeigen, sofern Strukturparadigmen vorgegeben waren, um festzulegen, auf welche Art und Weise das Layout stattfinden sollte. Außerdem wurde bereits in Ansätzen eine Schichtung von Kommunikationsarchitekturen berücksichtigt.

Abildung 12 zeigt ein Netz, das mit diesem ersten Prototypen erstellt wurde. Man sieht in der Vergrößerung die Darstellung eines Endsystems. Ganz unten ist als Ring die Netzwerkkarte dargestellt, die über eine physische Verbindung verfügt. Sobald man den Mauspfeil auf das dargestellte Objekt richtet, erscheint die IP-Adresse des Objektes. In der Mitte des Bildes, etwas erhöht, ist ein Router dargestellt. Drei Ringe kennzeichnen dort die Protokollinstanzen der unteren drei Schichten, von den beiden unteren gehen Verkehrsflüsse aus.

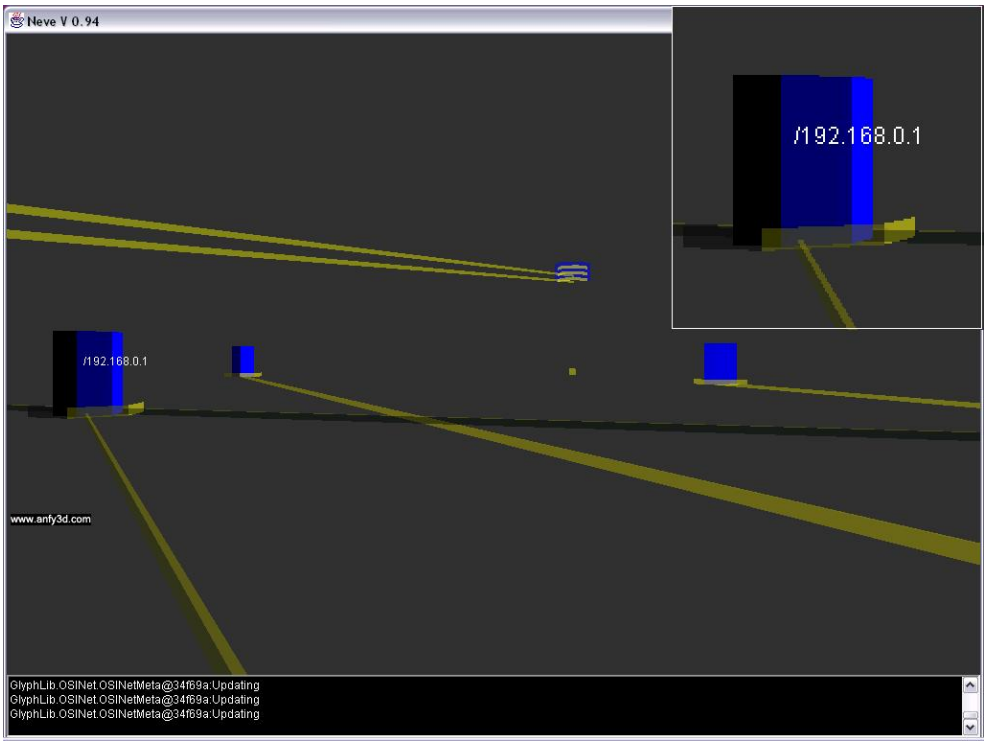


Abbildung 12: Screenshot des ersten Prototypen

4.2 JMX

Die auf Java basierende Komponentenarchitektur der Java Management Extensions (JMX) stellt die Grundlage des zweiten Prototypen dar. Dieses Kapitel gibt eine Einführung in die grundlegenden Mechanismen von JMX.

JMX verwendet den allgemeinen Ansatz einer dreistufigen Netzmanagement-Architektur, indem es die drei Ebenen Instrumentation Level, Agent Level und Distributed Services Level definiert (siehe Abbildung 13 aus [sun 00]).

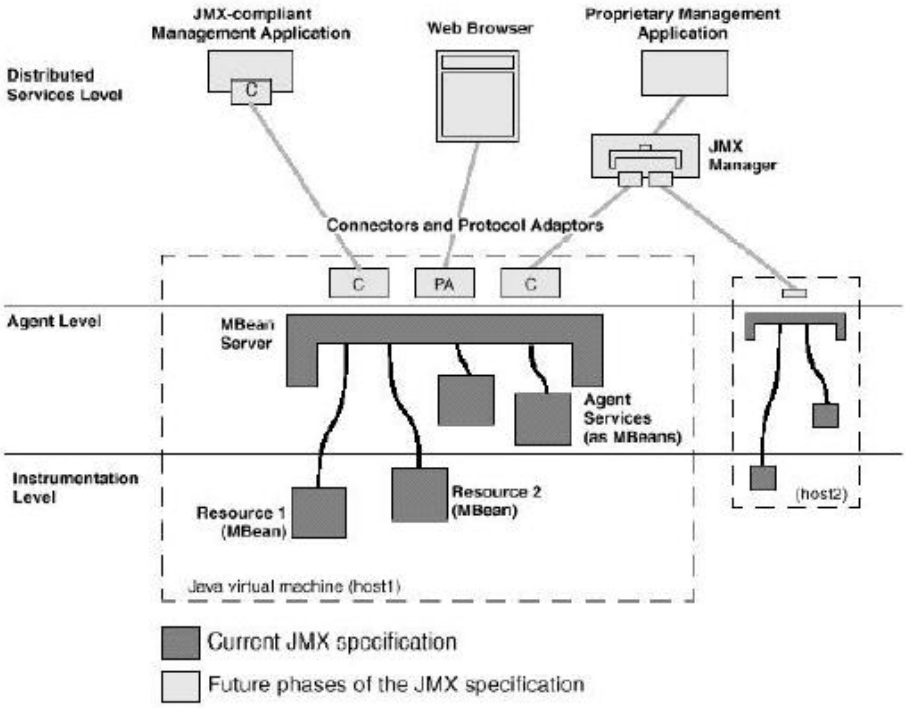


Abbildung 13: Die JMX-Architektur

Es ist zu erwarten, dass mit der Weiterentwicklung der JMX-Spezifikation Version 1.1 [sun 02] seitens Sun Microsystems auch die wenigen noch vorhandenen Schwachpunkte ausgeräumt werden, da zum gegenwärtigen Zeitpunkt der Distributed Services Level noch nicht fertig spezifiziert ist und keine ausgereiften Sicherheitsmechanismen zur Kommunikation bestehen. Version 1.5 verspricht diesbezüglich eine Vervollständigung und ist bereits in der Endphase der Entwicklung.

4.2.1 Instrumentation Level

Die Spezifikation des Instrumentation Level legt die Anforderungen fest, die eine Ressource bzw. ihr Repräsentant erfüllen muss, damit er über die JMX Architektur verwaltet werden kann. Die Implementierung dieses Repräsentanten wird in der JMX Terminologie Managed Bean

(MBean) genannt. MBeans sind die Kernkomponenten des Instrumentation Levels. Da lediglich die MBean-Spezifikation eingehalten werden muss, können Java-Entwickler ihre Produkte auf eine einfache Weise mit einer Management Schnittstelle versehen. Für sog. *Legacy-Systeme* (Altsysteme) ist es ausreichend, wenn sie mit einer Wrapperkomponente versehen werden, die die MBean-Spezifikation erfüllen. Die Managed Beans sind das Pendant zu den Managementobjekten in der Standardarchitektur.

In der Java Management Extensions Spezifikation wird ein MBean definiert als eine nicht abstrakte Klasse, die ein eigenes MBean Interface oder das Interface `DynamicMBean` und optional das Interface `NotificationBroadcaster` implementiert. Dadurch wird sichergestellt, dass ein JMX Agent in der Lage ist, dessen Managementschnittstelle zu erkennen.

Das MBean-Interface eines MBeans umfasst alle Attribute, Operationen, Benachrichtigungen und Konstruktoren, die der Agent kennen muss, um das MBean bzw. die zugrundeliegende Ressource zu verwalten. Die andere Möglichkeit, ein MBean zu erstellen, ist eine Java-Klasse zu definieren, die das `DynamicMBean` Interface implementiert.

Das Interface `DynamicMBean` umfasst sechs Methoden für den Zugriff auf die Managementschnittstelle. Im Gegensatz zu den Standard MBeans, deren Managementschnittstelle durch das eigene MBean Interface definiert wird, wird diese bei Dynamic MBeans zur Laufzeit über die Methode `getMBeanInfo` zurückgegeben. Diese Methode liefert ein MBeanInfo Objekt zurück, welches Informationen über die Managementschnittstelle enthält.

Das Lesen und Schreiben der Attributwerte erfolgt bei Dynamic MBeans für alle Attribute über die gleichen allgemeinen Methoden. Der Name des entsprechenden Attributs wird diesen Methoden als String-Parameter übergeben. Die allgemeinen Methoden sind dann für die Umsetzung auf das entsprechende Attribut verantwortlich.

Entsprechend erfolgt der Aufruf aller Operationen über die allgemeine Methode `invoke`. Dieser Methode wird der Name der auszuführenden Operation, die Parameter sowie die Signatur der Operation übergeben. Die Methode `invoke` sorgt dann intern für die Umsetzung auf die korrekte Operation.

Die MBeans des Visualisierungssystems sind als Dynamic MBeans konzipiert, da nur auf diesem Wege eine Vererbung von Attributen in der Art und Weise der Vererbung gewöhnlicher Javaklassen möglich ist.

4.2.2 Agent Level

Im Agent Level werden JMX Agenten definiert, deren Aufgabe es ist, die MBeans zu verwalten und zu steuern. Die JMX-Spezifikation beschreibt einen JMX Agent als eine Einheit bestehend aus einem MBean Server, einer Menge von MBeans, einigen Agent Services (Diensten, s.u.) und mindestens einem Protocol Adaptor oder Connector.

Der Kern jedes JMX Agents ist dabei der MBean Server. Er dient zur zentralen Registrierung der Managed Beans (Instrumentation Level) im Agenten und dient als Schnittstelle zwischen diesen

und den Managementanwendungen (Distributed Services Level). Jede Interaktion mit einem MBean, z.B. Lesen und Setzen von Attributen, Ausführen von Operationen oder Registrieren für den Empfang von Nachrichten, wird über den MBean Server abgewickelt. Darüber hinaus bietet der MBean Server die Möglichkeit, Abfragen nach seinen MBeans auf Basis des Objektnamens und der Attributwerte abzusetzen.

Bei seinen Aufgaben wird dieser durch eine Reihe sogenannter Agent Services unterstützt. Agent Services sind Objekte, die den MBean Server um Managementfunktionalitäten erweitern. Sie stellen Dienste zur Verfügung, die von MBeans zur Verrichtung ihrer Aufgaben genutzt werden können. Es ist empfehlenswert, Agent Services selbst als MBeans zu implementieren, da auf diese Weise gewährleistet werden kann, dass sie über eine JMX-Managementanwendung verwaltet werden können. Agent Services können sowohl durch die JMX Spezifikation definiert als auch von Drittanbietern vertrieben werden. Die JMX Spezifikation sieht Agent Services für einige Bereiche vor, dazu gehören ein Timer Service (schickt zu einstellbaren Zeiten Notifikationen), ein Relation Service (verwaltet Relationen zwischen MBeans) und ein Monitoring Service (überwacht Attribute von MBeans).

Der Prototyp startet beim Aufruf seiner Hauptklasse `baseAgent` einen Server und initialisiert die Visualisierungsservices (die die Rolle zusätzlicher Agent Services einnehmen) und die verwendeten Visualisierungsfunktionen und Managementanwendungen. Für spätere Versionen ist ein automatisches Einladen der verwendeten Anwendungen und Funktionen über eine Konfigurationsdatei oder automatisches Erkennen geplant.

4.2.3 Distributed Services Level

Auf der Ebene des Distributed Services Level werden die Schnittstellen zwischen dem MBean Server und den Managementanwendungen definiert. Diese Schnittstelle wird durch Connectors und Protocol Adaptors realisiert. Connectors realisieren die Kommunikation zwischen Agents und JMX-Managementanwendungen. Sie bestehen aus einem Connector Server, der auf der Seite des Agents eingesetzt wird, und einem Connector Client auf Seiten der Managementanwendung. Protocol Adaptors sind dagegen für die Umsetzung von JMX auf ein anderes Protokoll zuständig. Dadurch wird es ermöglicht, aus bestehenden Managementsystemen, z.B. basierend auf SNMP, auf JMX Agents und deren MBeans zuzugreifen und diese zu verwalten. Das Distributed Services Level und seine Komponenten sind jedoch in der gegenwärtig vorliegenden Spezifikation V1.1. leider noch nicht genauer spezifiziert, in Implementierungen wie MX4J finden sich aber bereits Funktionalitäten, die sich in der Version V1.5 finden werden.

