INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Konzeption eines Werkzeugs zur Erfassung von Betreiberanforderungen an ein integriertes Netz- und Systemmanagement

Christian Fischer

INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Konzeption eines Werkzeugs zur Erfassung von Betreiberanforderungen an ein integriertes Netz- und Systemmanagement

Bearbeiter	: Christian Fischer
Aufgabensteller	: Prof. Dr. Heinz-Gerd Hegering
Betreuer	: Dr. Sebastian Abeck
	: Dipl. Inf. Peter Segner
Abgabedatum	: 15. Februar 1995

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

An dieser Stelle möchte ich mich bei meinen Betreuern, Herrn Dr. Sebastian Abeck und Herrn Peter Segner, bedanken, die mich mit ihren Anregungen und ihrer Kritik unterstützt haben.

Ein weiterer Dank gilt allen Mitarbeitern des Lehrstuhls und meinen Studienkollegen, die durch zahlreiche Diskussionen wertvolle Hilfe geleistet haben.

Ehrenwörtliche Erklärung

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

München, den 15. Februar 1995

....

(Unterschrift des Kandidaten)

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Durch die seit dem Anfang der 90er Jahre anhaltende wirtschaftliche Rezession geraten viele Unternehmen unter starken Wettbewerbsdruck. Es ist daher nicht verwunderlich, daß nach neuen Wegen gesucht wird, um auf dem Markt größere Wettbewerbsfähigkeit zu erlangen. Einen Ansatzpunkt bietet hierbei das Informationsmanagement, das genauso zu einem Produktionsfaktor geworden ist wie die menschliche Arbeitskraft oder auch der Produktionsstandort (vgl. [Hei91]). Durch die wachsende Dezentralisierung von Unternehmensstrukturen, wobei z.B. ein Grund in der zunehmenden Öffnung der Märkte liegt (Schlagwort: EU-Binnenmarkt), steigt die Kommunikationsdichte stark an. Zudem expandiert auch die Informationsintensität, da ein Großteil der Unternehmen versucht, die jeweilige Produktpalette zu erweitern, um so das Unternehmen auf eine breitere Basis zu stellen, d.h. unabhängiger von Krisen einzelner Branchenzweige zu machen. Aufgrund der Steigerung der Intensität von Kommunikation und Information erscheint die Verwendung von Rechnernetzen als das geeignete Mittel, um dieses Dilemma zu lösen.

Durch die rasch fortschreitenden Entwicklungen im Bereich der Netztechnologien wurden die Netzbetreiber überrascht. Sie sind gezwungen, die bisher fehlenden Betriebskonzepte für ihre Corporate Networks zu entwerfen. Einen Rahmen für ein derartiges Konzept stellt das am Lehrstuhl von Prof. Hegering entwickelte Rahmenbetriebskonzept dar. Wie der Autor zeigen wird, ist die Erstellung eines Betriebskonzepts nicht trivial und ohne ein unterstützendes Werkzeug kaum zu bewerkstelligen.

Früher wurde die Kommunikation mittels Telefon fernmündlich, bzw. über eine Rohrpost schriftlich erledigt. Das Aufkommen neuer Kommunikationstechnologien, z.B. von Netzverbunden, wurde als Chance gesehen, diese effizient im Unternehmen einzusetzen. Sie boten einige Vorteile gegenüber den alten Medien:

- Transport großer Datenmengen in verhältnismäßig kurzer Zeit und
- Verkürzung der Laufzeiten von Nutzinformation.

Durch den gesteigerten Kommunikationsbedarf gewannen Netzverbunde immer weiter an Bedeutung. Heute ist es selbstverständlich, daß Informationsverarbeitung nicht ausschließlich lokal durchgeführt, sondern in einen unternehmensweiten Kontext eingefügt wird. Viele Unternehmensnetze (Corporate Networks) umspannen mittlerweile die gesamte Welt. Die Bestrebungen, verstärkt verteilte, auf Client/Server-Architektur basierende Systeme einzusetzen, fördert die Ausbreitung von Netzen. Zum Management dieser Netzverbunde werden Werkzeuge eingesetzt, deren Funktionen meist nur ein stark eingeschränktes Problemfeld bearbeiten können (Management einer Komponentenart eines bestimmten Herstellers, d.h. jeder Hersteller stellt eigene Werkzeuge zur Verfügung, die speziell auf sein Produkt abgestimmt sind). Die Folge daraus ist, daß der Betreiber eines Corporate Networks eine große Anzahl von Managementwerkzeugen einsetzen muß, die meist nicht aufeinander abgestimmt sind. Dieses Nebeneinander von Managementwerkzeugen ist äußerst ineffizient, deshalb streben die Entwicklungen in die Richtung eines integrierten Managementansatzes, dessen Grundlage Informationsmodelle sind. Es existieren zwei verschiedene Informationsmodelle,

- das OSI-Informationsmodell, basierend auf einem objektorientierten Ansatz und
- das Internet-Informationsmodell, dessen Grundlage ein statischer Ansatz ist, der durch eine Client/Server-Architektur realisiert wird, wobei der Client Manager und der Server Agent genannt wird.

Die beiden Informationsmodelle bedingen folglich auch verschiedene Kommunikationsmodelle, wodurch sich nur ein Problemfeld für das Netz- und Systemmanagement ergibt.

1.1 Problematik des integrierten Netz- und Systemmanagements

Die Corporate Networks sind heterogene Netze, die durch proprietäre Managementsysteme verwaltet werden. Diese Managementsysteme arbeiten auf ihren eigenen Datenbeständen, wodurch der Gesamtdatenbestand redundant und inkonsistent wird. Der verfolgte Ansatz besteht darin, die einzelnen Managementsysteme in eine Managementarchitektur zu integrieren, deren Implementierung z.B. eine Managementplattform ist. Es lassen sich drei Integrationsarten unterscheiden:

1. Die *Oberflächenintegration*, wobei z.B. in eine Managementplattform nur ein Knopf eingefügt wird, der das entsprechende Managementsystem aktiviert. Die

integrierten Managementsysteme arbeiten weiter isoliert, so daß sich keine logischen Verbindungen zwischen ihnen herstellen lassen. Jedes Managementsystem bietet dem Anwender seine eigene Oberfläche an, als Folge daraus ergibt sich ein inkonsistenter Gesamteindruck.

2. Die *Proxy-Integration*, bei der zwischen dem Managementwerkzeug und der Managementarchitektur ein Proxy-Agent geschaltet wird, wodurch eine einheitliche Darstellung gewährleistet wird. Die Nachteile liegen darin, daß keine gemeinsame Datenbasis besteht und die Schnittstellen vom eingesetzten System abhängen.

3. Bei der *vollständigen Integration* werden Agenten etabliert, die einen direkten Zugriff, z.B. auf MIBs, gestatten und auch logische Verbindungen zwischen verschiedenen Funktionen herstellen können.

Zusammenfassend ist festzustellen, daß es *das* integrierte Netz- und Systemmanagement nicht gibt, da die Problematik stark durch die Umgebung eines Netzbetreibers und die jeweiligen Präferenzen beeinflußt werden, d.h. das Netz- und Systemmanagement muß individuell auf einen Netzbetreiber abgestimmt sein (vgl. [HA93]).

1.2 Ziele eines integrierten Netz- und Systemmanagements

Ziel ist es, den möglichst effizienten Einsatz von Ressourcen durch

- Überwachung (Monitoring),
- geeignete Konfiguration von Komponenten und Werkzeugen und
- Fehlererkennung und -behebung

zu erreichen. Eine vollständige Integration der Managementsysteme ermöglicht eine einheitliche Datenbasis, die frei von Redundanzen und Inkonsistenzen ist.

Um das Management von einer zentralen Stelle für das Gesamtnetz durchzuführen, werden derzeit verstärkt Managementplattformen eingesetzt. Diese sind mit einer Schnittstelle (sog. Application Programming Interface, API) ausgerüstet, die die Integration der oben bereits erwähnten Managementwerkzeuge in die Plattform erlaubt. Der effiziente Einsatz einer solchen Plattform setzt aber ein Betriebskonzept (Policy) für das Corporate Network von Seiten des Netzbetreibers voraus. Dieses Betriebskonzept beschreibt in einem Top-Down-Ansatz, wie welche Art von Diensten von dem Netzbetreiber zu erbringen sind. Bisher fehlt ein solches Betriebskonzept bei den meisten Betreibern. Das Rahmenbetriebskonzept liefert eine Vorgehensweise (eine Methodik), anhand der ein individuelles Betriebskonzept für die Unternehmensumgebung entwickelt werden kann.

Der Autor stellt ein Werkzeug vor, das die Erstellung eines Rahmenbetriebskonzepts unterstützt. Die Arbeit ist unterteilt in zwei Teile:

- Teil I: *Grundlagen zur Erstellung eines Werkzeugs auf Basis des Rahmenbetriebskonzepts:*
  In diesem Teil wird das Rahmenbetriebskonzept (Kapitel 2) vorgestellt, die Motivation für ein Werkzeug gegeben (Kapitel 3) und daran anschließend werden Anforderungen an ein Werkzeug (Kapitel 4) formuliert.
- Teil II: *Konkrete Implementierung eines Werkzeugs zur Erfassung des Rahmenbetriebskonzepts:*
  Der zweite Teil der Arbeit ist stark technisch geprägt, so wird in Kapitel 5 eine Architektur entwickelt, die die Grundlage für die Implementierung eines Werkzeugs (Kapitel 6) bildet. Diese Implementierung wird in Kapitel 7 kritisch betrachtet.

Der Schwerpunkt der Arbeit lag auf einer prototypischen Entwicklung eines Werkzeugs, das die Erfassung des Ist-Zustandes im Bereich Netz- und Systemmanagement eines Netzbetreibers unterstützt. Im Anhang findet sich das Benutzerhandbuch zu diesem Werkzeug.

Teil I

Grundlagen zur Erstellung eines Werkzeugs auf Basis des Rahmenbetriebskonzepts

Kapitel 2

Das Rahmenbetriebskonzept

Die Ursache für die Heterogenität der Corporate Networks liegt in der gewachsenen Struktur der Rechner- und Kommunikationslandschaft. Aus diesem Grund ist es notwendig, neue Konzepte für die Erbringung von Dienstleistungen und die damit verbundene Verteilung von Ressourcen zu erarbeiten und diese in einen neuen Kontext zwischen Organisationseinheiten und deren Schnittstelle zur Außenwelt (Kunden/Lieferanten-Beziehung) einzubetten. Diese Schnittstelle wird „Dienst“ genannt. Der „Dienstnutzer“ (Kunde, User) ist nicht an den Abläufen, die der Diensterbringung dienen, interessiert, sondern am Leistungsumfang, an der verfügbaren Qualität und den damit anfallenden Kosten. Wie aber die konkreten Abläufe aussehen, ist nur den jeweiligen Zuständigen bekannt. Fallen beispielsweise Mitarbeiter der Störungsbearbeitung aus irgendeinem Grund aus, so muß ggf. mit der Fehlerbearbeitung gewartet werden; falls dies nicht möglich ist, muß improvisiert werden. Das Rahmenbetriebskonzept liefert einen Lösungsansatz für die umfassende Beschreibung und Analyse bestehender Abläufe innerhalb einer Organisation und der wechselseitigen Beziehungen.

2.1 Überblick

Zum besseren Verständnis wird im Vorfeld das Rahmenbetriebskonzept (RBK) kurz vorgestellt, um einige grundlegende Begriffe einzuführen.