Zusätzlich zu den drei genannten Ebenen der JMX Architektur sieht die JMX Spezifikation APIs vor, die als Schnittstelle zu anderen Managementstandards fungieren sollen. Im Gegensatz zu den Protocol Adaptors ermöglichen es diese sogenannten Additional Management Protocol APIs, Ressourcen eines anderen Managementstandards in ein JMX-basiertes Managementsystem einzubinden. Solche Additional Management Protocol APIs sind für die Protokolle SNMP und CIM/WBEM vorgesehen und teilweise bereits erhältlich. Die Spezifikation dieser APIs erfolgt außerhalb der JMX Spezifikation.

Das vorliegende Rahmenwerk sieht den Zugriff auf SNMP und ähnliche Dienste von den die Klasse `NeveAPI` implementierenden Managementanwendungen aus vor. Der Prototyp implementiert sie zunächst als lokale MBeans auf dem Agenten. Sollte ein Zugriff auf verteilte Managementanwendungen gewünscht sein, so kann dieser über bereits vorhandene oder zukünftig spezifizierte JMX-Schnittstellen geschehen.

4.2.4 Notifikationen

Die Definition der Attribute und Operationen eines MBean zum Zugriff von außen ist für ein umfangreiches Netzmanagement alleine nicht ausreichend. Es wird ein Mechanismus benötigt, der den MBeans ermöglicht, auf eigene Initiative Zustandsänderungen anzuzeigen. Auch im Falle der im Kapitel 2.1.4 betrachteten Ereignismeldungen ist ein solcher Mechanismus notwendig. Das JMX Notification Model ermöglicht es, Benachrichtigungen (Notifikationen) von einem MBean aus an andere Komponenten der JMX-Architektur zu schicken. Es basiert auf dem Java Event Model und besteht aus vier Komponenten: der Klasse `Notification`, dem Interface `NotificationBroadcaster`, dem Interface `NotificationListener` und dem Interface `NotificationFilter`. Die Nachrichten werden durch Objekte vom Typ `Notification` repräsentiert. Eine Unterscheidung zwischen verschiedenen Nachrichtentypen wird im Allgemeinen durch den Wert des Feldes `notification type` getroffen. Dieser Wert wird durch einen String angegeben, der aus mehreren durch Punkte getrennten Elementen besteht. Dadurch ist es möglich, eine Struktur von Nachrichtentypen zu definieren. So haben z.B. die Nachrichtentypen der JMX-Infrastruktur die Form `jmx.*`. Der Prototyp, wie auch das ihm zugrundeliegende Rahmenwerk, macht häufigen Gebrauch von Notifikationen, um dynamisch Zustandsänderungen der Daten und der grafischen Repräsentationen weiterzuleiten.

4.3 Implementierung des Prototypen

Auch der zweite Prototyp erhielt den Namen NEVE (NEtwork Visualisation Environment), trägt allerdings Versionsnummern höher als V 0.94. In diesem Kapitel werden die Besonderheiten der Implementierung des zweiten Prototypen beschrieben, der das in Abschnitt 3 entworfene Design implementiert und als dessen Tragfähigkeitsnachweis dienen soll.

Nicht zuletzt um eine Plattformunabhängigkeit zu erreichen, wurde der Prototyp in der Programmiersprache Java unter Verwendung von Java3D als grafische Engine sowie MX4J [MX4J], einer OpenSource Implementierung der Java Management Extensions (JMX, [sun 00]) ausgeführt. Die Java Management Extensions definieren eine auf Java basierende Architektur und Infrastruktur für das Anwendungs- und Netzmanagement. Sie sind dafür gedacht, die Entwicklung von Managementanwendungen für verteilte Systeme in allen Branchen zu ermöglichen [sun 99]. JMX wurde ausgewählt, da es eine moderne Managementumgebung für Java darstellt und alle im Design des Rahmenwerkes genannten Anforderungen wie z.B. die Unterstützung von Komponenten erfüllt. Es bot mit Diensten wie beispielsweise dem Relationservice (zur Verwaltung von Relationen) eine sehr gute Basis für das vorliegende Projekt und gängige Managementplattformen wie IBM Tivoli und HP Open View bieten Schnittstellen zu JMX an.

Die Gründe für die Wahl von Java3D als 3D-Engine und Beschreibungssprache wurden bereits im vorigen Kapitel ausgeführt. Ein weiterer Vorteil von Java3D ist das erst kürzlich erschienene Java3D(TM)Fly Through von Sun, welches eine Umgebung zur Navigation durch Java3D-Szenen bereitstellt, die bereits mit vielen nützlichen Merkmalen ausgestattet ist. Beispielsweise kann man zwischen verschiedenen Navigationsmodi wie Orbit (Drehen eines Objektes um seine Achse), Hover (Hovercraft) oder Fly (Flugzeug) wählen. Ferner stehen neben einer Texturdarstellung auch eine Drahtgitternetz- und einer Punktdarstellung zur Verfügung und man kann Wegmarkierungen setzen, an die nach Bedarf wieder zurückgekehrt werden kann. Das System ist außerdem in der Lage, Stereobildprojektionen zu erzeugen, was einer eventuellen späteren Visualisierung auf einer Holobench zugute kommt. Es ist von Sun als Demonstration und Programmiervorlage gedacht und bildet nach einigen kleinen Änderungen am Code sowie der Anbindung an die im Rahmenwerk spezifizierte Connectorklasse (siehe Kapitel 3.6.3) das Benutzer-Interface des Prototypen.

4.3.1 Management-Beans

Die im Design des Rahmenwerkes spezifizierten MIOs und PIOs lassen sich im Sinne von JMX als MBeans darstellen. Die Visualisierungsanwendungen als zentrale funktionale Teile des Prototypen können als Agent Services aufgefasst werden und wurden ebenfalls als MBeans implementiert. Es wird davon ausgegangen, dass die anzubindenden Managementanwendungen entweder ebenfalls als MBeans unter JMX vorliegen, oder ein Proxy-MBean[24] erstellt wird, dass die Anbindung an die Managementanwendung vornimmt. Alle verwendeten MBeans wurden gemäß der JMX-Spezifikation in eine Domänenstruktur eingeteilt. Sie befinden sich alle unterhalb der

[24] vgl. Proxyanwendung in den Abschnitten 3.4.2 und 3.4.3

Wurzel "neve".

Im Einzelnen gibt es die Domänen:

- `neve.base` - die zentralen Visualisierungsanwendungen (Anwendungsservice, Mappingservice, Visualisierungsservice) und der Layoutservice
- `neve.data.mib` - die gegenwärtig instantiierten Informationsobjekte
- `neve.data.pib` - die gegenwärtig instantiierten Präsentationsobjekte
- `neve.data.visualisierungsfunktionen` - die Funktionen zur Veränderung von MPOs zur Interaktion mit der Präsentationsschicht
- `neve.managementanwendungen` - enthält die beispielhaften Managementanwendungen, die zum Test des Prototypen erstellt wurden.

Die Domäne `neve.data.pib` enthält die in Kapitel 3.5.2 beschriebene Paketstruktur, wie sie beispielsweise in der Domäne `neve.data.pib.lrz.mail` zum Ausdruck kommt. Für den Prototypen wurde zunächst nur die Domäne neve.data.pib.standard implementiert. Die Domänenstruktur sowie deren Inhalt unterhalb von `neve.data.*` kann für neue Anwendungsfälle aber jederzeit erweitert werden. Beispielsweise können so durch Hinzufügen von MIOs und MPOs zu neve.data.mib und neve.data.pib weitere Ressourcen visualisiert oder durch neue Visualisierungsfunktionen in `neve.data.visualisierungsfunktionen` neue Sachverhalte und logische Zusammenhänge dargestellt werden.

Die Visualisierungsanwendungen in `neve.base` benötigen zusätzlich zum MBeanServer die JMX-Dienste RelationService und TimerService, diese wurden im Prototyp unterhalb der Domäne `agent` registriert. Ein weiteres auf dem Server notwendiges MBean ist der von MX4J zur Verfügung gestellte RMI-Adapter (Remote Method Invocation), der ein Protokoll zum entfernten Zugriff auf MBeans zur Verfügung stellt. Er wird vom Benutzer-Interface benötigt, um über eine Netzverbindung Notifikationen empfangen zu können und um auf die Funktionen des Visualisierungsservice zugreifen zu können. Der RMI-Adapter benötigt eine RMI-Registry als Namingservice, die in MX4J ebenfalls als MBean realisiert wurde. Ferner werden zwei HTTP-Adaptoren instantiiert, einer auf Port 9090 zur Benutzerinteraktion per HTML, der andere auf Port 9091. Letzterer basiert auf XML und soll späteren Erweiterungen des GUI die Möglichkeit bieten, Benutzerinteraktionen ganz in die dreidimensionale Visualisierung zu integrieren.

Die Paketstruktur der Implementierung sieht folgendermaßen aus:

- `neve.core.objects` - Die Basisklassen
- `neve.core.services` - Die Visualisierungsanwendungen
- `neve.core.api` - Unterstützendendes API für Managementanwendungen
- `neve.data.mib` - Managementinformationsobjekte, definieren Templates für Rohdaten
- `neve.data.pib.standard` - Managementpräsentationsobjekte
- `neve.data.pib.lrz` - Weitere Ebenen von Präsentationsobjekten aus dem Beispiel

in Kapitel 3.5.2

- `neve.data.functions` - Visualisierungsfunktionen
- `neve.server` - Basis-Agent, startbare Initialisierungsklasse
- `neve.client` - Benutzer-Interface

Die folgenden Kapitel beschreiben Besonderheiten der Implementierung dieser Teilpakete.

4.3.2 Basisklassen

Für die Implementierung der MBeans wurden einige Basisklassen eingeführt, die die Grundlage für die anderen Klassen bilden und die von diesen erweitert werden. Die Grundlage dafür liefert die Beobachtung, dass sich alle vom Prototypen verwendeten Objekte auf einige Grundformen zurückführen lassen. Im Einzelnen sind das MIOs, MPOs und Services, ausgeprägt in den Klassen `MIO`, `MPO` und `NeveService`.

Für die Realisierung dieser Klassen in einer JMX-Umgebung wurde das Modell dynamischer MBeans gewählt. Dynamische MBeans haben gegenüber Standard MBeans den Vorteil, dass ihre Attribute und Operationen bei entsprechender Programmierung des`DynamicMBeanInterface`, leichter vererbbar sind. Da das Objektmodell der Visualisierungsarchitektur Vererbung vorsieht, wurde für den Prototypen die Klasse `StandardNeveMBean` entworfen, die ein entsprechendes DynamicMBean implementiert und für alle vom Visualisierungssystem erzeugten MBeans vererbbare Methoden zur Verwaltung der MBean Attribute und Operationen bereitstellt. Sobald mit Hilfe der Funktion `setAttribute (Attribute attribute)` ein Attribut eines MBeans gesetzt wurde, wird `MBeanAttributeChanged(String Attribute)` aufgerufen. Diese Methode wird später von Unterklassen überschrieben. Zusätzlich ermöglicht das `StandardNeveMBean` allen von ihm abgeleiteten MBeans das einfache Senden von Notifikationen. Zu diesem Zweck erweitert es die in der JMX-Spezifikation definierte Klasse `NotificationBroadcasterSupport`. Zum schnellen Auffinden des MBean-Servers stellt es außerdem die Methode `getServer()` zur Verfügung.

Die Basisklassen `MIO` und `NeveService` erweitern `StandardNeveMBean` direkt, während `MPO` die Klasse `NeveService` erweitert. Der Grund dafür wird klar, wenn man sich ansieht, welche Operationen diese drei Basisklassen im Einzelnen bieten.

`MIO` stellt für die in Kapitel 3.4.1 spezifizierten Attribute für Managementinformationsobjekte Getter und Setter zur Verfügung. Ferner überschreibt die Operation `MBeanAttributeChanged(String Attribute)` die gleichgenannte Methode des `StandardNeveMBean` und sendet eine Notifikation des Typs `AttributeChangeNotification` an die beim MIO registrierten MPOs, sobald ein Attribut des MIOs verändert wurde.

Die Klasse `MPO` definiert die in Kapitel 3.5.1 spezifizierten Attribute für Managementpräsentationsobjekte und deren Getter und Setter. Eine Operation `start()` wird

vom Mappingservice sofort nach dem Erzeugen des Objekts aufgerufen. Sie meldet sich bei dem ihm zugehörigen Managementinformationsobjekt und beim Relationservice als `NotificationListener` an, was dazu führt, dass es zukünftig über Änderungen der Attribute des Managementinformationsobjektes und über Änderungen an der Relationsmenge informiert werden wird. Trifft dann eine Notifikation ein, so wird die Operation `handleNotification(Notification notification, Object handback)` des MPOs aufgerufen. Diese nimmt eine Vorsortierung vor und steuert je nach Notifikationstyp die Operationen `MIOAttributeChange(String attribut)`, `MPOAttributeChange(String attribut)` oder `RelationChange()` an. `MPO` wird von `NeveService` abgeleitet, da einige Präsentationsobjekte auch Visualisierungsfunktionen übernehmen können und damit auch über den Server auf andere MBeans zugreifen müssen. Wird beispielsweise ein MPO bearbeitet, welches die Daten des ihm zugehörigen MIOs mit in die Visualisierung einbeziehen will, so muss es über die Möglichkeit verfügen, die Attribute des MIOs auszulesen. Um ein anderes Beispiel zu nennen, könnte ein MPO auch beim Eintritt bestimmter Bedingungen weitere MPOs erzeugen und auch wieder löschen, und damit auch sehr komplexe grafische Repräsentationen für die Attribute einzelner MIOs realisieren.

Die Grundklasse `NeveService` schließlich legt einige statische Variablen fest, die interne Namens-Konventionen darstellen. So definiert sie Namen für die Relationen, mit deren Hilfe der Relationservice diese unterscheiden kann, außerdem legt sie eindeutige Namen für Mappingservice, Relationservice und Timerservice fest. Darüberhinaus stellt die Klasse einige nützliche Standardoperationen auf dem Server bereit, die von den Visualisierungsanwendungen verwendet werden. Darunter fallen beispielsweise das Erzeugen und Löschen von MBeans und Relationen sowie der Zugriff auf Attribute von MBeans[25].

4.3.3 Visualisierungsanwendungen

Die im Rahmenwerk beschriebenen Visualisierungsanwendungen wurden als MBeans im Sinne von JMX Agent Services ausgeführt. Sie erweitern jeweils die Basisklasse `NeveService`.

Der implementierte Anwendungsservice speichert die Namen der von ihm erzeugten MBeans intern und stellt ein Mapping auf eine ID vom Datentyp Integer zur Verfügung. Diese IDs werden bei der Erzeugung der MBeans in der Methode `createMIO(String name,String domain)` linear vergeben. Zum Erzeugen eines neuen MIOs wird im Paket `neve.data.mib` die Klasse mit dem angegebenen Namen gewählt und als MBean registriert. Dann werden die Attribute `Source`, `Domain`, `Type` und `MIOID` mit Werten belegt. Das MBean erhält auf dem Server den eindeutigen Namen

```
String JMXObjName="neve.data.mib:name="+typ+",mapping=
       neve.data.pib."+domain+",type=MO,MID="+mioid;
```

Außerdem implementiert der Anwendungsservice die weiteren von seinem Interface geforderten

[25] detailierte Beschreibungen aller Operationen und deren Parameter finden sich für alle Klassen in der Javadoc-Dokumentation des Prototypen.

Funktionen zum Löschen von MIOs und zum Erzeugen und Löschen von Relationen. Um dies zu erreichen, wird der Relationservice angesprochen.

Der Mappingservice überwacht mit Hilfe von Server-Notifikationen die Menge der erzeugten MIOs. Sobald ein neues MIO gefunden wird, erzeugt er ein dem Eintrag im Attribut `Domain` des MIOs entsprechendes MPO. Dies bedeutet, dass ein neues MBean als die von

```
String Class="neve.data.pib."+domain+"."+classname;
```

bestimmte Klasse registriert wird. Als Klassenname "classname" dient dabei der Typ des MIO. MPOs erhalten bei der Registrierung auf dem Server den eindeutigen Namen

```
String JMXObjName=neve.data.pib."+domain+":name="+classname+
       ",type=PO, PID="+poid;
```

Wird ein MIO auf dem Server deregistriert, so erkennt dies der Mapper via Notifikation ebenfalls und deregistriert das entsprechende MPO. Um mit der Methode `getMapping(String mioname)` anderen Anwendungen der Präsentationsschicht eine Referenz auf die POs zur Verfügung stellen zu können, wird für jedes MPO eine lineare Id `poid` vergeben und intern auf den MBean-Namen abgebildet und gespeichert. Kommt eine Anfrage nach einem MIO-Namen, so liefert diese Funktion den Namen des MPO zurück. Sie wird beispielsweise vom Layoutservice benötigt.

Der im Prototypen verwendete Layoutservice hält sich an die im Design vorgegebene Form. Die Methoden `doLayout()` und `buildConnectionVectors()` sind die zentralen Methoden, die das Layout durchführen. Letztere wird von Server- und Relationservice Notifikationen ausgelöst und bezieht sich dabei auf alle auf dem Server vorhandenen MIOs. Sie unterteilt diese in Knoten (alle MIOs, die keine Adjazenzrelation darstellen) und Kanten (alle MIOs, die eine Adjazenzrelation darstellen) ein. Das Ergebnis wird in den Vektoren `nodes` und `edges` gespeichert. Sobald ein Layoutschritt ausgelöst wird, werden anhand der beiden vorher erzeugten Vektoren und einem Spring-Embedding-Algorithmus alle MPOs im Raum an neue Positionen verschoben. Dies erfolgt entsprechend der Knoten und Kanten, also entsprechend der Lage der MPOs und der Verbindungen zwischen den korrespondierenden MIOs und berücksichtigt eine Gewichtung der Kanten, die im Attribut `Weight` der Klasse `PDU` gespeichert ist. Dieses kann für jede Verbindung einzeln gesetzt werden. Der Layoutservice führt dabei Zugriffe auf den Mappingservice, den Relationservice und auf die MIOs der Verbindungen durch.

Für die Datenhaltung im Visualisierungsservice wurde die Klasse `DefaultTreeModel` verwendet, so wie sie von Java Swing bereitgestellt wird. Dies erleichtert die Verwaltung des aufgebauten Szenegraphen und erlaubt dessen Darstellung in einem Fenster zum Debugging oder für evtl. spätere Benutzeroberflächen des Servers.

Das TreeModel hält dabei an seinen Knoten die eindeutigen Namen der MBeans, die MPOs repräsentieren. Sie sind entsprechend der Enthaltenseinsrelation im Managementinformationsraum angeordnet. Erhält der Service eine Notifikation über die Änderung von MPOs oder die Änderung der Relation, so wird der Inhalt des TreeModel entsprechend dem Inhalt der Notifikation abgeändert. `DefaultTreeModel` ist so konzipiert, dass

es seinerseits spezielle Methoden aufruft, sobald es verändert wird. Hierbei handelt es sich unter anderem um die Methoden `treeNodesChanged(TreeModelEvent e)` und `treeNodesInserted(TreeModelEvent e)`. Der Prototyp nutzt dies aus, um daraufhin mit Hilfe der in Kapitel 3.6.1 spezifizierten Operationen `addCluster(int cluster)`, `updateCluster(int cluster)` und `removeCluster(int cluster)` die Clients über die Änderungen zu verständigen.

4.3.4 Managementanwendungen

Im Paket `neve.core.api` stellt der Prototyp die Klasse `NeveAPI` bereit, mit deren Hilfe Managementanwendungen die Visualisierungsdienste nutzen können. Das Paket implementiert alle im Design der API spezifizierten Funktionen. Die Verbindung zum Anwendungsservice wird über die Funktionen des MBean-Servers hergestellt. Für die Managementanwendungen selbst ist das Paket `neve.data.applications` vorgesehen. Für Testzwecke wurden verschiedene Managementanwendungen erstellt, darunter der `ExampleNetCreator`, welcher automatisch Datenbasen für beliebige Netzwerkgrößen erstellt. Es kann eingestellt werden, aus wievielen Switches das Beispielnetz bestehen soll und wieviele Endsysteme an jedem Switch angschlossen sind. Alle Switches werden mit einem Router verknüpft. Router, Switches und Endsysteme erhalten die notwendigen Interfaces und alles wird mit TwistedPair-Kabeln über die entsprechenden Relationen verknüpft. Mit dieser Anwendung können schnell Visualisierungen erstellt werden. Sie liefert keine realen Daten, stellt aber durch die Skalierbarkeit der erzeugten Beispieldaten einen zweckmäßigen Test dar.

Das Kapitel 4.4 enthält die Beschreibungen weiterer im Prototypen implementierter Anwendungen.

4.3.5 Anwendungsfallorientierte Objekte und Funktionen

Um ein Anpassen der Visualisierung an neue Anwendungsfälle möglich zu machen, sieht das Design zur Gestaltung der Visualisierung Objekte und Funktionen vor. Sie werden in der Paketstruktur als eigenes Paket mit dem Namen `neve.data` geführt. Hier kann für jede neue Anwendung die Definition der Informations- und Präsentationsobjekte aber auch die der Visualisierungsfunktionen erweitert werden.

Alle MIOs befinden sich dabei im Paket `neve.data.mib`. Eine weitere Unterteilung in Unterpakete ist nicht vorgesehen, könnte aber aus Gründen der Übersichtlichkeit eventuell später durchgeführt werden, falls die Anzahl der Primitive zu groß werden sollte. Alle MIOs sind von der Basisklasse `MIO` abgeleitet und tragen somit die im Design definierten Attribute. Als grundlegende MIO-Typen wurden `System` und `Architecture` berücksichtigt. Ferner wurden die Klassen `Internet`, `PDU` und `ProtocolInstance` realisiert und mit verschiedenen Funktionalitäten versehen.

So implementiert `PDU` alle Operationen des JMX-Interfaces `Relation`, um zu gewährleisten,

dass alle abgeleiteten Klassen vom MBean-Server als Relation-MBeans geführt werden können. Dies ermöglicht Konstruktionen wie beispielsweise eine Klasse `TwistedPair`, die als `PDU` eine Adjazenzrelation darstellt und zwei andere MBeans (beispielsweise `Host`) miteinander verbindet. Alle Netzressourcen, die andere Ressourcen miteinander verbinden (d.h. Adjazenzrelationen darstellen), müssen von `PDU` abgeleitet werden, um bei MBean-Server und Relationservice als Relation-MBean verwaltet werden zu können.

Die Klasse `Internet` stellt das Attribut `Layer` zur Verfügung, das die Schichtzugehörigkeit einer Ressource bezeichnet. Dieses wird später vom dazugehörigen Präsentationsobjekt `Internet` benötigt und dient dazu, Verkehrsflüsse und Protokollinstanzen entsprechend ihrer Schichtzugehörigkeit darstellen zu können.

Die Managementpräsentationsobjekte befinden sich in den Paketen unterhalb von `neve.data.pib`. Wie im Design vorgesehen, wird die Paketstruktur von hier aus weiter verzweigt. Für den Prototypen wurden im Paket `neve.data.pib.standard` mehrere MPOs für Netzkomponenten geschaffen. Zur Vereinfachung der grafischen Ausgestaltung wurde für einige Objekte dieses Pakets ein Loader für 3D-Modelle, die mit dem Shareware Modeler Milkshape3D erzeugt wurden, verwendet. Neben MPOs, die nur das Attribut `Shape` für den jeweiligen Darstellungstyp definieren, wurden die MPOs `Internet` und `PDU` mit weiteren Funktionalitäten ausgestattet.

Die Klasse `PDU` überschreibt die Methoden `POAttributeChange(attribute)` und `RelationChange()` der Basisklasse `MPO`. Von dort wird `RelationChange()` ausgelöst, sobald sich im Informationsraum eine Relation ändert. Hier werden die Quell- und Endpunkte der Verbindung festgestellt, die `PDU` im konkreten Fall darstellt und die folgende global definierte Datenstruktur mit dem Ergebnis belegt:

```
ObjectName QuellMIO=null;
ObjectName QuellMPO=null;
Vector ZielMPO=new Vector();
Vector ZielMIO=new Vector();
```

QuellMIO und QuellPO fassen nur jeweils den Namen eines MBeans, ZielMIO und ZielMPO einen Vector aus MBean-Namen. Der Grund dafür ist die Definition der Adjazenzrelation im Kapitel 3.4.