Allgemeine Begriffe

- *Rahmenbetriebskonzept:*
  Organisatorische und technische Festlegungen, die im Bereich von IV-Abteilungen für ein effizientes und effektives Erreichen der Unternehmensziele geregelt werden müssen.

- *Dienstleistungskonzept:*
  Darstellung der Dienstleister und Dienstnutzer, sowie deren Dienstleistungsbeziehungen.

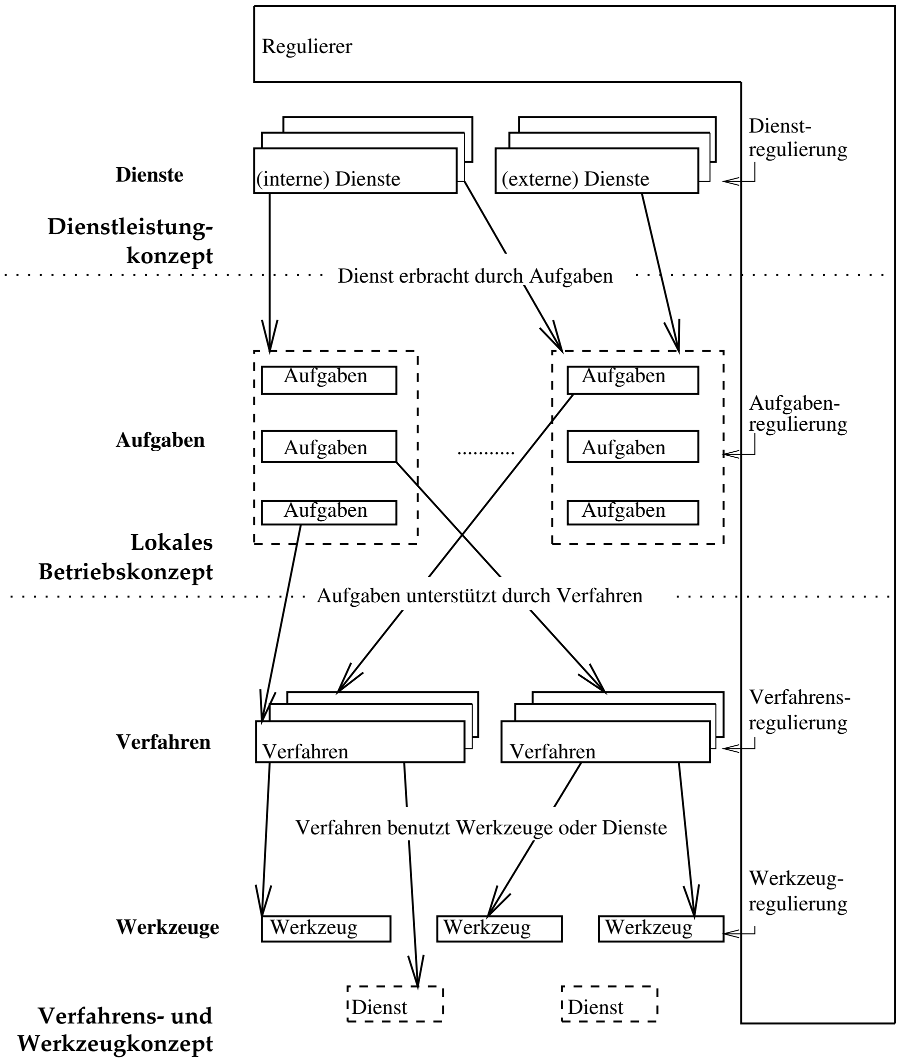

Abbildung 2.1: Schema des Rahmenbetriebskonzepts

- *Lokales Betriebskonzept:*
  Definition der Aufgaben, die von einem Dienstleister ausgeführt werden müssen, um die von ihm angebotenen Dienste erbringen zu können.

- *Verfahrenskonzept:*
  Beschreibung der zur Aufgabenerledigung eingesetzten Verfahren.

- *Werkzeugkonzept:*
  Beschreibung der zur Realisierung der Verfahren eingesetzten Betriebsmittel.

Diese Liste wird im späteren Verlauf noch weiter ergänzt. Im Anschluß werden die Ebenen des Rahmenbetriebskonzepts erläutert.

2.2 Dienstleistungskonzept (Dienstebene)

Das Dienstleistungskonzept beschäftigt sich mit dem Begriff des Dienstes und seinen Beziehungen zu anderen Objekten. Ein Dienst wird durch seinen Inhalt, sowie durch

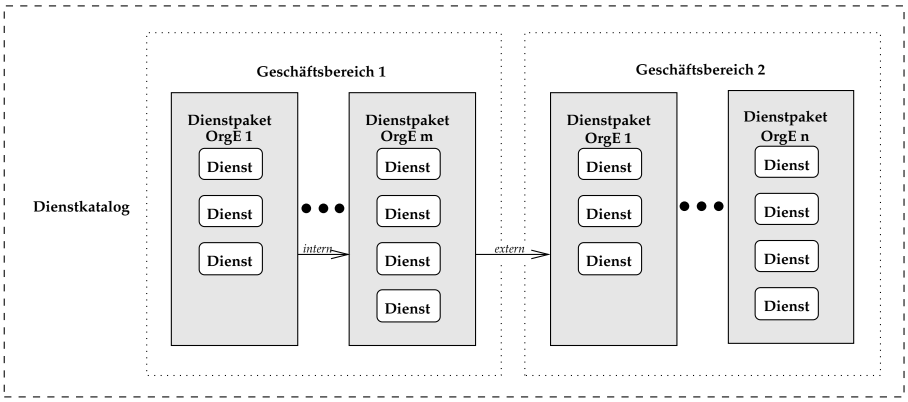

Abbildung 2.2: Schematische Darstellung der Dienstebene

Meßgrößen, die der Abrechnung dienen (*accountable units*), eindeutig bestimmt. Die Charakteristika der Dienste werden durch Attribute beschrieben.

Häufig werden nicht nur einzelne Dienste von einer Organisationseinheit angeboten, sondern ganze Dienstpakete, die aus einer Zusammenfassung von Diensten hervorgegangen sind. So bietet z.B. ein Netzkontroll-Zentrum das Dienstpaket „Netzzugang bereitstellen“ an, das sich aus den Diensten „Planung“, „Dimensionierung“, „Beschaffung“, „Installation“, „Abnahme“ und „Last-Level-Support“ zusammensetzt. Die Bereitstellung von Dienstpaketen erhöht die Akzeptanz beim Kunden, da ihm eine „Gesamtlösung“ angeboten wird. Der Netzbetreiber erhält dadurch eine übersichtliche Struktur seiner Dienstleistungen.

Nachfolgend werden die innerhalb dieser Rahmenbetriebskonzeptebene benötigten Begriffe definiert.

Begriffe des Dienstleistungskonzepts

- *Dienst:*
  Der Dienst ist eine Arbeitsleistung, die von einem im Dienstleistungskonzept auftretenden Dienstleister nach außen angeboten wird und gemäß einer Vereinbarung von einem Dienstnutzer in Anspruch genommen werden kann. Ein Dienst muß die Eigenschaft aufweisen, im Grundsatz abrechenbar zu sein, sowie eine gewisse Dienstqualität (Quality of Service) zu garantieren.

- *Interner Dienst:*
  Dienst, den ein Dienstnutzer von einem Dienstleister in Anspruch nimmt, wobei beide in der gleichen Organisationseinheit tätig sind.

- *Externer Dienst:*
  Dienst, bei dem Dienstnutzer und Dienstleister nicht der gleichen Organisationseinheit angehören.

- *Dienstpaket:*
  Ein Dienstpaket umfaßt einen oder mehrere Dienste.

- *Dienstkatalog:*
  Zusammenstellung aller von einer Organisationseinheit erbrachten Dienstpakete und Dienste; ein Dienstkatalog ist erst zu dem Zeitpunkt vollständig, zu dem alle Dienstpakete zu Diensten verfeinert wurden.

- *Organisationseinheit:*
  Eine Organisationseinheit stellt Dienste zur Verfügung bzw. nutzt sie. Eine Organisationseinheit kann noch unterteilt werden in mehrere (untergeordnete) Organisationseinheiten bzw. sie kann Bestandteil einer (übergeordneten) Organisationseinheit sein.

- *Dienstleister, Lieferant:*
  Organisation, die einen Dienst anbietet.

- *Dienstnutzer, Kunde:*
  Organisationseinheit oder einzelne Person, die einen Dienst in Anspruch nimmt.

- *Regulierer:*
  Eine Einheit, die auf jeder Ebene des Rahmenbetriebskonzept regulierend eingreifen kann.

2.3 Lokales Betriebskonzept (Aufgabenebene)

Wie bereits angesprochen, stellen Dienste die Schnittstelle zwischen dem Dienstanbieter (z.B. Netzbetreiber) und dem Dienstnutzer (z.B. Benutzer einer verteilten Anwendung) dar. Die Erbringung eines Dienstes übernehmen Organisationseinheiten durch Kontrolle und Steuerung von Aufgaben. Die Güte eines Dienstes ist somit

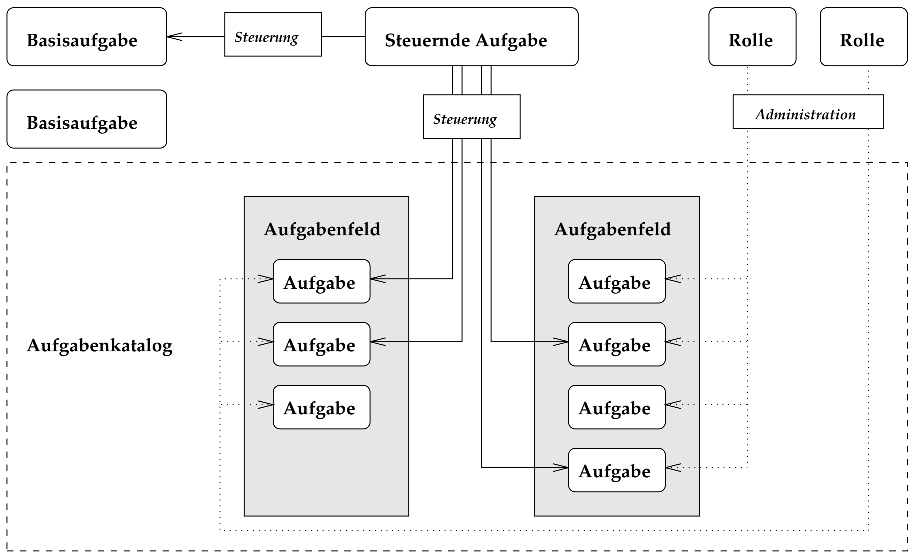

Abbildung 2.3: Schematische Darstellung der Aufgabenebene

abhängig von der Güte der zugrundeliegenden Aufgaben.

Begriffe des Lokalen Betriebskonzepts

- *Aufgabe:*
  Eine innerhalb einer Organisationseinheit erbrachte Tätigkeit.
- *Dienstbezogene Aufgabe:*
  Aufgabe, die direkt zur Erbringung eines Dienstes beiträgt.
- *Basisaufgabe:*
  Aufgabe, die <u>nicht</u> direkt einer Diensterbringung zugeordnet werden kann.
- *Steuernde Aufgabe:*
  Aufgabe, die den Ablauf der (Ausführungs-)Aufgaben zur Diensterbringung koordiniert.

- *Ausführungs-Aufgabe:*
  Aufgabe, die einen in sich abgeschlossenen Teilbereich zur Diensterbringung, unter Abstützung auf Verfahren, definiert.

- *Rolle:*
  Profil für den Mitarbeiter bzw. die Mitarbeitergruppe, die für die Ausführung einer Aufgabe zuständig ist.

- *Aufgabenfeld:*
  Zusammenfassung aller Aufgaben, die in einer logischen und inhaltlichen Beziehung zueinander stehen.

- *Aufgabenkatalog:*
  Zusammenstellung aller Aufgabenfelder und Aufgaben einer Organisationseinheit. Die Zuordnung der Aufgabenfelder und Aufgaben zu den Diensten, die durch sie erbracht werden, muß gegeben sein. Der Aufgabenkatalog ist erst dann vollständig, wenn *alle* Aufgabenfelder zu Aufgaben verfeinert worden sind.

2.4 Verfahrens- und Werkzeugkonzept

Betrachtet man bereits bestehende Verfahren (Arbeitsabläufe), so lassen sich zwei Arten von Verfahren definieren:

- lokale Verfahren, die nur innerhalb einer Organisationseinheit eingesetzt werden und

- globale Verfahren, die Organisationseinheiten-übergreifend eingesetzt werden.

Ein Verfahren setzt sich aus einer Abfolge von elementaren Arbeitsschritten (Aktionen) zusammen. Aufgaben stützen sich auf Verfahren ab.

Begriffe des Verfahrens- und Werkzeugkonzepts

- *Verfahren:*
  Regelung einer Abfolge von Aktionen oder Teilverfahren unter Angabe der Informationsflüsse.

- *Lokales Verfahren:*
  Verfahren, das ausschließlich innerhalb einer Organisationseinheit zur Anwendung kommt.

- *Globales Verfahren:*
  Verfahren, das mindestens in zwei Organisationseinheiten verwendet wird.

- *Teilverfahren:*
  Ein Teilverfahren ist logisch eigenständig und wird durch die sequentielle Abfolge von Aktionen realisiert.

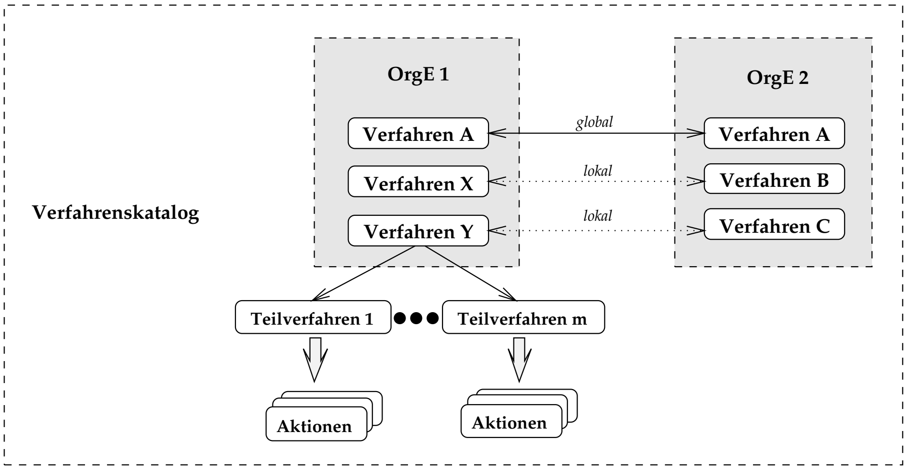

Abbildung 2.4: Schematische Darstellung der Verfahrensebene

- *Werkzeug:*
  Ein technisches System, das gewisse in den Verfahren beschriebene Arbeitsschritte unterstützt.

- *Aktion:*
  Ein durch ein Werkzeug bedingter Arbeitsschritt innerhalb eines Verfahrens.

- *Akteur:*
  Person, die eine Aktion ausführt.

- *Verfahrensfeld:*
  Zusammenfassung aller Verfahren und Teilverfahren, die logisch und inhaltlich zusammengehören.

- *Verfahrenskatalog:*
  Zusammenstellung aller Verfahren und der ihnen zugeordneten Aktionen, die beim Netzbetreiber eingesetzt werden. Jedes Verfahren muß dabei einer Aufgabe zugeordnet sein.

2.5 Ziele des Rahmenbetriebskonzepts

Die bereits angedeuteten Ziele des Netzbetreibers werden im folgenden präzisiert und gemäß dem Rahmenbetriebskonzept (Abbildung 2.1) klassifiziert.

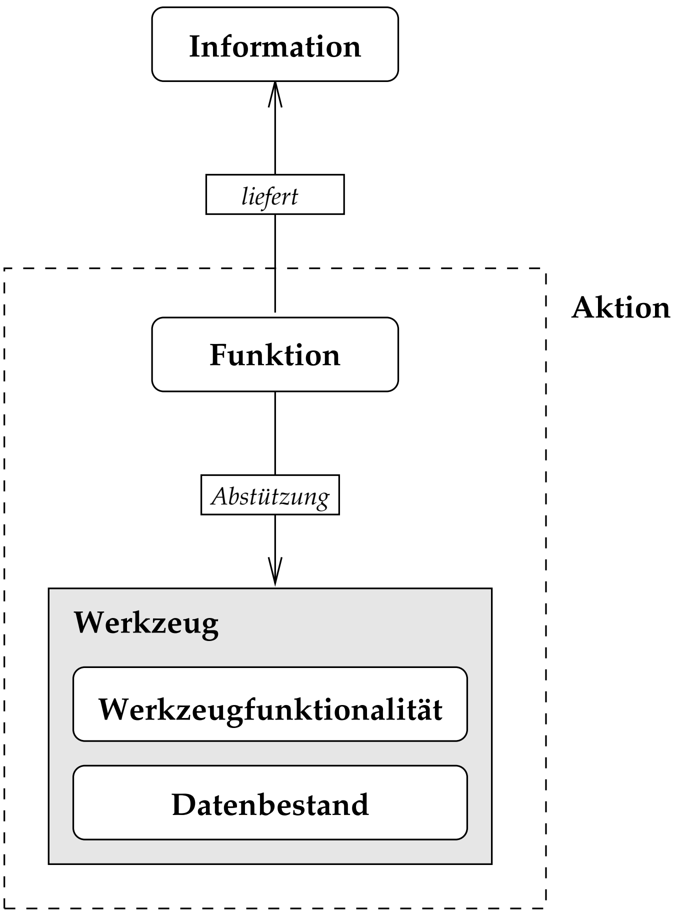

Abbildung 2.5: Präzisierung einer Aktion

2.5.1 Ebenenübergreifende Ziele

Die Vorteile, die ein Netzbetreiber aus einer Darstellung der Organisation im Rahmenbetriebskonzept erhält, sind folgende:

- Offenlegung von Aufwand und Kosten, dadurch Stärkung der Erfolgskontrolle

- Gewährleistung der Abrechenbarkeit von Diensten unter Einbeziehung aller „abrechnungswürdigen" Ressourcenverbräuche der Aufgabe-, Verfahrens- und Werkzeugebene (Stichwort: Accounting)

- Verbesserte Personalplanung durch Aufstellen von Qualifikationsanforderungen an spezielle Rollen und dadurch Wahrung von Know-how

- Richtlinien für ein verbessertes Qualitätsmanagement auf allen Ebenen und Stärkung des Qualitätbewußtseins gegenüber den Kunden

- Ökonomischer Einsatz von Ressourcen entsprechend den Anforderungen der Kunden

- Aufschlüsse über die Anordnungsflexibilität von Ressourcen (Stichwort: internes oder externes Outsourcing)

- Definition von Schnittstellen und Informationsflüssen zwischen verschiedenen Organisationseinheiten, die an einer gemeinsamen Erbringung von Diensten beteiligt sind

- Festlegung eindeutiger Richtlinien in Bezug auf Ansprechpartner, Zuständigkeiten und Informationsflüssen zwischen allen Beteiligten

- Effizienter Ressourceneinsatz durch Definition von Diensten und deren Dienstqualität dem Kunden gegenüber, bedingt durch eine verbesserte Kapazitätsplanung

2.5.2 Dienstbezogene Ziele

Ziele des Dienstleistungskonzepts sind:

- Bereitstellung eines Leistungsangebots an die Kunden:
  Anhand des Dienstkatalogs kann dem Kunden ein umfassendes Angebot der Leistungen unterbreitet werden.

- Darstellung von Kunden/Lieferanten-Beziehungen:
  Die Kunden/Lieferantenstruktur soll genau aufgezeigt werden. Durch diese Darstellung ist es z.B. möglich, Aussagen über die Effektivität der eingesetzten Ressourcen zu treffen.

- Transparenz bzgl. Leistungen und Preisen von Diensten:
  Für eine Organisationseinheit ist es wichtig, die Leistungen mit den anfallenden Kosten in Relation zu setzen. Durch die Bestrebungen der Unternehmen, die anfallenden Kosten dem Kostenverursacher (Kostenträger) eindeutig zuzuordnen („Accounting" = Abrechnung von Leistungen), fällt die Definition der Kostenträger ebenfalls in diesen Bereich.

- Präzisierung der Ausprägung von Diensten:
  Der Netzbetreiber muß in Zusammenarbeit mit dem Kunden die Qualität der angebotenen Dienste definieren. Die Einhaltung dieser Dienstqualität wird vom Dienstnutzer und Dienstleister permanent überwacht.

- Leistungsnachweis:
  Derzeit ist vielen Netzbetreibern unklar, welche Leistungen auf welchem Qualitätsniveau von ihnen angeboten werden. Hinter demselben Begriff unterschiedlicher Organisationseinheiten können verschiedene Leistungsumfänge stehen.

2.5.3 Aufgabenbezogene Ziele

Das Rahmenbetriebskonzept verfolgt ein großes Ziel: die reibungslose Abwicklung des Supports. Die im Hintergrund stehende Frage ist: Wie wird die Zuständigkeit geregelt? Diese noch stark betriebswirtschaftlich geprägte Fragestellung wird durch die bestehende Organisationsstruktur bestimmt. Wird das Ziel von der technischen Seite her betrachtet, ist z.B. die Qualifikation eines Mitarbeiters und die Qualität der Aufgabe genau zu definieren.

2.5.4 Verfahrens- und werkzeugorientierte Ziele

Diese Ziele sind technisch geprägt.

- Stukturierung und Optimierung von Abläufen/Verfahren

- Kurze und garantierte Reaktionszeit im Störfall:
  Abhängig von der Dauer des Ausfalls einer Komponente im Netz entstehen dem Unternehmen zusätzliche Kosten. Für diesen Fall müssen gewisse Informationsflüsse der Verfahren bereits standardisiert vorliegen, um die Reaktionszeit und damit auch die Ausfallzeit zu minimieren.

- Darstellung und effiziente Unterstützung des Informationsflusses:
  Anhand der Darstellung des Ist-Zustandes kann von den Mitarbeitern der Informationsfluß zwischen den Organisationseinheiten erkannt werden und somit Verzögerungen in der Ausführung von Verfahren und Aufgaben vermieden werden.

- Einsatz von Werkzeugen zur Überwachung der Dienstgüte:
  In diesem Fall sollen Werkzeuge eingesetzt werden, die durch Definition von Meßpunkten und Schwellwerten die Qualitätsgüte überwachen.

- Automatisierung von Abläufen durch gezielten Werkzeugeinsatz:
  Der Netzbetreiber versucht, Abläufe zu automatisieren, um anschließend diese „Standardabläufe" als Funktionalitätsanforderungen an ein Werkzeug zu stellen.

- Konfiguration von Werkzeugen:
  Die von einem Werkzeug angebotenen Möglichkeiten, sollen optimal eingesetzt werden, wobei eine sinnvolle und einheitliche Konfiguration Grundvoraussetzung ist.