Die überschriebene Methode `POAttributeChange(attribute)` wird indirekt vom Layoutservice für alle Verbindungen regelmäßig getriggert, sobald die neuen Koordinaten aller Knoten-MPOs berechnet sind. Dies geschieht, indem ein beliebiges Attribut der Kante nach dem Layout der Knoten mit seinem altem Wert überschrieben wird. Sobald die Abarbeitung der Methode begonnen hat, beginnt die Feststellung der absoluten Koordinaten, bei denen die von "PDU" darzustellende Verbindungslinie beginnt und endet. Dies ist notwendig, da das MPO, von dem die Verbindung ausgeht, später vom Visualisierungsservice im Enthaltenseinsbaum unterhalb eines anderen Knoten eingehängt wird, wenn das darzustellende Objekt in diesem enthalten ist. Dies resultiert darin, dass die Koordinaten des Knotens von Java3D relativ zu den Koordinaten des darüberliegenden Knotens betrachtet werden.

Zur Feststellung der absoluten Koordinaten wird zunächst das MIO des Objektes betrachtet, von dem die Verbindung ausgeht. Dann erfolgt eine Anfrage an den Relationservice, ob das MIO an einer Enthaltenseinsrelation[26] beteiligt ist. Das selbe geschieht rekursiv mit den so festgestellten Vaterknoten des Objektes, bis schließlich die Enthaltenseinsrelation nicht mehr anwendbar ist. Bei jedem Schritt werden dabei die Koordinaten des dem gerade betrachteten Knoten-MIO zugehörigen MPOs aufaddiert, was einer Vektoraddition entspricht. Das Ergebnis ist schließlich die absolute Koordinate des Startpunktes. Das Verfahren stellt sich im Programmcode wie folgt dar:

```
float[] coords=getCoords(QuellMIO); // hole MBean-Koordinaten
                                    // des PO, das zu dem MIO
                                    // gehört
x=coords[0];
y=coords[1];
z=coords[2];

// Entlang der Enthaltenseinsrelation hochpropagieren,
// für absolute koordinaten
ObjectName parent=QuellMIO; // beginne beim Objekt, das an der
                            // Adjazenzrelation beteiligt ist
ObjectName oldparent=null;
while (parent!=null){
     oldparent=parent;
     parent=getEnthaltenseinsParent(parent); // Suche nach evtl.
                                             // Eltern in der
                                             // Enthaltenseins-
                                             // relation
     if (parent!=null){
          coords=getCoords(parent);
          x=x+coords[0];
          y=y+coords[1];
          z=z+coords[2];
     }
}
```

`getEnthaltenseinsParent(ObjectName name)` und `getCoords(ObjectName name)` dienen dabei dem Zugriff auf den MBeanServer, um die entsprechenden Daten über das als Parameter übergebene MBean zu erhalten. Das rekursive Verfahren wird dann noch einmal auf den Zielpunkt angewendet. Sind Start- und Zielkoordinaten bekannt, kann ein Java3D Linienobjekt definiert werden, welches vom Startpunkt zu den Zielpunkten Linien zieht und damit das Attribut `Shape` des `Connection`-MBeans gesetzt werden.

Als weiteres Paket wurde `neve.data.functions` definiert. Es kann ebenfalls von Anwen-

[26] in Kapitel 3.4 definiert

dungsprogrammierern erweitert werden und enthält ausschließlich Visualisierungsfunktionen, die auf der Ebene des Präsentationsraumes arbeiten und abhängig von Benutzerinteraktionen und Informationsraum die Attribute der MPOs und damit deren grafische Darstellung verändern.

4.3.6 Das Benutzer-Interface

Das Benutzer-Interface besteht aus zwei Teilen: dem Visualisierungsteil und dem Interaktionsteil.

Der Visualisierungsteil basiert auf dem Java3D FlyThrough Beispiel und wurde für den Prototypen angepasst und um die von der Spezifikation des Rahmenwerkes definierte Klasse `Connector` erweitert, um die Darstellungsdaten dynamisch vom Server einlesen zu können. Die grafische Darstellung und Navigation erfolgt selbst bei großen Netzen sehr flüssig. Anwender können nach der Herstellung einer Verbindung durch das Netz navigieren. Hierfür stehen mehrere Modi ("Fly","Hover","Drive","Orbit") zur Verfügung, die das Verhalten der Maussteuerung regeln. Der Modus "Orbit" ermöglicht eine Drehung des dargestellten Ausschnittes um seine Achse und ist ein praktisches Werkzeug, um die Objekte von allen Seiten zu betrachten. Die anderen Modi ermöglichen mehrere verschiedene Arten der Fortbewegung durch die Darstellung.

Die allgemeine Darstellung des Netzes kann verändert werden, indem Texturen und Lichter abgeschaltet werden oder die Abbildungstiefe, d.h. die Sichtweite des Betrachters, verändert wird. Ferner gibt es die Möglichkeit, die aktuelle Visualisierung abzuspeichern und später wieder einzuladen. Sofern gewünscht, kann auch eine Kollisionserkennung aktiviert werden. Zum Wiederauffinden bestimmter Punkte innerhalb des Darstellungsraumes können Wegepunkte gesetzt und später wieder angesteuert werden. Java3D FlyThrough unterstützt Stereobild-Projektion und eignet sich daher gut für eine Visualisierung auf VR-Medien.

Der Interaktionsteil des Benutzer-Interfaces besteht aus einem HTML-Adapter aus der Sun JMX Reference Implementation, der es ermöglicht, über einen Webbrowser auf die Attribute aller MBeans lesend und schreibend zuzugreifen und Operationen zu starten. Damit können die verschiedenen, auf dem Server vorhandenen Funktionen und Anwendungen gesteuert werden. Der HTML-Adapter bindet sich an Port 9090 und kann von außen über das Netz angesprochen werden. Es ist geplant, die momentan über den Webbrowser angebotene Funktionalität in einer späteren Version des Prototypen vollständig in die dreidimensionale Darstellung zu integrieren. Hierfür wurde ein HTTP-Adapter auf Port 9091 vorgesehen, der über eine XML-Schnittstelle verfügt und in der Lage ist, automatische Anfragen an alle Objekte, die auch über den Webbrowser erreichbar sind, zu bearbeiten. Abbildung 17 zeigt das vollständige Benutzer-Interface, links ist der Visualisierungsteil, rechts der Interaktionsteil zu sehen.

Für die Herstellung der Verbindung zwischen der Klasse `Connector` im Visualisierungsteil des Benutzer-Interfaces und dem Visualisierungsservice im Server instantiiert die Klasse `Connector` den von MX4J mitgelieferten `JRMPConnector`. Die Verbindung zum Visualisierungsservice erfolgt von da aus über RMI, was "Remote Method Invocation" bedeutet und das Java-Pendant zu RPC (Remote Procedure Call) darstellt. Dieser von MX4J zur Verfügung

gestellte Mechanismus unterstützt auch den Empfang der vom Visualisierungsservice ausgesendeten Notifikationen. Wird eine solche empfangen, wird entsprechend der Designspezifikation die Methode `getCluster(int id)` aufgerufen, um einen aktuellen Teilbaum des Szenegraphen zu empfangen. Das Benutzer-Interface hält einen internen Speicher, in dem sich alle Teilbäume mit ihren IDs befinden. Sobald sich darin etwas ändert, wird das von Java3D darzustellende Universum erneuert und es ergibt sich eine aktuelle Darstellung der Daten im Server.

4.4 Realisierung der Szenarien

Zur Überprüfung der Funktionsfähigkeit und zur Feststellung der Leistungsfähigkeit des Prototypen wurden mehrere Tests auf einem Intel Celeron Prozessor mit 1,2 GHz, 256 MB Hauptspeicher und dem Betriebssystem Windows XP durchgeführt. Dabei wurden Server und Client beide auf demselben Rechner ausgeführt. Bei weiteren Versuchen konnte außerdem die Lauffähigkeit auf Unixsystemen gezeigt werden.

Die folgenden Abschnitte geben einen Überblick über die durchgeführten Tests und deren Ergebnisse.

4.4.1 Darstellung der Netztopologie

Zum Test der Topologiedarstellung wurden mehrere unterschiedlich große Netze über Beispielanwendungen in den Prototypen eingegeben. Abbildung 14 zeigt die Topologie eines Netzes mit einem Router, drei Switches und fünf Endsystemen. Die Endsysteme sind als durchsichtige Quader dargestellt, innerhalb der Endsysteme sieht man die Netzkarten als kleine Quader mit einem Pfeil. In der Mitte des Bildes kann man den als Zylinder dargestellten Router erkennen, an den die Switches angeschlossen sind. Diese sind als flache Quader mit vier Pfeilen repräsentiert. Die Linien stellen Twisted-Pair Kabel dar, d.h. die physikalische Ebene der Kommunikationsarchitektur. Sie verbinden jeweils die Interfaces der einzelnen Komponenten.

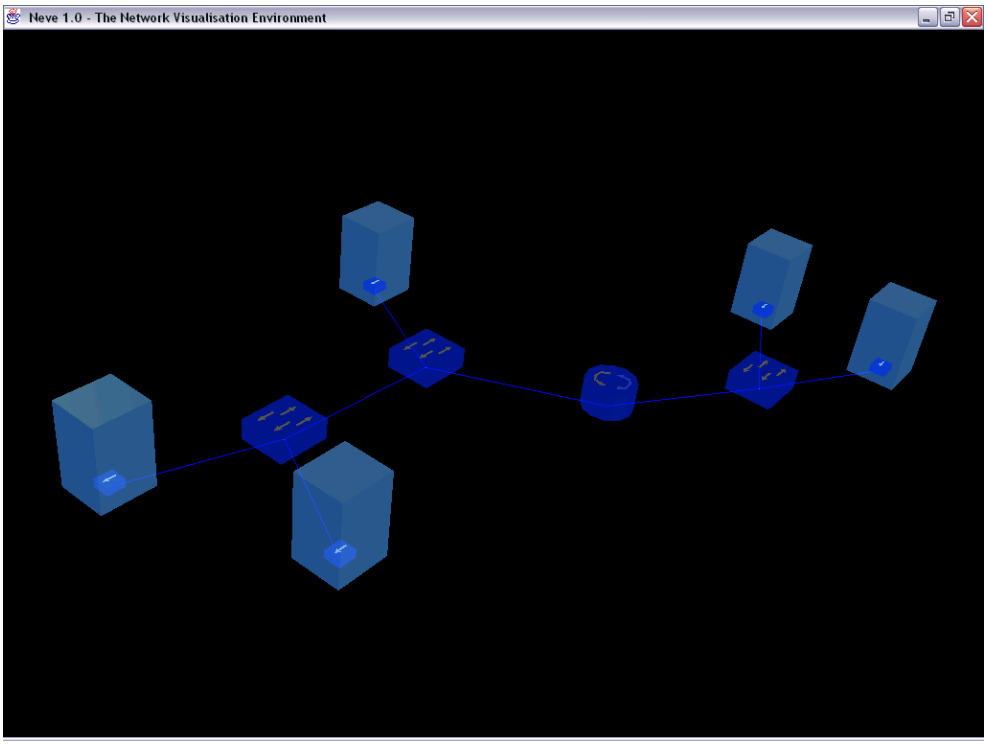


Abbildung 14: Visualisierung der Netztopologie

Es zeigte sich, dass es bei kleinen Netzen wie diesem möglich ist, den Layoutservice zeitgesteuert zwei- bis dreimal pro Sekunde automatisch arbeiten zu lassen, ohne dass Probleme mit der Geschwindigkeit auftreten. Dies bringt aber nur wenig Vorteile. Ist das grobe Layout einmal durchgeführt, resultieren weitere Layoutschritte in einer unruhigen Darstellung, bei der alle Netzknoten ihre Position nur um wenige Pixel verändern. Dies führt dazu, dass die Darstellung dem Betrachter unangenehm erscheint. Außerdem benötigt diese Art des Layouts einen großen Teil der zur Verfügung stehenden Rechenzeit. Es bremst damit die Ausführung wichtiger Programmteile, was bei der Berechnung größerer Netze zu einer Einschränkung der Prozessorkapazität führt.