- Folgen einer Werkzeugmigration:
  Häufig werden Werkzeuge durch leistungsfähigere ersetzt. Beim Einsatz des neuen Werkzeugs sind die bestehenden Randbedingungen zu beachten.

- Durchgängiges Konzept für den Datenschutz und die Datensicherheit:
  Dieser Aspekt wird im verstärkten Maße für die Unternehmen immer wichtiger, da auch sensible Daten über das Corporate Network verschickt werden. Die Unversehrtheit und Vertraulichkeit dieser Daten ist zu garantieren.

2.6 Das Rahmenbetriebskonzept aus objektorientierter Sicht

Das Rahmenbetriebskonzept ist ein Instrument, das den Unternehmen ein Mittel in die Hand gibt, um ein Betriebskonzept für ihre Unternehmensnetze (Corporate Networks) zu erstellen. Ziel ist es, ein Werkzeug zu entwickeln, das die Erstellung des Betriebskonzepts unterstützt.

In der Softwareentwicklung existieren derzeit verschiedene Ansätze zum Entwurf großer Softwaresysteme. Für diese Art von Problemstellung bietet sich ein objektorientierter Ansatz an, da anhand der bereits vorhandenen Struktur des Rahmenbetriebskonzepts eine Identifikation der Objekte bereits eindeutig möglich ist; diese sind z.B. Dienst, Ausführungs-Aufgabe, Verfahren und Werkzeug. Die Beziehungen zwischen diesen Objekten sind ebenfalls vorgegeben, so z.B. Dienst *wird erbracht durch* Aufgabe oder Verfahren *stützt sich ab auf* Werkzeug.

Ziel dieses Abschnitts ist es, die Struktur des Rahmenbetriebskonzepts in objektorientierte Modelle abzubilden und in einer an [RBP+91] angelehnten graphischen Notation darzustellen.

2.6.1 Vorteile eines objektorientierten Ansatzes

Die Beschreibungsmethodik des Rahmenbetriebskonzepts veranschaulicht die Beziehungen des betriebswirtschaftlichen und des technischen Standpunkts.

Organisationseinheiten werden in Relation zu erbrachten Diensten und Aufgaben gesetzt. In der technische Ebene wird die konkrete Umsetzung von Aufgaben in Verfahren, Werkzeuge und Aktionen modelliert. Neben statischen Elementen, z.B. der Organisationsstruktur, existieren auch dynamische Vorgänge, wie die Anwendung bestimmter Verfahren in Abhängigkeit eines ausgesuchten Werkzeugs. Ein objektorientierter Ansatz stellt durch die Konstruktion von Klassen eine statische Beschreibungsstruktur zur Verfügung, die mit dem Rahmenbetriebskonzept vergleichbar ist. Die Instantiierung des objektorientierten Modells ist der Abbildung der realen Welt in das Rahmenbetriebskonzept gleichzusetzen.

Durch die Zerlegung des Rahmenbetriebskonzepts in einzelne Objekte erhält man syntaktisch klar getrennte Einheiten mit explizit beschriebenen Schnittstellen. Die strukturierten Elemente einer Rahmenbetriebskonzept-Ebene können losgelöst von den anderen Ebenen betrachtet und verstanden werden (modulare Verständlichkeit; vgl. [Bro92]).

2.6.2 Phasen zur Erstellung eines objektorientierten Ansatzes

Der objektorientierte Ansatz stellt das Objektdesign vor das Algorithmusdesign.

Analyse des zu lösenden Problems: In dieser ersten Phase setzt sich der Analysator eingehend mit dem Problem auseinander. Um eine vollständige Problembeschreibung der komplexen Vorgänge zu erhalten, ist ein enge Zusammenarbeit zwischen Analysator und Auftraggeber notwendig. Die Analyse gibt nur vor, *was* das neue System dem Anwender bieten soll, liefert also als Ergebnis einen Anforderungskatalog über den Leistungsumfang des zu erstellenden Systems. Auf die Frage der konkrekten Implementierung wird nicht eingegangen.

Systemdesign: In dieser Stufe werden High-Level-Entscheidungen über das Design bzw. Architektur des Systems getroffen. Die entworfene Architektur ist noch grob und nicht auf konkrete Softwareprodukte ausgerichtet. Im Systemdesign wird das Zielsystem in Untersysteme (Module) aufgeteilt, die sich einerseits auf die durch die Analyse gewonnenen Strukturen stützen und andererseits auf der angestrebten Architektur basieren. Zusätzlich muß entschieden werden, welche Performanceaspekte in welchen Umfang zu berücksichtigen sind, bzw. wo Optimierungsmöglichkeiten bestehen und mit welcher Strategie das Problem anzugehen ist. In diesem Stadium wird z.B. schon das Kommunikationsprotokoll oder die Speicherstrategie festgelegt.

Objektdesign: Im Objektdesign wird ein Modell generiert, das auf dem Analysemodell beruht; dies weist aber bereits technische und implementierungsrelevante Merkmale auf (z.B. Datenstrukturen, Algorithmen). Die im Systemdesign entwickelten Strategien müssen beachtet werden.

Implementierung: In dieser Phase werden die durch das Design festgelegten Objektklassen und ihre Beziehungen in eine konkrete Programmiersprache umgesetzt. Dieser Vorgang sollte bereits größtenteils „mechanisch" ablaufen, da alle wichtigen Fragen bereits in den einzelnen Designphasen behandelt wurden. Das Design darf sich daher nicht an speziellen Eigenschaften einer Programmiersprache orientieren, um die Unabhängigkeit des Designs zu gewährleisten.

Die Analyse des Problems wurde durch das Rahmenbetriebskonzept bereits ausreichend bearbeitet, jedoch fehlte bisher eine formale Spezifikation der Struktur. Die Umsetzung in das Softwaresystem wird nach [RBP+91] durch drei verschiedene Modelle beschrieben.

1. *object model:*
   In diesem Modell wird die statische Struktur der einzelnen Objekte festgelegt. Dies kann durch eine graphische Notation geschehen, wobei die Knoten des Graphen die Objektklassen und die Kanten die Beziehungen zwischen diesen Objektklassen darstellen. Eine solche graphische Notation nennt man Objektdiagramm (object diagram).

2. *dynamic model:*
   Dieses Modell beschreibt Veränderungen, die zur Laufzeit entstehen. Die kontrollierenden Aspekte werden mit in das System aufgenommen und die Veränderungen durch ein Statusdiagramm (state diagram) veranschaulicht. In diesem Graphen stellen die Knoten verschiedene Zustände und die Kanten Transaktionen dar, welche durch Ereignisse (events) ausgelöst werden.

3. *functional model:*
   Die Aufgabe dieses Modells besteht darin, die Transformationen eines Wertes anzuzeigen. Dazu werden Datenflußdiagramme (data flow diagrams) benützt, wobei die Knoten Prozessen und die Kanten Datenströmen entsprechen.

2.6.3 Objektorientierter Ansatz für das Rahmenbetriebskonzept

Die in Abschnitt 2.6.2 vorgestellten Phasen sind eigentlich zum Entwurf von objektorientierter Software gedacht, aber das Rahmenbetriebskonzept läßt sich durch das Erstellen eines mit dem „object model“ vergleichbaren Modell sehr gut erfassen und darstellen. Die Objekte, die das Modell enthalten muß, sind durch das Rahmenbetriebskonzept bereits vorgegeben, die Beziehungen zwischen den Objekten können anhand der gewählten graphischen Notation verdeutlicht werden. Das Gewinnen von Beziehungstypen (vergleichbar zu den Objektklassen eines objektorientierten Ansatzes) ist ein wichtiger Faktor, da sie zur Darstellung von Views (Sichtweisen) auf die jeweilige Betreiberorganisation genutzt werden können.

Der große Vorteil eines Modells besteht darin, daß eine überschaubare Struktur erkennbar wird. Da zu diesem Zeitpunkt keine konkreten Instantiierungen vorgenommen wurden, reduziert sich der Umfang der betrachteten Daten erheblich.

2.6.4 Objekte der einzelnen Ebenen

Innerhalb der vier Ebenen existieren verschiedene Objekte, die im Anschluß kurz vorgestellt werden.

Objekte in der Dienstebene: Die Dienstebene legt die Schnittstelle zwischen einem Dienstleister und einem Dienstnutzer fest. Daher haben diese Objekte einen stark betriebswirtschaftlichen Charakter, ein Zeichen dafür ist die Abrechenbarkeit eines Dienstes. Im einzelnen existieren in der Dienstebene folgende Objekte:

- Dienst
- Dienstpaket
- Dienstkatalog

Objekte in der Aufgabenebene: Die Aufgabenebene ist ebenfalls noch sehr stark von betriebswirtschaftlichen Aspekten geprägt, obwohl der Grad, im Vergleich zur Dienstebene, schon abgenommen hat. In der Aufgabenebene wird beschrieben, wie ein Dienst erbracht wird. Zu diesem Zweck existieren verschiedene Objekte:

- Basisaufgabe
- Steuernde Aufgabe
- Ausführungs-Aufgabe

- Aufgabenfeld
- Aufgabenkatalog

Objekte in der Verfahrensebene: Zwischen der Aufgaben- und Verfahrensebene liegt die Schnittstelle der betriebswirtschaftlich orientierten Ebenen (Dienst- und Aufgabenebene) und den technisch geprägten Ebenen. In dieser Ebene wird eine genaue Vorgehensweise vorgegeben, wie eine Aufgabe unter Zuhilfenahme eines Verfahrens erfüllt werden kann.

- Verfahren
- Teilverfahren
- Aktion
- Verfahrensfeld
- Verfahrenskatalog

Objekte innerhalb der Werkzeugebene: In der Werkzeugebene werden die in einer Organisationseinheit eingesetzten Werkzeuge und die von ihnen bereitgestellte Funktionalität behandelt. Dementsprechend gibt es auch dort nur zwei verschiedene Objekte:

- Werkzeug
- Funktion

2.6.5 Einteilung der Objekte in Templates

Wie man leicht erkennen kann, ist die Struktur der einzelnen Ebenen nahezu identisch. Es liegt nahe, die einzelnen Objekte in polymorphe Templates einzuteilen. Der Autor versteht in diesem Zusammenhang unter dem Begriff polymorphe Templates eine Sammlung von Objektklassen, die alle dieselbe Struktur haben und sich nur in den verwendeten Datentypen unterscheiden. Es lassen sich vier dieser Templates erkennen:

1. Katalog-Template:
   In diese Templates lassen sich die Kataloge der einzelnen Ebenen einordnen, also die Objektklassen Dienstkatalog, Aufgabenkatalog und Verfahrenskatalog. Alle drei Klassen sind eine Zusammenfassung von Objektklassen der jeweiligen Ebene.

2. Paket/Feld-Template:
   In dieses Template lassen sich die drei Klassen Dienstpaket, Aufgabenfeld und Verfahrensfeld einordnen, die ihrerseits wieder Zusammenfassungen der logisch zusammengehörigen Elemente einer Ebene sind.

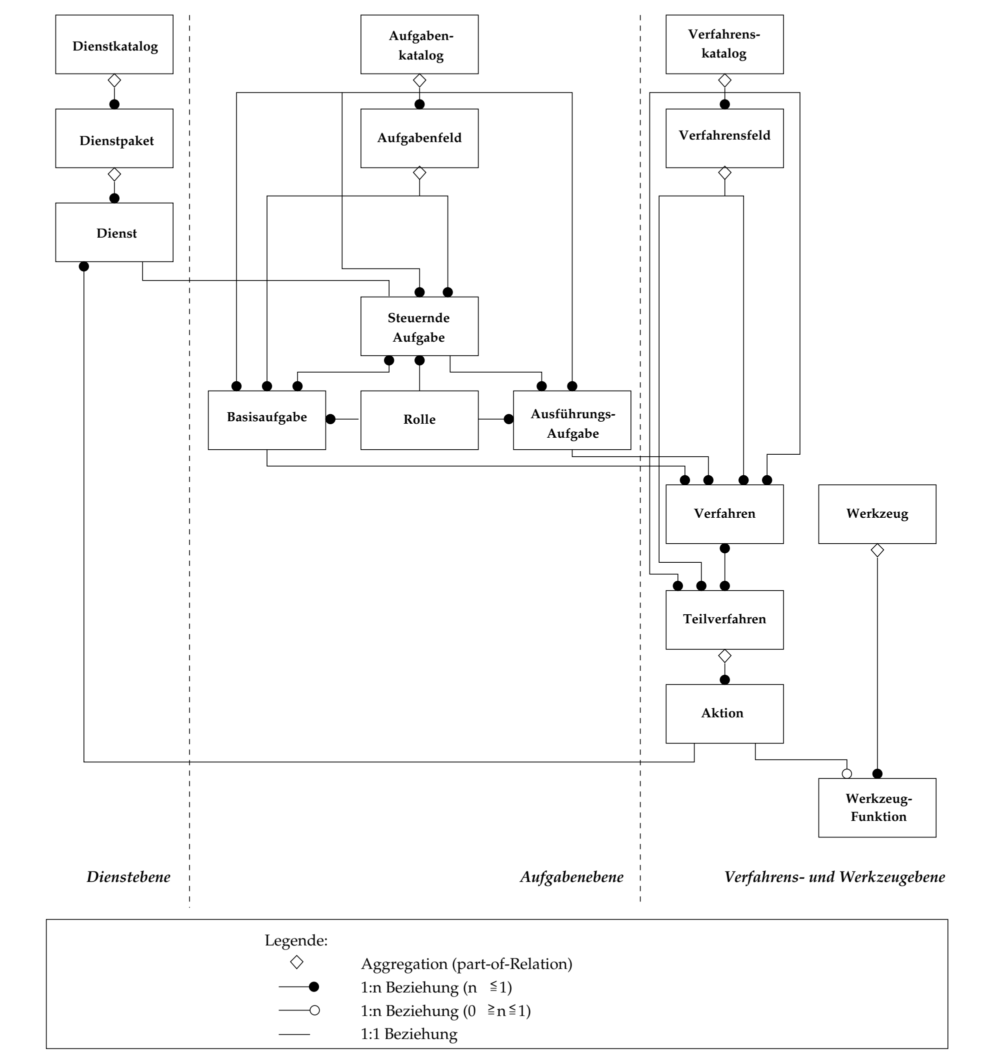

Abbildung 2.6: Struktur des RBKs aus oo-Sicht (Klassendiagramm)

3. Template der steuernden und regelnden Objektklassen:
   Diese beschreiben Objekte, die steuernde bzw. regelnde Aspekte innerhalb des Rahmenbetriebskonzepts behandeln. Dazu gehören die Steuernden Aufgaben, die Verfahren und die Teilverfahren.

4. Template der Ausführenden Objekte:
   In dieses Template fallen die restlichen Objektklassen der ersten drei Ebenen, also die Dienste, Ausführungs-Aufgaben, Basisaufgaben und die Aktionen.

5. Werkzeug-Template:
   In der Werkzeugebene läßt sich keine zu den drei oberen Ebenen des Rahmenbetriebskonzepts kongruente Struktur finden, weshalb ein eigenes Template dafür geschaffen wurde. Dieses Template umfaßt die Objektklasse der Werkzeuge, sowie der (Werkzeug-)Funktionen.

Diese Einteilung ist sinnvoll, da sich dadurch leichter Vererbungsmechanismen erkennen lassen, da die Struktur der Ebenen untereinander ähnlich ist. Ausgenommen ist die Werkzeugebene, deren Einbindung aber trivial erscheint.

Kapitel 3

Motivation für ein Werkzeug

Viele Unternehmen versuchen derzeit, ihre Organisation und ihre Produktion zu straffen. Begründet wird dieses Verhalten durch die angespannte wirtschaftliche Gesamtsituation, die durch die anhaltende Rezession verursacht wurde. Um wirtschaftlich konkurrenzfähig zu bleiben, sind die Unternehmen gezwungen, Rationalisierungsmaßnahmen durchzuführen. In den meisten Unternehmen fehlt aber eine Dokumentation des Ist-Zustandes der Arbeitsabläufe, die aber Grundvoraussetzung für eine Analyse wäre.

Darüber hinaus versuchen viele Unternehmen, das „Qualitätssiegel" ISO 9000 (vgl. [ISO87b]) zu erhalten. ISO 9000 ist eine Gruppe von Normen (ISO 9000 bis ISO 9004), die die Sicherung der Qualität von Arbeitsabläufen innerhalb eines Unternehmens regelt; eine Analyse des Ist-Zustandes ist zwingend erforderlich. Durch die Festlegung der Toleranzen kann dem Kunden gegenüber ein präziseres Angebot unterbreitet werden, wodurch eine verbesserte Marktakzeptanz erreicht wird.

Als Konsequenz daraus ergibt sich für einen Netzbetreiber die Notwendigkeit, ein Betriebskonzept zu entwickeln, da momentan

- die Kosten betriebswirtschaftlich nicht eindeutig auf die Kostenverursacher (Kostenträger) zugeordnet werden können. In Zukunft wird dieser Aspekt, das Accounting, immer weiter an Gewicht innerhalb eines Unternehmens zunehmen.
- Ressourcen (Personal, Komponenten, Werkzeuge, etc.) ineffizient eingesetzt werden.

Das Rahmenbetriebskonzept, dessen Grundzüge in Kapitel 2 dargestellt wurden, liefert den Ansatz zur Erfüllung dieser Forderungen.

Die Situation der Corporate Networks in allen größeren Betrieben ist ähnlich. Die gestellten Anforderungen an die Netzmanagement-Werkzeuge sind intuitiv definiert worden und darum nicht umfassend, begründet im Fehlen eines durchgängigen Betriebskonzepts für das Corporate Network. Um diesen Mißstand zu beseitigen, wurde das Rahmenbetriebskonzept entwickelt, das die Erstellung eines auf die individuellen Gegebenheiten eines Netzbetreibers angepaßten Betriebskonzepts erst ermöglichen, bzw. in einem späteren Stadium erleichtern soll. Ziel ist die Einteilung nach exakt festgelegten Kriterien, wie Dienst, Aufgabe, Verfahren oder Werkzeug.

Die Motivation für die Entwicklung eines Werkzeugs liegt in der Komplexität der gewonnen Daten durch die Anwendung des Rahmenbetriebskonzepts. Die hochdynamischen Effekte bei Änderungen innerhalb des Datenbestands werden durch die vermaschte Struktur der einzelnen Objekte hervorgerufen. Es gibt eine Vielzahl von Beziehungen zwischen den Objekten einer Ebene, bzw. zwischen Objekten verschiedener Ebenen, die nicht sofort erkennbar sind. Die Hauptaufgabe eines Werkzeugs besteht in der Arbeitserleichterung für den Benutzer. Es soll den Nutzer bei der Erstellung eines korrekt instantiierten Modells, dem Betriebskonzept, unterstützen. Die gewonnene Dokumentation des Ist-Zustandes der Betreiberorganisation ermöglicht die Durchführung von Analysen für diese Betreiberorganisation. Das Hauptaugenmerk dieser Arbeit wird auf der Unterstützung des Benutzers bei der korrekten Erfassung von Objekten, wie Dienst, Aufgabe, Verfahren und Werkzeug, liegen.

3.1 Motivation des Netzbetreibers zur Erstellung eines Betriebskonzepts

Wie bereits erwähnt, ist es einem Netzbetreiber derzeit nicht möglich, die Anforderungen, die er an seine Netzmanagement-Werkzeuge stellt, detailliert zu definieren.

Die meisten Großunternehmen sind im Bereich der Datenverarbeitungstechnik in einer Umstrukturierungsphase. Bis vor einigen Jahren wurden Mainframe-basierte Systemarchitekturen als die an die Anforderungen am besten angepaßt Lösung angesehen. Mittlerweile werden verstärkt Workstation-Clusters, die auf einer Client/Server-Architektur basieren, eingesetzt. Meist wird in der Workstation-Umgebung Unix als Betriebssystem verwendet, während in der Mainframewelt MVS, VMS oder VM als Betriebssystem favorisiert werden.

Für die Systemarchitekturen werden unterschiedliche Protokollfamilien eingesetzt, was sich in der heterogenen Struktur wiederspiegelt. Um eine Kommunikation untereinander zu realisieren, wird derzeit TCP/IP auch auf Mainframebasis implementiert ([Hol94] und [Jag94]) und in der Industrie eingesetzt. Bereits hier läßt sich erkennen, daß die Netzbetreiber versuchen, die bisher in proprietären Inseln vorhandenen Netztechnologien weiter zu verwenden. Durch die rasante Entwicklung im Bereich der Datenverarbeitung (Hard- und Software) ist es einem Unternehmen allein schon aus wirtschaftlichen Gründen nicht möglich, immer den aktuellen Stand der Technik zu repräsentieren. Die Investitionen, die in diesem Bereich zu tätigen wären, gehen sehr schnell in zwei- bis dreistellige Millionenbeträge. Es ist daher einsichtig, daß ein Netzbetreiber die bestehenden Technologien nur sukzessive ersetzen kann und somit, begünstigt durch die rasche Entwicklung, gezwungen wird, mehrere Technologien parallel zu benutzen.

Bisher werden in den einzelnen „Kommunikationsinseln" unterschiedliche Werkzeuge zum Netzmanagement eingesetzt, die sog. „Element-Management-Werkzeuge". Diese sind lediglich in der Lage, einen speziellen Gerätetyp eines Herstellers zu managen. Die Funktionalität orientiert sich an den technischen Gegebenheiten, die aus der Bottom-Up-Sicht des Herstellers gewonnen werden; ein Netzbetreiber hat-

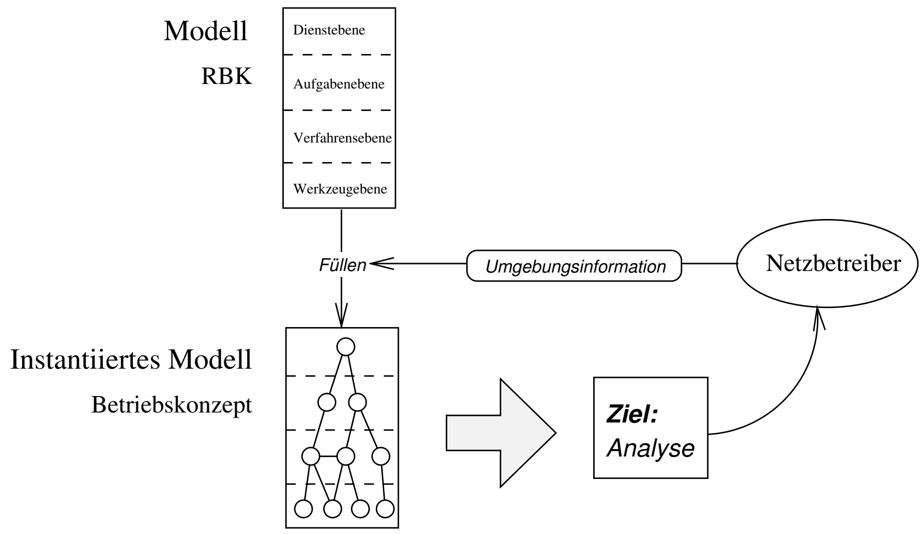

Abbildung 3.1: Motivation eines Netzbetreibers für ein Werkzeug

te keinerlei Einfluß auf die bereitgestellte Funktionalität. Aufgrund der fehlenden Normung der Managementagenten in den Komponenten lassen sich diese nur durch das Element-Management-Werkzeug des Herstellers überwachen. Die Bestrebungen der Netzbetreiber, ein integriertes Netz- und Systemmanagement zu etablieren, d.h. die bisher vorhandenen unabhängigen Teillösungen zusammenzuführen und zu einem einheitlichen Netz- und Systemmanagement zusammenzufassen, setzt die Sammlung eines gemeinsamen Anforderungskatalogs voraus.