Deshalb wurde der Layoutservice bei den folgenden Tests direkt nach dem Erstellen eines neuen Netzes mit einigen hundert Iterationen aufgerufen, um ein ausgeglichenes Netz zu schaffen, danach aber nur noch nach dem Hinzufügen neuer Knoten. Mit dieser Konfiguration konnte eine zufriedenstellende Visualisierung auch großer Netze erreicht werden. In Abbildung 15 ist die Topologiedarstellung eines von der Beispielanwendung "ExampleNetCreator" erzeugten, größeren Netzes mit 160 Komponenten abgebildet.

Bei Versuchen mit noch größeren Netzen zeigte sich, dass das System auch bei großen Datenmengen mit bis zu 1600 Netzkomponenten noch zufriendenstellend funktioniert. So konnte ein Router mit 100 Ports, 100 Switches mit insgesamt 400 Switchports und 300 Endsystemen mit 300 Netzkarten sowie 400 TwistedPair-Anschlusskabeln dargestellt werden. Allerdings bremst die Erzeugung der MPOs und das Layout in einem solchen Fall den Initialisierungsvorgang des Servers. Werden große Netze auf einmal eingelesen, so kann dieser Prozess mehrere Minuten dauern. Nach der Erzeugung der Netze und dem Layout nimmt die Prozessorlast jedoch stark ab und einzelne Änderungen an den Daten werden sofort angezeigt. Findet keine Neuerzeugung von Objekten und kein Layout statt, so benötigt der Server kaum Prozessorzeit und der Benutzer kann auch mit großen Netzen und deren Daten interagieren. Die erstmalige Datenübertragung des Netzes vom Server zum Benutzerinterface dauerte in diesem Versuch ca. 25 Sekunden.

4.4.2 Visualisierung und Konfiguration von VLANs

Zum Test der Darstellung und Konfiguration von VLANs wurde ein kleines Netz mit neun Endsystemen, drei Switches und einem Router verwendet.

Für die notwendige Funktionalität wurden folgende Klassen realisiert:

- `SwitchInterface` im Paket `neve.data.mib`
- `SwitchInterface` im Paket `neve.data.pib`
- `VLANHighlighter` im Paket `neve.data.functions`
- `ExampleNetCreator` im Paket `neve.data.applications`

Das MIO `SwitchInterface` ist von `ProtocolInstance` abgeleitet und stellt ein Attribut `VLAN` zur Verfügung, welches die VLAN-Zugehörigkeit des entsprechenden Switchports beschreibt und später von der Visualisierungsfunktion `VLAN-Highlighter` ausge-

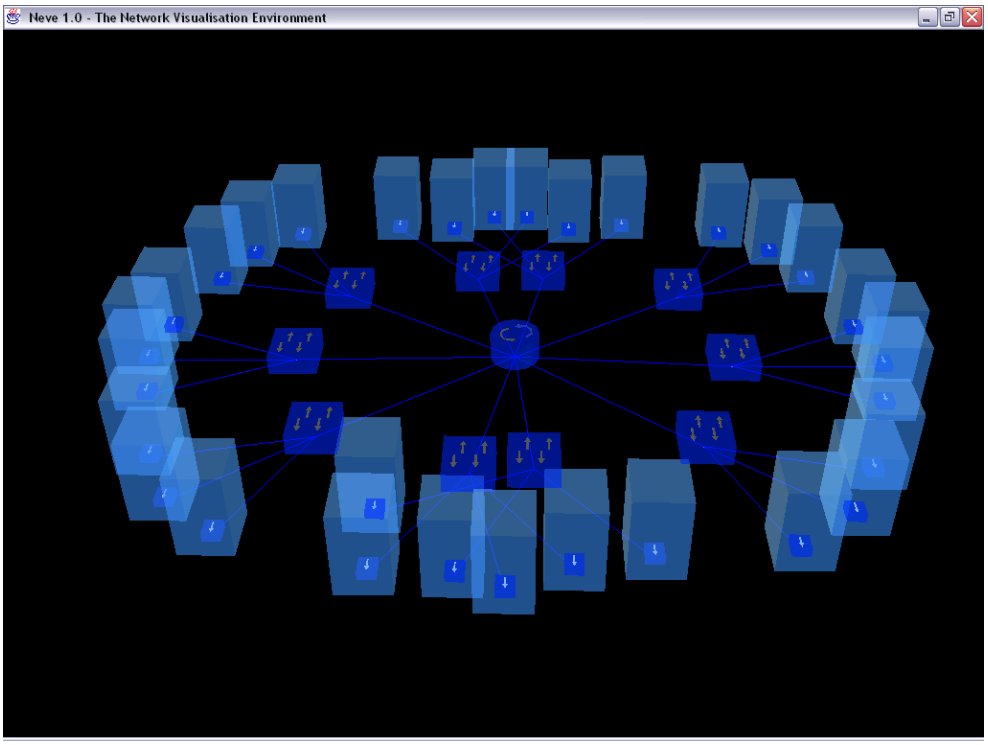


Abbildung 15: Topologiedarstellung eines großen Netzes mit 160 Komponenten

wertet wird. Das MPO `SwitchInterface` ändert die Farbe seiner grafischen Darstellung, je nach seiner Zugehörigkeit zu einem VLAN. Es überschreibt dabei die Methode `MIOAttributeChange(String attribut)`, um eine Änderung des Attributes `VLAN` an dem ihm zugehörigen MIO zu überwachen. Tritt eine solche ein, so wird die Definition des Attributes `Shape` des MPOs `SwitchInterface` so geändert, dass das Interface in einer Farbe entsprechend der VLAN-Zugehörigkeit dargestellt wird. Diese wird mit einem Aufruf der Methode

```
getAttribute(MIO,"VLAN");
```

am MBean Server festgestellt. Danach wird mit Hilfe von

```
super.MIOAttributeChange(attribut);
```

sichergestellt, dass die gleichgenannte, überschriebene Methode des MPOs `Internet` weiterhin abgearbeitet wird.

In den anschließenden Testläufen erfolgte bei einer Änderung der VLAN-Zugehörigkeit der Switchport-MIOs sofort eine entsprechende Änderung der Farbe. Ferner erhält die steuernde Managementanwendung eine Notifikation an ihre Methode `MIOAttributeChange(int id,String attribute)`. Das Beispiel `ExampleVLANCreator` beispielsweise gibt, sobald die VLAN-Zugehörigkeit eines Switchports geändert wurde, eine Meldung aus, welche Komponente betroffen ist und wie der neue Wert lautet. In der Praxis könnte von dort aus

über Managementmechanismen auch die tatsächliche Konfiguration am Switch vorgenommen werden.

Abbildung 16 zeigt auf der linken Seite die Draufsicht auf ein VLAN, die rechte Seite zeigt das MIO eines Switchinterfaces, an dem auch die VLAN-Zugehörigkeit eingestellt werden kann.

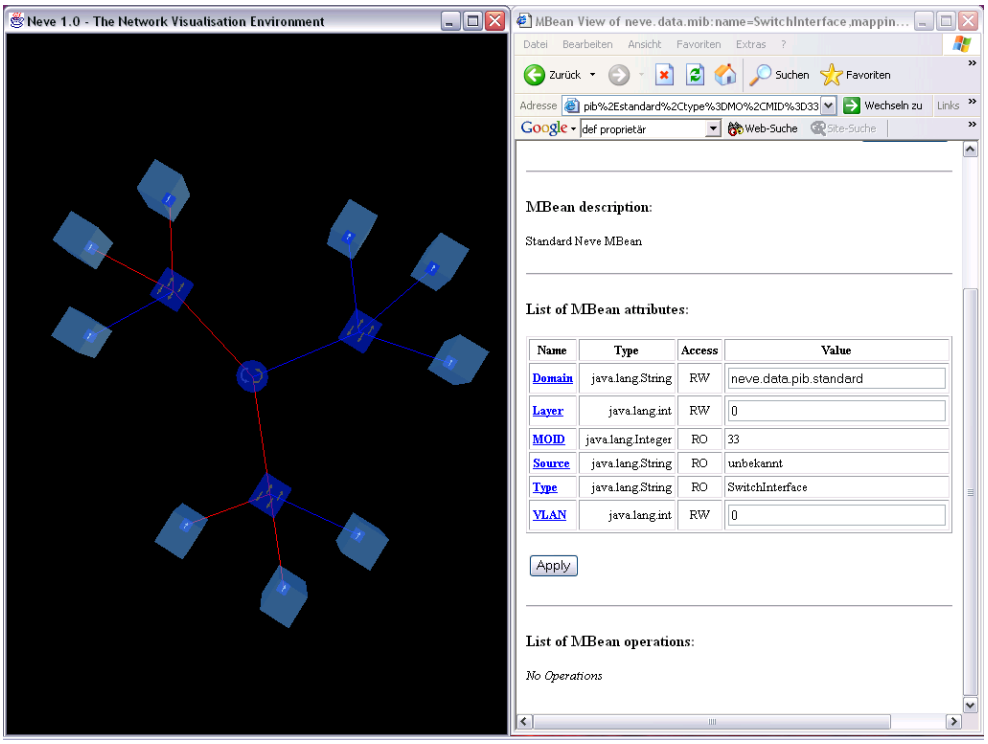


Abbildung 16: VLAN-Konfiguration eines Switches

Zur Markierung der konfigurierten VLANs wurde die Klasse `VLANHighlighter` verwendet. Sie hat die Aufgabe, bestimmte VLANs farbig zu markieren, um VLAN-spezifische Sichten zur Verfügung zu stellen. und verfügt über die Operationen `start` und `stop` mit denen er aktiviert bzw. deaktiviert werden kann. Ist er gestartet, so meldet er beim Timerservice ein im Attribut `Timer` einstellbares Intervall für Notifikationen an. Trifft eine solche ein, wird die Methode `highlight()` ausgeführt, die vom Server die Namen aller MIOs vom Typ `SwitchInterface` anfordert. Für jedes wird nun getestet, ob die im Attribut `VLAN` gespeicherte VLAN-Nummer mit der im Highlighter eingestellten Nummer übereinstimmt. Stimmen die Werte überein, so wird das Attribut `Highlight` der MPOs aller vom entsprechenden SwitchInterface-MIO ausgehenden Verbindungen auf den Wert `true` geändert. Dies führt dazu, dass die Verbindungen ab sofort in einer anderen Farbe dargestellt werden, da das MPO eine Notifikation über die Änderung erhält und seine Operation MIOAttributeChange(String attrib) aufgerufen wird. Das MPO der Verbindung ändert dann seine Farbe. Stimmen die Werte nicht überein, so erhält das Attribut `Highlight` den Wert `false`, was die Verbindungen in anderen VLANs in einer unauffälligen Farbe erscheinen läßt. Es gibt auch die Möglichkeit, das Highlighting direkt, ohne regelmäßigen zeitgesteuerten Aufruf zu starten. Der VLAN-Highlighter

kann wie alle anderen MBeans vom Benutzer über das HTTP-Interface angesteuert werden.

Abbildung 17 zeigt das Beispielnetz schräg von oben, beispielhaft die Markierungen eines VLANs sowie das Benutzer-Interface des VLANHighlighters. VLAN-Trunks und -Protokolle sind in der momentanen Version des Prototypen nicht implementiert.

In den Tests wurden alle in Frage kommenden Verbindungen markiert. Eine Änderung des zu markierenden VLANs führte im Timermodus des VLAN-Highlighters nach Ablauf des eingestellten Intervalls zu einer Änderung der Darstellung. Auch eine Markierung ausgewählter VLANs auf Abruf (durch den Benutzer direkt ausgelöst, also ohne Timer) ist problemlos möglich und führt zu dem gewünschten Ergebnis. Wie auch beim Layoutservice, gilt, dass eine auf Abruf erfolgende VLAN-Markierung sinnvoller erscheint, als die Markierung per Timer, da auch der VLAN-Highlighter alle Knoten betrachtet und damit zu einer merklichen Prozessorlast führen kann. Diese fällt aber bei einem Highlighting auf Abruf nicht ins Gewicht, da sie in diesem Fall nur kurze Zeit auftritt.