Das Rahmenbetriebskonzept deckt diesen Teilaspekt mit ab. In verschiedenen Projekten ([Ber95, Ege94, Hab94, vdH94, Wei94]) wird derzeit, das Rahmenbetriebskonzept mit konkreten Daten gefüllt. Da die Netzumgebung abhängig vom Netzbetreiber ist und daher stark differiert, ist die Zusammenarbeit mit dem jeweiligen

Betreiber zwingend erforderlich. Ist das Rahmenbetriebskonzept mit den Daten der jeweiligen Betreiberorganisation gefüllt, erhält man ein Betriebskonzept, das den Ist-Zustand der jeweiligen Betreiberorganisation repräsentiert. Dieses Betriebskonzept dient als Grundlage für die Lokalisierung von Defiziten, d.h. das herausragende Ziel liegt in der Analyse des Ist-Zustandes. Der Vergleich des Ist-Zustandes mit dem angestrebten Soll-Zustand, der vom Netzbetreiber zu spezifizieren ist, ermöglicht es, Konsequenzen für die Organisation zu ziehen (vgl. Abbildung 3.1).

3.2 Phasen der Anwendung des Rahmenbetriebskonzepts

Wie bereits in Abschnitt 3.1 angeklungen, zerfällt die Anwendung des Rahmenbetriebskonzepts in mehrere Phasen; anhand der Abbildung 3.2 lassen sich grundsätzlich drei Phasen unterscheiden.

3.2.1 Erfassungsphase

Die Erfassungsphase des Ist-Zustands ist gleichzusetzen mit dem „Füllen“ des Modells. Es wird also eine Dokumentation der

- Organisationsstruktur,
- Dienstleister/Dienstnutzerstruktur,
- Dienstangebot und
- Arbeitsabläufe, worunter die Aufgaben, Verfahren und die zu deren Realisierung eingesetzten Werkzeuge fallen,

erstellt. Das Erfassen gestaltet sich unterschiedlich schwierig. So können die Objekte selbst (Dienst, Aufgabe, Verfahren etc.) verhältnismäßig problemlos durch Eingabemasken erfaßt werden. Die Beziehungen zwischen den einzelnen Objekten sind detailliert aus dem Rahmenbetriebskonzept zu entnehmen. Es ergeben sich aber Defizite in Bezug auf Daten, die zur Initialisierung eines Verfahrens durch eine Aufgabe, eingesetzt oder von einem Objekt als Resultat zurückgeliefert werden. Diese Daten dürfen nicht vergessen werden, da sie die Steuerung und Regelung eines Ablaufes bestimmen. Bisher werden sie direkt in die Objekte eingetragen. Zu überlegen wäre, ob es nicht sinnvoll ist, diese Daten, die eine Klammer zwischen Objekten bilden, aus diesen herauszuziehen und als Attribute der Beziehung darzustellen.

In den Arbeiten [Ber95, Ege94, Hab94, vdH94, Wei94] wird versucht, eine Betreiberorganisation, die BMW AG, gemäß dem Rahmenbetriebskonzept aus verschiedenen Blickwinkeln heraus zu betrachten.

3.2.2 Bewertungs- und Analysephase

In dieser Phase werden die Daten, die in der Erfassungsphase gewonnen wurden, weiter verarbeitet. Das vom Netzbetreiber verfolgte Ziel ist das Aufspüren von Defiziten der Organisationsstruktur (was im weiteren Verlauf der Arbeit nicht weiter betrachtet wird) oder der effektivere Einsatz von Ressourcen (Komponenten, Personal, etc.). Weitere vom Netzbetreiber verfolgte Ziele:

- Sicherheitsaspekte:
  Die Unternehmen haben ein grundsätzliches Interesse, ihren Sicherheitsstandard zu überprüfen. Durch die Feststellung des Status Quo, ist es den Unternehmen möglich, Sicherheitslücken zu erkennen und geeignete Gegenmaßnahmen einzuleiten.

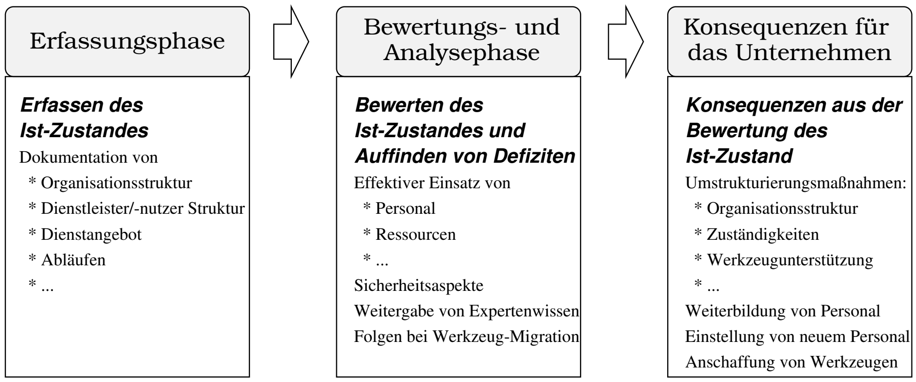

Abbildung 3.2: Anwendungsphasen des Rahmenbetriebskonzepts

- Weitergabe von Expertenwissen (Erfahrungen):
  Derzeit existieren nur sehr eingeschränkte Kenntnisse in Bezug auf Abläufe von Aktionen. Es wird hauptsächlich nur mit „Schlagworten", wie z.B. „Fehlermeldeverfahren", gearbeitet. Von Interesse ist aber die Angabe eines standardisierten Ablaufs (Verfahrens), da dadurch z.B. ein ungelernter Angestellter eine kürzere Einarbeitungszeit hat (z.B. Tutorsystem) und bei anfallenden Fehlern diese schneller behandelt werden können.

- Folgen bei Werkzeugmigration:
  Eine Fragestellung, die für einen Netzbetreiber von großer Bedeutung ist:

  - Welche Funktionalität kann gewonnen werden?

- Welcher organisatorische Aufwand muß aufgebracht werden, um das vorhandene Werkzeug durch ein neues zu ersetzen ?
- Wie hoch ist der finanzielle Aufwand der Migration?
- Welche Organisationseinheiten, Rollen, Aufgaben oder Verfahren greifen auf das zu migrierende Werkzeug zurück?
- Rechtfertigt die gewonnene Funktionalität den Aufwand?

Soll eine Werkzeugmigration durchgeführt werden, ist ein Anforderungskatalog für das zu migrierende Werkzeug zu erstellen. Anschließend sind verschiedene Produkte zu evaluieren und anhand von KO-Kriterien auszuwählen. Ein derartig strukturiertes Vorgehen wird u.a. ausführlich in [Ber95] beschrieben.

3.2.3 Konsequenzen für das Unternehmen

Aus der Bewertung des Ist-Zustandes leiten sich Konsequenzen für das Unternehmen ab, da es einem stetem Wandel unterliegt. Dieser wird einerseits durch den technischen Fortschritt (z.B. neue bessere Komponenten), andererseits betriebswirtschaftlich (Rationalisierung) motiviert. Denkbare Konsequenzen sind:

- Umstrukturierungsmaßnahmen bzgl.
  - Organisationsstruktur
  - Zuständigkeiten
  - Personalstruktur
    - Weiterbildung von Personal
    - Einstellung von neuem Personal mit geeigneter Qualifikation
  - Werkzeuge
    - Anschaffung neuer Werkzeuge
    - Neuartiger Einsatz von vorhandenen Werkzeugen
- Ein gesteigertes Vertrauen der Kunden, da die Qualitätstoleranzen (QoS) präzisiert werden; dies geschieht durch die Dokumentation der Verfahren und den dazugehörigen Aktionen. Als Folge daraus ergibt sich eine höhere Akzeptanz beim Kunden.

3.3 Anwendung des Rahmenbetriebskonzepts am Beispiel des Task-View-Modells

Viele Unternehmen reformieren derzeit ihr Netzmanagement grundlegend. Der Trend geht eindeutig in die Richtung eines integrierten Managements, weg von den isoliert arbeitenden Werkzeugen (Element Managers). Diese Werkzeuge waren nur auf einen Typ von Netzkomponenten einer Herstellerfirma zugeschnitten. Diese Entwicklung ist rückläufig, da die Hersteller auf Druck der Unternehmen gezwungen wurden, ihre Systeme offener zu gestalten.

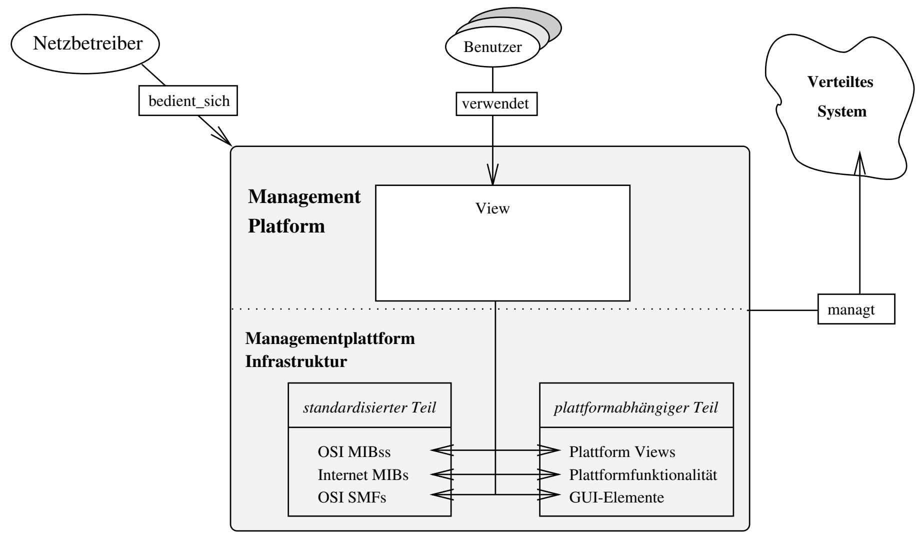

Abbildung 3.3: Status Quo bei Managementplattformen

Die Hersteller bemühen sich derzeit die von den Element Managern bereitgestellte Funktionalität in „integrierte Managementplattformen" zu integrieren. Die Daten, auf die sich die Managementplattform (MPF) abstützt, werden in der *Managementplattform Infrastruktur*-Ebene abgelegt. Innerhalb dieser Ebene existiert

- ein standardisierter Teil, der z.B. OSI MIBs oder Internet MIBs enthält und
- ein plattformabhängiger Teil, der z.B. die Plattformfunktionalität, GUI-Elemente oder auch die Plattform-Views enthält.