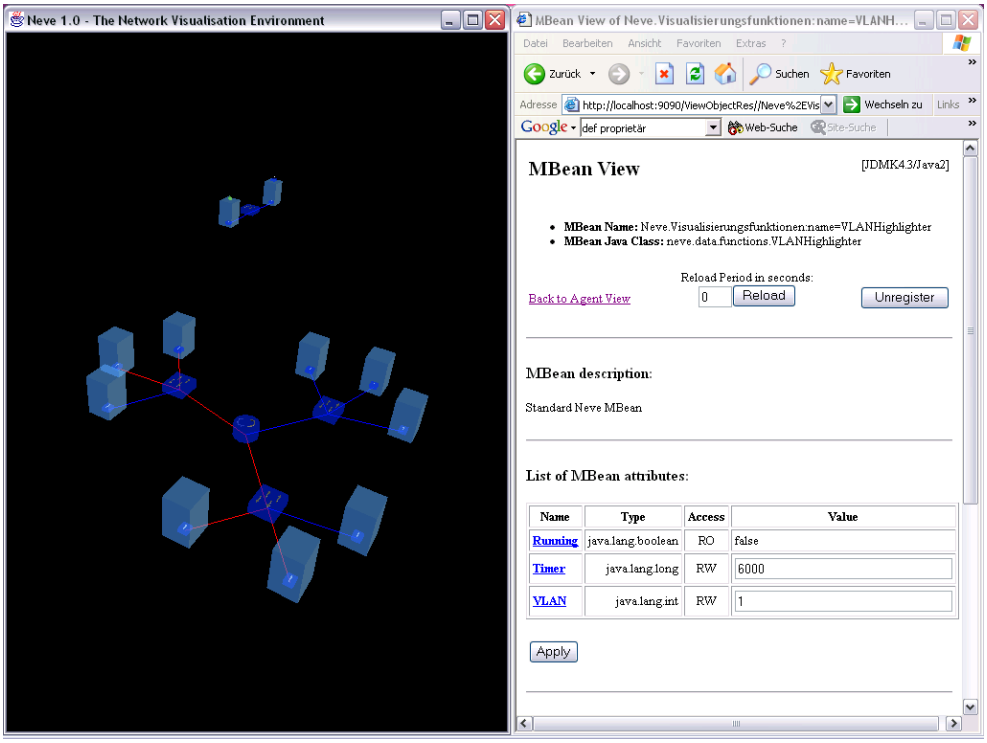


Abbildung 17: Markieren von VLANs

4.4.3 Darstellung von Verkehrsflüssen

Für den Anwendungsfall der Visualisierung von Verkehrsflüssen und zur Abschätzung der Leistungsfähigkeit von dynamischen Visualisierungen wurden anhand eines HTTP-Verkehrsflusses auf Schicht sieben einige Versuche durchgeführt. Hierzu wurde beispielhaft die Konfigurati-

on zweier Endsysteme an einem Switch simuliert. Auf dem einen Endsystem befindet sich ein Webserver, auf dem anderen ein Webbrowser. Zwischen beiden kann nun ein Verkehrsfluss stattfinden. Abbildung 18 zeigt zwei so konfigurierte Endsysteme und einen HTTP-Verkehrsfluss als Teil eines grösseren Netzes.

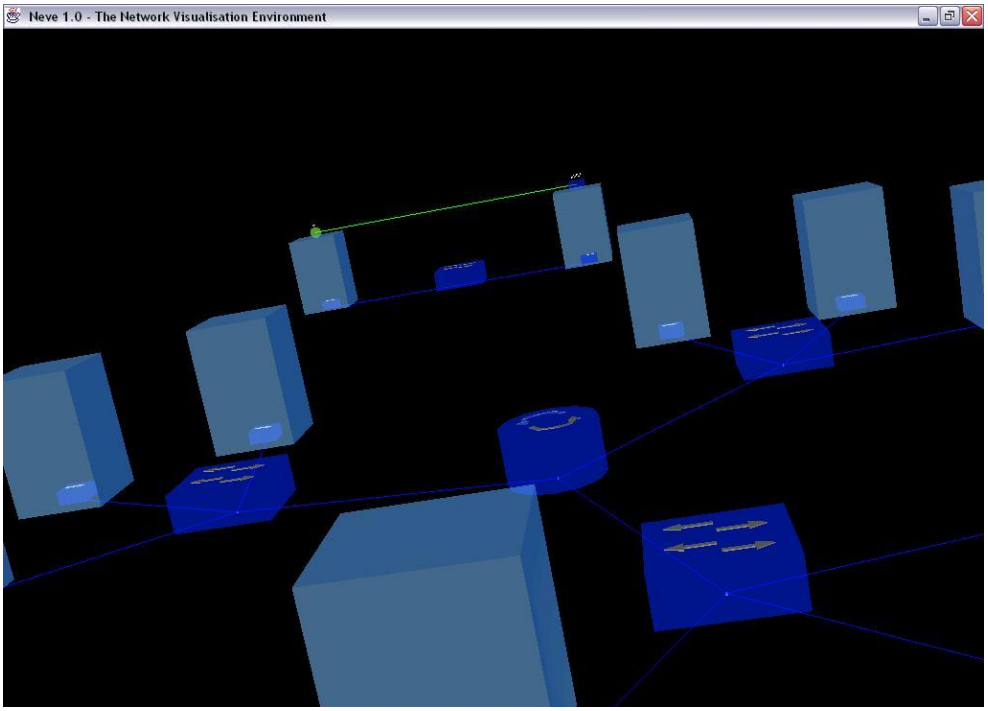


Abbildung 18: Das Verkehrsfluss-Szenario

Für das Szenario wurden jeweils MIO und MPO für jedes der drei benötigten Objekte definiert.

- `HTTP` im Paket `neve.data.mib`
- `HTTP` im Paket `neve.data.pib.standard`
- `WebServer` im Paket `neve.data.mib`
- `WebServer` im Paket `neve.data.pib.standard`
- `WebBrowser` im Paket `neve.data.mib`
- `WebBrowser` im Paket `neve.data.pib.standard`
- `ExampleNetCreator` im Paket `neve.data.applications`

`WebBrowser` und `WebServer` sind Protokollinstanzen und ihre MIOs und MPOs sind von `ProtocolInstance` und damit von `Internet` abgeleitet. Für sie kann also mit Hilfe des Attributes `Layer` eine Zugehörigkeit zu Schicht sieben der Kommunikationsarchitektur definiert werden, was dazu führt, dass sie, wie auch die Flüsse zwischen ihnen, oben auf den Endsystemen, über den Kabelverbindungen, angezeigt werden. Wird das Attribut während der Laufzeit verändert, so verändert sich auch die Y-Koordinate des dargestellten Verkehrsflusses.

Dies zeigt, dass Verkehrsflüsse auf allen Schichten grafisch darstellbar sind.

`HTTP` repräsentiert einen Verkehrsfluss des HTTP-Protokolls und ist als solcher von `PDU` abgeleitet.

Der `ExampleHTTPCreator` als beispielhafte Managementanwendung erzeugt nun ein Beispielnetz mit einem Webserver und einem Webbrowser. Auf Knopfdruck am Benutzer-Interface wird dann eine Darstellung eines Verkehrsflusses vom Webbrowser zum Webserver generiert, indem ein Objekt HTTP und eine Relation, die WebBrowser und WebServer über HTTP verbindet, erzeugt wird. Dies geschieht mit Hilfe der Operationen `createMIO(String name,String domain)` und `relate("verbunden-mit",webserverid,httpid,webbrowserid)`. Nach einer einstellbaren Zeitspanne wird das Verkehrsflussobjekt mit `removeMIO(int id)` wieder gelöscht. So ist es möglich, die Darstellung der Verkehrsflüsse dynamisch zu ändern, so dass eine Simulation möglich wird. Der Verkehrsfluss wird für diese Zeitspanne zwischen den Protokollinstanzen als grüne Linie sichtbar. Eine Darstellung dieses Settings in einer Detailaufnahme findet sich in Abbildung 19. Man erkennt links und rechts im Vordergrund die beiden Endsysteme als Quelle und Ziele des Verkehrsflusses. Als Transitsystem zwischen beiden fungiert ein Switch. Die Abstufung nach der Schicht, auf der ein Verkehrsfluss stattfindet, erfolgt auf der Höhenachse des Koordinatensystems. Der HTTP-Verkehrsfluss und seine Protokollinstanzen werden oben auf dem Endsystem dargestellt, da sie sich auf der logischen Schicht 7 des ISO/OSI-Modells befinden. Die Protokollinstanzen sind in diesem Fall ein Webserver, zu sehen auf dem linken System, und ein Webbrowser, zu sehen auf dem rechten System. Verkehrsflüsse in niedrigeren Schichten und deren Protokollinstanzen werden prinzipiell genauso dargestellt, aber auf der Höhenachse niedriger und mit anderen grafischen Repräsentationen. So läge die Darstellung einer IP-PDU etwa in der Mitte zwischen dem HTTP-Verkehrsfluss und der physikalischen Verbindung ganz unten.

Unabhängig von der Anzahl der Komponenten im dargestellten Netz konnten bei niedriger Systemauslastung Verkehrsflüsse bis herab zu einer Dauer von 25 ms dargestellt werden[27]. Für kürzere Zeitspannen war nicht immer eine Darstellung sichtbar, da die Visualisierung in Einzelfällen mehr Zeit in Anspruch nahm, als der Verkehrsfluss bestanden hatte. Dies bedeutet, dass eine Echtzeitsimulation langdauernder und in großen Intervallen auftretender Verkehrsflüsse möglich ist, mit durchschnittlich schnellen Rechensystemen aber eine Visualisierung von Flüssen unterer Schichten, bei denen mit sehr schnellen Übertragungsgeschwindigkeiten zu rechnen ist, aber nicht mehr einzeln erfolgen kann. Diese müssen deshalb akkumuliert werden. Es sind auch Simulationen denkbar, bei denen die Zeit langsamer abläuft als in Wirklichkeit. Für akkumulierte Verkehrsflüsse oder höherschichtige, die in nicht zu kleinen Zeitintervallen auftreten, wie beispielsweise Email oder HTTP dürfte das System aber in der Lage sein, auch mit durchschnittlich schnellen Rechensystemen Darstellungen annähernd in Echtzeit zu liefern.

In der Implementierung verbinden die MPOs der Verkehrsflüsse die beteiligten Systeme auf dem direkten Weg durch eine gerade Linie. Ein höherer Grad an Wirklichkeitstreue und eine

[27] für dieses Ergebnis ist ein sehr kurzes Intervall am Ringbuffer des Visualisierungsservices notwendig

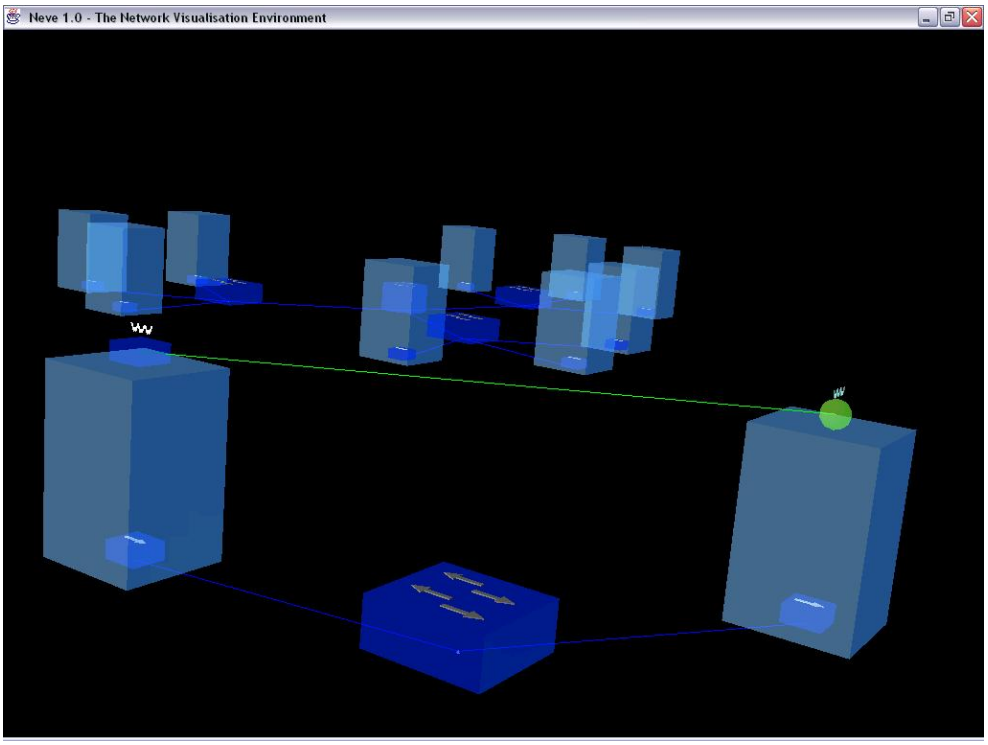


Abbildung 19: Detailansicht eines HTTP-Verkehrsflusses

genauere Übereinstimmung mit den logischen Hintergründen bei der Kommunikation würde erfordern, dass die Definition der Präsenationsobjekte für PDUs insoweit erweitert wird, dass die Verkehrsflüsse die ihnen zugrundeliegende physische und logische Topologie berücksichtigen. Die dazu notwendige Information findet sich bereits in der MIB.

4.5 Erkenntnisse aus der Umsetzung

Das System bietet für kleine bis mittelgroße Netzdarstellungen eine gute Performanz. Bei großen Netzen benötigt der Server eine längere Anlaufphase, um die Daten einzulesen und das Layout durchzuführen. Eine Optimierung des Layoutservices und eine Vergrößerung der Kapazität des Ringbuffers des Visualisers dürfte hierfür noch bessere Ergebnisse erwarten lassen. Auch durch eine Änderung des Settings, beispielsweise durch eine Verteilung von Server und Client auf zwei Rechner oder den Einsatz eines schnelleren Rechners würde die Performanz positiv beeinflusst werden.

Die vom Prototypen erreichte Darstellung ist konsistent und übersichtlich und erfüllt die im Kapitel 2.2.1 festgelegten Anforderungen. Sie wurde von Anwendern als sehr anschaulich bezeichnet, was die Erfüllung der psychologischen Paradigmen aus Kapitel 1.3 bestätigt.

Die Tests zeigten, dass es wichtig ist, die zur Verfügung stehende Prozessorkapazität sinnvoll einzusetzen, was durch die vielfältigen Möglichkeiten zur Konfiguration der Visualisierungsservices und -funktionen erleichtert wird. So konnte auch festgestellt werden, dass ein andauerndes Layout nur wenig Vorteile bringt, sobald eine ausgeglichene Darstellung eines Netzes erreicht ist. Da der Layoutservice sehr viel Rechenzeit braucht, ist es angebracht, ihn nur zu bestimmten Ereignissen, beispielsweise nach der Neuregistrierung von MPOs automatisch zu starten. Ein solcher Mechanismus kann automatisiert werden, was durch den Einsatz einfacher Heuristiken innerhalb der Operation handleNotification (Notification notification, Object handback) stattfinden kann. Dies gilt ebenso für den VLAN-Highlighter und es kann verallgemeinert werden, dass der Einsatz des Timerservices für Visualisierungsfunktionen in kurzen Abständen in der Regel weniger sinnvoll ist. Empfehlenswert ist ein Aufruf nur dann, wenn die Funktion wirklich benötigt wird, d.h. nach einem Start durch den Benutzer oder automatisch, wenn bestimmte Rahmenbedingungen gegeben sind.

Ansonsten konnte der Prototyp zeigen, dass die Definitionen der Dienste, Objekte und Visualisierungsfunktionen den Anforderungen gewachsen sind und Möglichkeiten zur Implementierung aller der für komplexe Darstellungen und Funktionalitäten erforderlichen Mechanismen bieten. Die gewählte Art und Weise der Interaktion mit den Objekten hat sich als vorteilhaft herausgestellt. Die Tests haben gezeigt, dass damit ein Eingreifen in den Prozess der Visualisierung, aber auch ein Verändern der Datenbasis gut möglich ist. Eine komplette Integration des Zugriffes in die dreidimensionale Darstellung wäre allerdings für zukünftige Versionen des Prototypen noch wünschenswert, die vorgesehene XML-Schnittstelle bietet die Grundlagen dafür.

Die Art und Weise der Anbindung der Managementanwendungen hat sich in den Tests ebenfalls als tragfähig erwiesen. Testanwendungen erhielten alle an der Visualisierungs-MIB vom Anwender durchgeführten Änderungen an der Konfiguration. Umgekehrt wurden alle Änderungen an der Datenbasis sofort in Visualisierungen umgesetzt.

Werden allerdings mehrere Managementanwendungen an einem Visualisierungssystem betrieben, die sich auf die selben Netzkomponenten beziehen, ist noch eine Erweiterung der Implementierung des Anwendungsservice notwendig, um zu vermeiden, dass für ein und dieselbe Netzkomponente mehrere Repräsentationen angelegt werden. Ein Erkennungsmechanismus

könnte verhindern, dass bereits bekannte Objekte noch einmal registriert werden, indem die ID eines bereits registrierten MIOs zurückgegeben wird.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Zusammenfassend kann festgestellt werden, dass es mit der entworfenen Architektur gelungen ist, die Grundlagen für ein VR-Netzmanagementsystem zu legen.

Im Unterschied zu anderen Ansätzen ist der Aufbau des Systems nicht proprietär, sondern es erfolgt eine Integration in standardisierte Netzmanagementumgebungen. Dies ermöglicht eine einfache Anbindung einer Vielzahl von Anwendungen und den Zugriff auf unterschiedliche Managementprotokolle oder andere Managementmechanismen.

Der Entwurf basiert auf einer Anforderungsanalyse, die von einem Top-Down Ansatz und psychologischen Paradigmen ausgeht. Dadurch wird eine Vielzahl möglicher Aufgaben des Netzmanagements und der praktischen Anwendungssituation berücksichtigt. In der Folge werden Mechanismen bereitgestellt, die es gestatten, auch für zunächst nicht berücksichtigte Anwendungsfälle Visualisierungen zu erstellen und mit diesen zu interagieren. Dies wird durch ein allgemeines Modell der Informationsvisualisierung unterstützt. Es stellt sicher, dass für beliebige Netzkomponenten Objekte formuliert werden können, die deren Darstellung beschreiben. Damit bleibt das System flexibel.

Neben der Internet-Kommunikationsarchitektur, die als grundlegendes Modell im Bereich Netzmanagement bereits im Entwurf berücksichtigt wurde, ist auch eine Erweiterung des Systems um weitere logische und grafische Kontexte möglich. Zur Anordnung der Objekte können mehrere geeignete Layoutservices angefertigt werden. So ist es denkbar, in den Visualisierungen auch Sachverhalte aus dem Systemmanagement oder aus gänzlich anderen Fachbereichen darzustellen und zu verwalten.

Ferner wurde ein Prototyp erstellt, der auf der Managementumgebung JMX basiert. Einerseits bietet JMX eine einheitliche Schnittstelle zur Interaktion mit den Visualisierungsdaten, andererseits kann auf eine Vielzahl von Managementanwendungen verschiedener Hersteller für die Datensammlung und -veränderung zurückgegriffen werden. Außerdem bietet JMX Schnittstellen für den Zugriff auf Managementprotokolle und andere Mechanismen des Netzmanagements. Gängige Managementplattformen wie IBM Tivoli und HP Open View bieten Schnittstellen zu JMX. Der Prototyp ist in der Lage, den aktuellen Zustand eines Netz bzw. dessen Komponenten dynamisch und dreidimensional zu visualisieren und er gibt dem Anwender die Möglichkeit, aktiv auf die Konfiguration der Ressourcen einzuwirken. Dabei steht ihm ein plattformunabhängiger Client zur Verfügung, der auf verschiedenen Betriebssystemen lauffähig ist und verschiedene Ausgabemedien wie gewöhnliche Monitore, aber auch Geräte zur Stereobildprojektion unterstützt.

Beispielhaft konnte anhand dreier Szenarien gezeigt werden, dass sich das Rahmenwerk zur Darstellung der Netztopologie, zur Visualisierung und Konfiguration von VLANs und zur Darstellung von Verkehrsflüssen eignet.

5.2 Ausblick

Als nächste Stufe der Entwicklung des Systems sollte die Anbindung des geschaffenen Kernsystems an bestehende Netzmanagementsysteme angestrebt werden. Im Münchner Wissenschaftsnetz könnte dies bedeuten, dass die Einträge in der hauseigenen Netzdokumentation, der VLAN-Datenbank und dem CNM (Customer Network Management) als dreidimensionale Netze dargestellt werden. Dies würde es den Administratoren erleichtern, sich in dieser umfrangreichen Datenbasis zu orientieren. Die dreidimensionale Darstellung verhindert dabei lange Suchpfade, die bei einer eindimensionalen textuellen Ausgabe der Datenbank in der Regel notwendig werden. Durch die menschgerechte Darstellung der Daten dürfte ferner die Suche nach Fehlern erleichtert werden, beispielsweise bei der Vergabe von VLAN-Zugehörigkeiten.

Durch einen weiteren Ausbau des Benutzerinterfaces könnte eine komplette Integration des Interaktionsteiles in die Visualisierung stattfinden. Die ensprechende Funktionalität wird zwar bereits vom Browser bereitgestellt, ein ergonomisch gut gestaltetes Benutzerinterface sollte aber auch intuitive Methoden zur Bedienung vorsehen.

Da alle Systeme vom gleichen Typ in der Darstellung momentan noch identisch aussehen, muss hier eine Differenzierung erfolgen. Wie im ersten Prototypen sollten bestimmte Kennzahlen der dargestellten Komponenten auch im vorliegenden weiterentwickelten System bereits bei Berührung mit dem Mauspfeil angezeigt werden können. Die Anzeige von IP-Adressen und Hostnamen beispielsweise erleichtert den Anwendern die Identifikation und Zuordnung der grafischen Darstellungen zu den physischen Komponenten. Ferner sollte das Benutzerinterface dahingehend erweitert werden, dass bei einem Mausklick auf eine Komponente ein Fenster geöffnet wird, in dem dann die Attribute des MIOs und des MPOs geändert und Operationen ausgeführt werden können. Visualisierungsfunktionen sollten per Tastendruck o.ä. in das Bild eingeblendet und auf dieselbe Art und Weise bedient werden.

Für verschiedene Aufgaben des Managements sind in der Praxis außerdem oft auch Listen notwendig. Deshalb ist es sinnvoll, neben einer grafischen Ausgabe beispielsweise von VLANs auch traditionelle Listen zur Verfügung zu stellen. Hierzu könnten Visualisierungsfunktionen implementiert werden, die aus den Einträgen im Informationsraum entsprechende Listen erstellen und diese dann auf dem Bildschirm und/oder einem Drucker ausgeben.

Eine weitere Entwicklung des Systems sollte aber nicht nur auf pragmatische Gesichtspunkte eingehen, sondern auch die Skalierbarkeit erhöhen. So ist es denkbar, Übertragungsprotokolle zur Kommunikation zwischen Visualisierungsservern zu entwickeln, um so eine verteilte Visualisierung größerer Netze wie dem Internet zu erreichen. Beispielsweise könnte jedes Rechenzentrum einen oder mehreren Visualisierungsserver zur Verfügung stellen, die dann untereinander ihre Daten austauschen. Die jeweiligen Zuständigkeitsbereiche bleiben so sinnvoll verwaltbar. Darüber hinaus kann die Administration des Informationsraums beispielsweise nur den örtlichen Servern erlaubt oder die Leserechte eingeschränkt werden, der Präsentationsraum aber anderen Servern auch schreibend zugänglich gemacht werden. So kann jedes Rechenzentrum nach außen eine aktuelle Darstellung seines Netzes anbieten. Die Struktur des entworfenen Systems stellt ein hierarchisches Schichtenmodell mit Zugriffspunkten (den Diensten) dar und unterstützt eine

solche Verteilung. Es ist in Abbildung 20 konkret dargestellt.