Darüber angesiedelt findet sich die eigentliche Managementplattform, welche die in der Management-Infrastruktur abgelegten Daten interpretiert, verarbeitet und an den Benutzer weiterreicht. In dieser Ebene wird auch die Viewbildung vorgenommen, auf die im folgenden etwas genauer eingegangen wird. Unter der Bezeichnung „View" werden die vom Betrachter ausgewählten Daten des Gesamtdatenbestands verstanden; sie bilden ein Mosaiksteinchen zur Anpassung (Customizing) einer Managementplattform an unternehmensrelevante Gegebenheiten. Dieses Customizing wird in Zukunft stark zunehmen und für die Unternehmen an Bedeutung gewinnen.

Die unter Abschnitt 3.2 vorgestellte Thematik wird anschließend durch ein Beispiel, die Task-Views (vgl. [Seg95]) veranschaulicht, die für das Customizing einer Managementplattform eingesetzt werden.

Erfassungsphase: Ist-Zustand beim Einsatz von Managementplattformen

In der Managementplattform-Ebene sind die Benutzer-Views abgelegt. Diese Views sind nicht auf unternehmensspezifische Gegebenheiten abgestimmt, da sie direkt mit der gewählten Plattform ausgeliefert werden und von den Herstellern nur „Default-Views“ (Voreinstellung) angeboten werden (Abbildung 3.3).

Die Bewertungs- und Analysephase: Soll-Zustand beim Einsatz von Managementplattformen

Der Umstand, daß die Plattform nur eine Teilmenge der möglichen Views unterstützt, ist aber meist für die Unternehmen nicht befriedigend, so daß nach Auswegen gesucht wird. Die Netzbetreiber sind bestrebt, nicht nur einen View, der ihnen

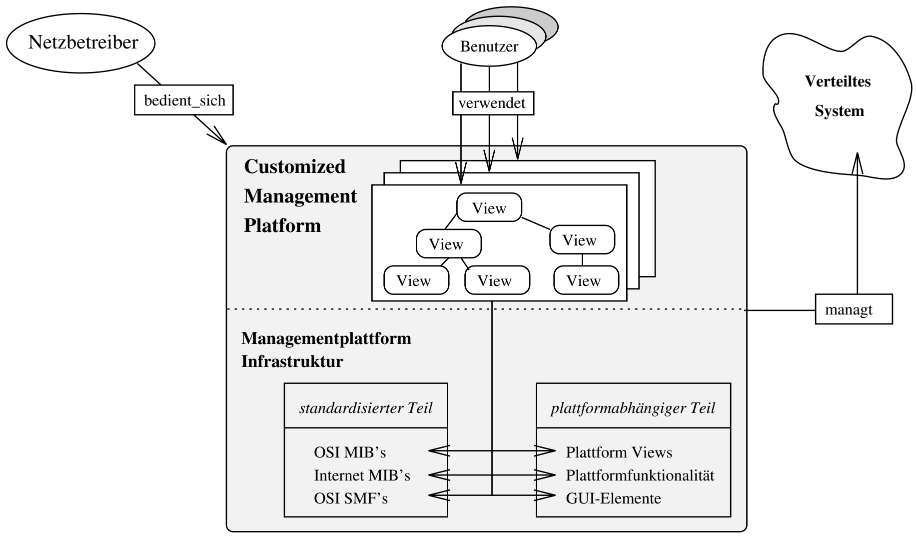

Abbildung 3.4: Angestrebte Lösung: Task-Views

von seiten der Managementplattformhersteller angeboten wird, auf ihr Netz zu haben, sondern eine Menge von Views, die den Anforderungen des jeweiligen Benutzers angepaßt sind. Dadurch wird aus dem starren Instrument der Managementplattform ein flexibles Werkzeug, eine *Customized Management Platform* (vgl. Abbildung 3.4) geschaffen.

Die Konsequenz für das Unternehmen: Das Task-View-Modell

Der zuvor dargestellte Soll-Zustand kann durch das Task-View-Modell (vgl. [Seg95]) modelliert werden. Eine Grundvoraussetzung bilden hierbei die Analyse der anfallenden Aufgaben und die zu deren Bewältigung eingesetzen Verfahren. Die daraus resultierenden Strukturen werden in View-Templates zusammengefaßt und in die Managementplattform abgebildet (vgl. Abbildung 3.5). So kann auf die individuellen Anforderungen eines Benutzers eingegangen werden. Task-Views sind, vereinfacht ausgedrückt, benutzerspezifische Sichtweisen. Sie gehen explizit auf die Bedürfnisse eines Benutzer einer Managementplattform ein und ermöglichen ihm, neben den technischen Views auch andere, speziell auf sein Aufgabengebiet zugeschnittene Views. Task-Views setzen daher auch eine Klassifikation der Benutzerpopulation voraus.

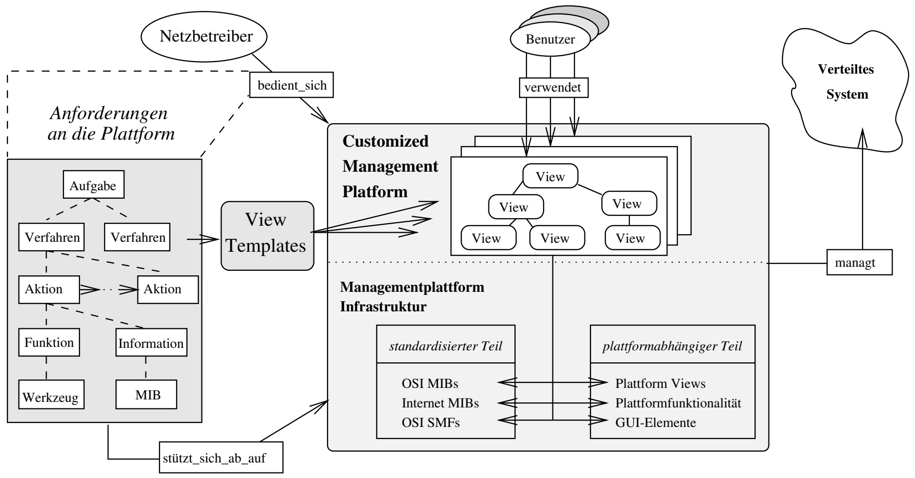

Abbildung 3.5: Task-View Lösungsmodell

Abbildung 3.6 zeigt die Phasen der Anwendung des Rahmenbetriebskonzepts anhand des Beispiels der Task-Views. Ausgehend von dem Ziel, einer sog. *Customized Management Platform*, muß der Netzbetreiber überlegen, welche zusätzlichen Informationen von einem Werkzeug zu erfassen sind. Es ergibt sich ein Feedback vom Ziel zur Erfassungsphase. Die Task-View-relevanten Daten sind nur ein kleiner Ausschnitt aus der Gesamtmenge von Informationen die erfaßt werden. Es handelt sich um Daten der „Technischen Ebene“ (Darstellungselemente, MIBs der Komponenten und Endsysteme, Managementanwendungen) und um Daten, die die Rolle des Benutzer betreffen, d.h. der Fokus liegt auf den spezifischen Objekten und Beziehungen.

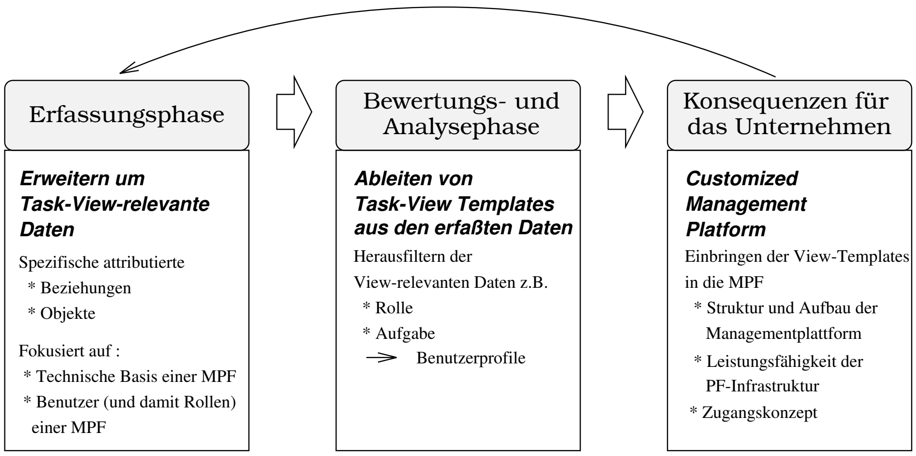

Abbildung 3.6: Anwendung des Rahmenbetriebskonzepts auf Task-Views

In der Bewertungs- und Analysephase werden die view-relevanten Daten herausgefiltert, z.B. die Rolle eines Mitarbeiters oder die von ihm zu erbringenden Aufgaben. Aus den Task-View-Templates lassen sich die gewünschten Daten ableiten.

Die Konsequenz für das Unternehmen ist das Erreichen des Ziels, die angepaßte Managementplattform.

3.4 Abbildung der realen Welt

Da sich die Komplexität der realen Welt nicht in technischen Systemen abbilden läßt, müssen vereinfachende Modelle für die Darstellung verwendet werden. Sie setzen sich aus Formalismen zusammen, die aus syntaktisch klar voneinander getrennten Einheiten aufgebaut sind. Die ersten Probleme ergeben sich bereits bei der Festlegung dieser Einheiten, da ein Modell immer einer dynamischen Entwicklung unterworfen ist. Soll die Abbildung in das Modell geschehen, so sind diese Einheiten

1. zu identifizieren,

2. mit Daten zu füllen und

3. miteinander in Beziehung zu setzen.

Das Rahmenbetriebskonzept liefert bereits ein Gerüst für die Beziehungen und die entsprechenden Objekte. Das instantiierte Rahmenbetriebskonzept erreicht eine so hohe Komplexität, daß eine Werkzeugunterstützung zwingend notwendig ist. Es unterstützt die Erfassung von Daten, durch Vorgabe von Richtlinien, um

- die Objekte eindeutig zu identifizieren und
- sie voneinander abzugrenzen.

3.5 Strukturiertes Vorgehen beim Erfassen von Daten

Die Erfassung von Daten dürfte das schwerwiegendste Problem bei der Anwendung des Rahmenbetriebskonzepts sein. Die einzelnen Objekte müssen so erfaßt werden, daß sie eindeutig einer abgegrenzten Objektklasse zugeordnet werden können. Daher wurde eine Methodik entwickelt, die Richtlinien vorgibt, wie ein Betriebskonzept für einen Netzbetreiber systematisch erstellt werden kann. Die Methodik gliedert sich in mehrere Schritte:

1. *Methodikschritt 1:*
   Bestimmen und Abgrenzen einer Betreiberorganisation, für die ein Betriebskonzept zu erstellen ist.
2. *Methodikschritt 2:*
   Sämtliche innerhalb einer Betreiberorganisation auftretenden Organisationseinheiten sind zu ermitteln.
3. *Methodikschritt 3:*
   Für jede in der Betreiberorganisation auftretende Organisationseinheit sind die von dieser Organisationseinheit angebotenen Dienste in Form von Dienstkatalogen zu beschreiben.
4. *Methodikschritt 4:*
   Zu jedem in den Dienstkatalogen auftretenden Dienst sind die zu dessen Erbringung notwendigen Aufgaben anzugeben.
5. *Methodikschritt 5:*
   Ergänzen der aus den Diensten abgeleiteten Aufgaben, um die in den Organisationseinheiten anfallenden Basisaufgaben.