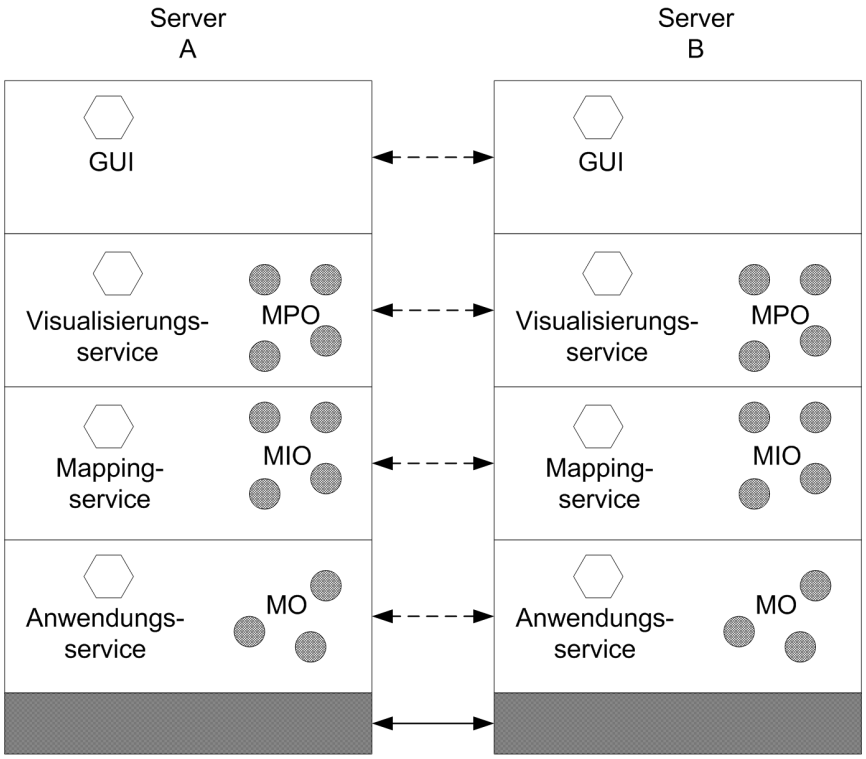


Abbildung 20: Das Schichtenmodell der Architektur, dessen Datenobjekte und Zugriffspunkte

Die Architektur sieht mehrere Benutzerinterfaces vor, die auf das selbe Darstellungsmodell zugreifen. Dadurch ist es naheliegend, einen Dienst auf dem Server zu implementieren, der für jeden Benutzer ein MPO erzeugt und die Positionen der einzelnen Benutzer innerhalb des visualisierten Universums mit den Koordinaten des MPOs abgleicht. So können sich alle Anwender des Systems im Universum sehen. Durch die Verbindung einer solchen Technik mit Funktionen zum Nachrichtenaustausch könnte die Kernarchitektur in Richtung CSCW-System weiterentwickelt werden. So könnten Projektgruppen auch über grössere Distanzen gemeinsam an der Administration eines Netzes arbeiten und dabei Informationen austauschen.

Abkürzungsverzeichnis

AANF Architekturanforderung

AM Attributmenge

API Application Programming Interface

AVO Abstraktes Visualisierungsobjekt

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

FTP File Transfer Protocol

GUI Graphical User Interface

HTTP Hyper Text Transfer Protocol

IANF Interaktionsanforderung

IM Informationsmenge

IO Informationsobjekt

IP Internet Protocol

IR Informationsraum

IS Informationsstruktur

ISO International Standard Organisation

IVis Informationsvisualisierung

JMX Java Management Extensions

LAN Local Area Network

LRZ Leibniz-Rechenzentrum

MANF Managementanforderung

MBean Managed Bean

MIB Managementinformationsbasis

MPO Managementpräsentationsobjekt

MIO Managementinformationsobjekt

MO Managementobjekt

MOC Managed Object Class(es)

MVC Model-View-Control Paradigma

MWN Münchner Wissenschaftsnetz

NEVE NEtwork Visualisation Environment

OSI Open Systems Interconnection

PDU Protocol Data Unit

PIB Präsentationsinformationsbasis

PM Präsentationsmenge

PO Präsentationsobjekt

PR Präsentationsraum

RMI Remote Method Invocation

SMFA Systems Management Functional Areas

SMTP Simple Mail Transfer Protocol

SNMP Simple Network Management Protocol

TCP Transmission Control Protocol

UDP User Datagram Protocol

VANF Visualisierungsanforderung

VENoM virtual environment network monitoring

VLAN Virtual Local Area Network

VR Virtuelle Realität

VRML Virtual Reality Modeling Language

Abbildungsverzeichnis

Literaturverzeichnis

[AGL+ 00] ABEL, P., P. GROS, D. LOISEL, C. RUSSO DOS SANTOS und J.P. PARIS: *Cybernet: A framework for managing networks using 3D metaphoric worlds*. In: *Annales des Telecommunications*, April 2000.

[anfy] *ANFY3D, 3D Engine for Java*. http://www.anfy3d.com.

[Baur 00] BAUR, TIMO: *Plattform- und anwendungsunabhängige Visualisierung und Analyse von Netzwerkstrukturen, Entwurf und Implementierung des Prototypen 3D-Net*. Systementwicklungsprojekt, Lehrstuhl für Rechnerkommunikation und maschinelle Deduktion, Fakultät für Informatik, TU München, Mai 2000.

[Bert 81] BERTIN, J.: *Graphics and Graphic Information Processing*. Walter de Gruyter & Co., Berlin, 1981.

[BEW 95] BECKER, RICHARD A., STEPHEN G. EICK und ALLAN R. WILKS: *Visualizing Network Data*. IEEE Transactions on Visualization and Computer Graphics, 1(1):16–28, 1995.

[Borm 94] BORMANN, SVEN: *Virtuelle Realität: Genese und Evaluation*. Addison Wesley, Bonn, Paris; Reading, Mass., 1994.

[Buxt 92] BUXTON, W.: *Telepresence: Integrating Shared Task and Person Spaces*. In: BAECKER, R. (Herausgeber): *Readings in Groupware and Computer-Supported Cooperative Work*. Morgan Kaufmann Publishers, 1992.

[Chi 00] CHI, ED H.: *A Taxonomy of Visualization Techniques using the Data State Reference Model*. In: *Proceedings of the Symposium on Information Visualization (InfoVis '00)*, Seiten 69–75, Salt Lake City, Utah, 2000. IEEE Press.

[CLFZ 93] CRUTCHER, LAURENCE A., AUREL A. LAZAR, STEVEN FEINER und MICHELLE X. ZHOU: *Management of Broadband Networks Using a 3D Virtual World*. In: *HPDC*, Seiten 306–315, 1993.

[CMS 98] CARD, S., J. D. MACKINLAY und B. SCHNEIDERMAN (Herausgeber): *Readings in Information Visualisation: Using Vision to think*. Morgan Kaufmann Publishers, San Mateo, CA, 1998.

[CuEg 98] CUBETA, A. und D. EGTS: *VENoM - Virtual Environment for Network Monitoring*. http://www.nrl.navy.mil/CCS/people/cubeta/venom/paper.html, 1998.

[DSL 96] DEDE, C., M. SALZMANN und R. LOFTIN: *Science Space: Virtual realities for learning complex and abstract scientific concepts*. In: *, IEEE Virtual Reality Annual International Symposium*, Seiten 246–253. 1996.

[Eisl 10] EISLER, RUDOLF: *Wörterbuch der philosophischen Begriffe [in drei Bänden]*, Seiten 225–254. Verlag Mittler und Sohn, Berlin, Dritte Auflage, 1910.

[FPF 88] FAIRCHILD, K.M., S.E. POLTROCK und G.W. FURNAS: *Three-dimensional Graphic Representations of Large Knowledge Bases*. In: *Cognitive Science and Its Applications for Human-Computer Interaction*, Seiten 201–233, Fairlawn, NJ, 1988. Lawrence Erlbaum Associates.

[FWK 98] FREEMAN, L., WEBSTER und KIRKE: *Exploring social structure using dynamic three-dimensional color images*. In: *Social Networks 20*, Seiten 109–118. Elsevier Science B.V., 1998. http://eclectic.ss.uci.edu/ lin/77.pdf.

[Garb 91] GARBE, KLAUS: *Management von Rechnernetzen*. Leitfäden der angewandten Informatik. Teubner Verlag, Stuttgart, 1991.

[Gibs 87] GIBSON, WILLIAM: *Neuromancer*. Heyne Verlag, München, 1987.

[HaMc 90] HABER, R.B. und D.A. MCNABB: *Visualization Idioms: A Conceptual Model for Scientific Visualization Systems.* In: NIELSON, G.M., S. B. und L. J. ROSENBLUM (Herausgeber): *Visualization in Scientific Computing*, Seiten 74–93. IEEE Computer Society Press, 1990. BIBLIOGRAPHY 245.

[HAN 99] HEGERING, H.-G., S. ABECK und B. NEUMAIER: *Integriertes Management vernetzter Systeme*. dpunkt-Verlag, 1999.

[IEEE 802.1Q] *IEEE Std 802.1Q-1998, IEEE standard for local and metropolitan area networks: Virtual Bridged Local Area Networks*. Technischer Bericht, Institute of electrical and electronics engineers, inc., New York, USA, 1999.

[ISO 7498-4] *Addendum 4 zum ISO/OSI-Referenzmodell*. Standard 7498-4, International Standard Organisation.

[KNK 99] KOUTSOFIOS, E., S. NORTH und D. KEIM: *Visualizing Large Telecommunication Data Sets*. In: *IEEE Computer Graphics and Applications*, Seiten 16–19, Mai/Juni 1999.

[KrPo 88] KRASNER, G. und S. POPE: *A cookbook for using the model-view-controller paradigm in smalltalk-80*. In: *Journal of Object Oriented Programming*, Nummer 3 in *1*, Seiten 26–49, August/September 1988.

[KuFo 96] KUMAR, ARUNA und RICHARD H. FOWLER: *A Spring Modeling Algorithm to Position Nodes*. Technischer Bericht, Department of Computer Science - University of Texas, Edinburg, Texas 78539, 1996.

[Meyers] *Meyers Lexikon - Das Wissen A-Z, Online-Version*, 1993. http://www.iicm.edu/meyers.

[MX4J] *MX4J, OpenSource JMX Implementierung*. http://mx4j.sourceforge.net.

[Nova 92] NOVAK, MARCOS: *Liquid Architectures in Cyberspace*. In: BENEDIKT, MICHAEL (Herausgeber): *Cyberspace - First Steps*, Seiten 225–254. MIT Press, Cambridge, Mass., London, 2 Auflage, 1992. 1. Auflage 1991.

[PaPh 01] PATTISON, TIM und MATTHEW PHILLIPS: *View Coordination Architecture*

for Information Visualisation. In: EADES, PETER und TIM PATTISON (Herausgeber): *Conferences in Research and Practice in Information Technology*, Band 9, Sydney, December 2001. Australian Symposium on Information Visualisation.

[Poin 97] POINTNER, ANTON: *Entwicklung und prototypische Implementierung einer graphischen Bedienoberfläche für das VLAN-Management*. Diplomarbeit, Technische Universität München, 1997.

[Proe 02] PROEBSTER, WALTER: *Rechnernetze: Technik, Protokolle, Systeme, Anwendungen*. Oldenbourg, zweite Auflage, 2002.

[Sand 90] SANDKÜHLER, HANS-JÖRG (Herausgeber): *Europäische Enzyklopädie zu Philosophie und Wissenschaften [in vier Bänden]*. Felix Meiner Verlag, 1990. in Zusammenarbeit mit dem Istituto Italiano Per Gli Studi Filosofici.

[ScBa 99] SCHNOTZ, W. und M. BANNERT: *Einflüsse der Visualisierungsform auf die Konstruktion mentaler Modelle beim Text- und Bildverstehen*. In: *Zeitschrift für experimentelle Psychologie*, Band 46 (3), Seiten 217–236. 1999.

[ScBu 01] SCHWAN, STEPHAN und JÜRGEN BUDER: *Lernen mit virtuellen Realitäten: ein pädagogisch-psychologischer Überblick*. Überblicksreferat auf der 8. Fachtagung Pädadgogische Psychologie, Landau, September 2001.

[sun 99] *Java Management Extensions White Paper*. Technischer Bericht, Sun Microsystems, Inc. (Hrsg.), Palo Alto, 1999.

[sun 00] *Java Management Extensions Instrumentations and Agent Specification, v1.0*. Spezifikation, Sun Microsystems, Inc. (Hrsg.), Palo Alto, 2000.

[sun 02] *Java Management Extensions Instrumentations and Agent Specification, v1.1*. Spezifikation, Sun Microsystems, Inc. (Hrsg.), Palo Alto, Mai 2002.

[TBET 99] TOLLIS, IOANNIS, GIUSEPPE DI BATTISTA, PETER EADES und ROBERTO TAMASSIA: *Graph Drawing: Algorithms for the visualisation of graphs*. Prentice Hall, 1999.

[Weid 89] WEIDENMANN, B.: *Der mentale Aufwand beim Fernsehen*. In: J.GROBEL und P. WINTERHOFF-SPURK (Herausgeber): *Empirische Medienpsychologie*, Seiten 134–150. PVU, München, 1989.

[Winn 93] WINN, W.: *A conceptual basis for educational applications of virtual reality*. Technical Publication R-93-9, Human Interface Technology Laboratory of the Washington Technology Center, Seattle: University of Washington, 1993.

[Wüns 98] WÜNSCHE, VEIKKO: *Eine Begriffswelt für die Informationsvisualisierung*. In: *Rostocker Informatik Berichte*, Band 21. Fraunhofer Institut für Arbeitswirtschaft und Organisation, Stuttgart, 1998.