6. *Methodikschritt 6:*
   Beschreiben der innerhalb der jeweils betrachteten Organisationseinheit genutzten Verfahren in Form einer Abfolge von Aktionen und Zuordnung jedes Verfahrens zu den Aufgaben, bei deren Erbringung das Verfahren verwendet wird.

7. *Methodikschritt 7:*
   Ermitteln der zur Ausführung der Verfahren verwendeten Werkzeugfunktionen.

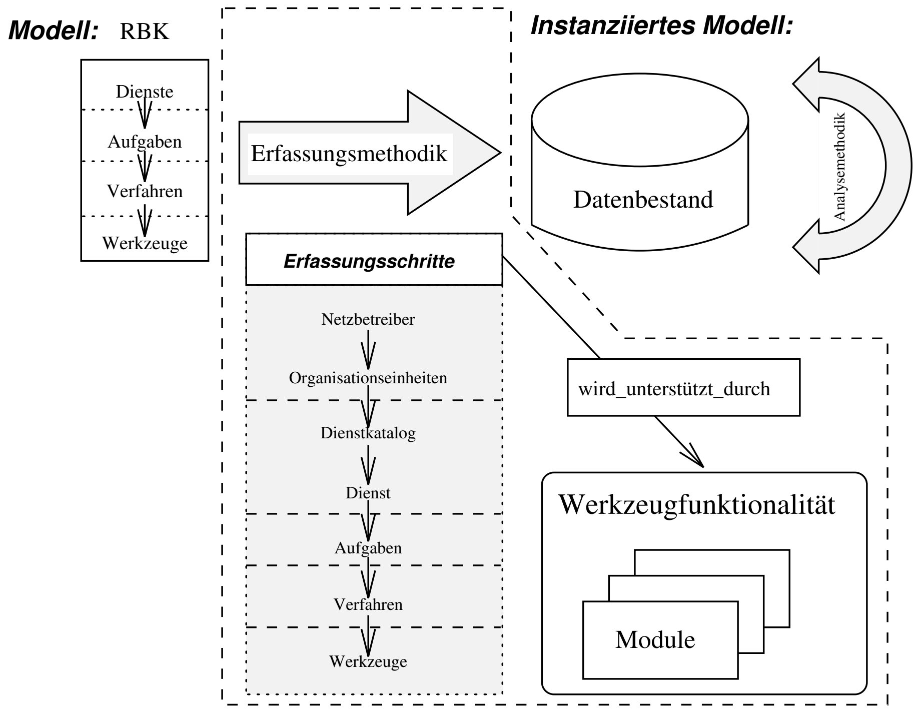

Abbildung 3.7: Einordnung der Erfassungsmethodik in den Gesamtkontext

Eine erste Anwendung dieser Erfassungsmethodik wurde von [Wei94] in Bezug auf die Integration von Datenbeständen vorgenommen. Die Erfassungsmethodik hat ihre Tauglichkeit bewiesen, aber es darf nicht verschwiegen werden, daß sich einige Probleme ergaben, die noch zu lösen sind. So muß schon für den *Methodikschritt 1* eine Richtlinie angegeben werden, die eindeutig angibt, was unter einer Betreiberorganisation zu verstehen ist. Neben dem allgemeinen Rahmen einer „technischen“ Definition des Begriffs der Betreiberorganisation ist durch Gespräche auf die genauen unternehmensspezifischen Details einzugehen. Eine enge Kooperation mit dem jeweiligen Netzbetreiber ist daher unumgänglich. Ähnliche Probleme treten auch in den anderen Methodikschritten auf, wobei die betriebswirtschaftlichen Aspekte immer weiter in den Hintergrund treten und die technischen Gesichtspunkte stark an Gewicht zunehmen. Die Erfassung der technischen Details ist schwierig, da diese Objekte starken Veränderungen unterworfen sind und somit auch nicht die relativ starre Struktur der Organisationseinheiten aufweisen. Durch die Erfassungsmethodik ist gewährleistet, daß alle Objekte Top-Down erfaßt werden.

Kapitel 4

Anforderungen an ein Werkzeug

Die gestellten Anforderungen an ein Werkzeug hängen eng mit den Zielen des Rahmenbetriebskonzepts zusammen. Man muß aber bedenken, daß ein Teil der bereits angesprochenen Ziele betriebswirtschaftlicher Natur sind, die bei weiterer Betrachtung nicht mehr berücksichtigt werden. Im folgenden Verlauf wird daher nur auf die technischen Anforderungen eingegangen; Quellen dafür sind:

- Grundlegende Anforderungen an ein Werkzeug, die vom Benutzer direkt verlangt werden und
- Anforderungen, die aus dem Modell hervorgehen.

4.1 Grundlegende Anforderungen an ein Werkzeug

Direkte und auch sofort einsichtige Anforderungen an das Werkzeug sind:

- Verteiltes Arbeiten:
  Mehrere Benutzer sollen gleichzeitig auf verschiedenen Datenbeständen mit dem Werkzeug arbeiten können. Arbeiten mehrere Benutzer auf dem gleichen Datenbestand, so sind Änderungen des einen Benutzers den anderen Benutzern sofort mitzuteilen.

  Eine Lösung für diese Probleme stellt eine Client/Server-Architektur dar, die eine Möglichkeit für die Realisierung eines verteilten Systems ist.

- Konsistenz der Daten:
  Durch diese verteilte Verarbeitung, muß auch die Konsistenz der bearbeiteten Datenbestände gewährleistet werden. Seiteneffekte bei der Bearbeitung eines Datenbestands auf einen anderen Datenbestand sind zu vermeiden.

- Arbeiten auf einen Datenbestand:
  Das Werkzeug muß offen sein gegenüber Änderungen in der Datenbasis bzgl.:
  - Eintragen neuer Informationen
  - Löschen von Informationen
  - Ändern von Informationen
- Umgang mit Expertenwissen:
  Das bisher nicht in aufgeschriebener Form vorhandene Expertenwissen soll strukturiert abgelegt werden, um es auch Mitarbeitern, z.B. anderer Organisationseinheiten, zugänglich zu machen sowie auf seine Richtigkeit hin zu überprüfen (Verifikation).
- Sichtweisen:
  Durch den Einsatz von Expertenwissen und die komplexe Struktur des Rahmenbetriebskonzepts ist es nicht zumutbar, dem Benutzer nur einen „Gesamtview“ anzubieten. Vielmehr muß auf die Bedürfnisse von Benutzergruppen eingegangen werden, die mit individuellen Views ihrer jeweiligen Aufgabe entsprechend ausgestattet werden. Durch die gesteigerte Übersichtlichkeit wird Arbeitszeit eingespart.
- Navigation:
  Es muß dem Benutzer eine gewisse Freiheit in der Navigation innerhalb des ihm zugänglichen Datenbestands gewährt werden. Zu diesem Zweck soll es dem Benutzer möglich sein, sich einen Überblick über die Zusammenhänge der einzelnen Objekte seines Datenbestands zu schaffen, d.h. es wird eine Visualisierung der jeweiligen Betrachtungsweise vorgenommen.

4.2 Anforderungen aus dem Modell

Neben den grundlegenden Anforderungen aus Abschnitt 4.1 können auch Anforderungen aus dem zugrundeliegenden Modell, dem Rahmenbetriebskonzept, gewonnen werden.

Die notwendige Typisierung von Objekten und Beziehungen ergibt sich direkt aus dem Aufbau des Rahmenbetriebskonzepts, da die Unterscheidung der Objekte der einzelnen Ebenen möglich sein muß. Weiterhin existieren innerhalb jeder Ebene verschiedene Objekttypen, wie man leicht anhand des ausgewählten Beispiels in Abbildung 4.1 erkennen kann. Neben der Einteilung von Objekten in Typen hat auch eine Einteilung von unterschiedlichen Beziehungen in Typklassen zu geschehen. Analog zu den zwischen den Objekten einer Ebene existierenden Beziehungen, existieren auch ebenenübergreifende Beziehungen.

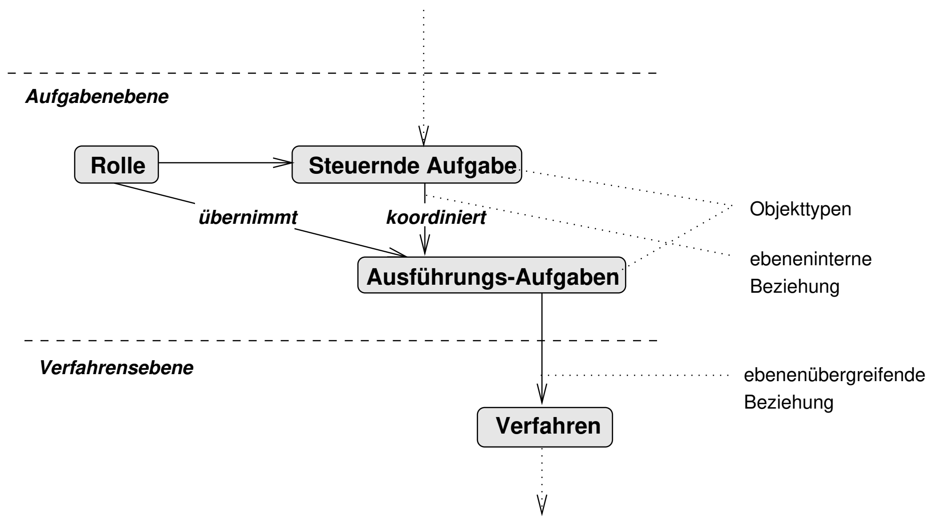

Abbildung 4.1: Objekt- und Beziehungstypen

4.3 Erstellung von Benutzerprofilen

Vor der Erstellung eines Konzepts für ein Werkzeug sollte der Entwickler überlegen, welches Ziel mit dem Werkzeug verfolgt werden soll und an welche Benutzergruppe es sich wendet. Abhängig von dieser grundlegenden Überlegung sind die Anforderungen an das Werkzeug zu gestalten. Im folgenden werden einige Benutzerprofile vorgestellt.

4.3.1 Das Benutzerprofil des „Erfassers“

Als erstes werden die Aufgaben, die ein Erfasser zu erledigen hat, genauer spezifiziert, um davon ausgehend einige Szenarien zu entwerfen, in denen zutage treten wird, welche Anforderungen von Seiten des Erfassers an das die Erfassungsmethodik unterstützende Werkzeug gestellt werden.

Aufgaben des Erfassers:

- Sammeln von Daten
- Strukturieren der gesammelten Daten
- Eingabe der Daten

- Manipulation der Daten
  - Verändern der Datenstruktur (z.B. Hinzufügen neuer Attribute)
  - Verändern von Attributwerten
- Pflegen des Datenbestands
  - Gewährleistung von Datenschutz und Datensicherheit
  - Aktualisierung des Datenbestandes

Aus den Aufgaben, die der Erfasser zu bewältigen hat, sind die Views des Erfassers zu definieren.

Um Redundanzen zu vermeiden, ist dem Erfasser ein Überblick über den Gesamtdatenbestand zu gewähren, da sonst z.B. semantisch identische Teilverfahren mit verschiedenen Namen im Datenbestand entstehen können und so das Prinzip der modularen Komposition (Zusammensetzung) durchbrochen wäre.

Anforderungen des Erfassers

- Schnittstelle:
  Ein Großteil der Daten ist bereits in unterschiedlichen Formaten an verschiedenen Lokalitäten innerhalb des Unternehmens verfügbar. Daher ist es aus mehreren Gründen, die im Anschluß aufgeführt werden, sinnvoll, eine Integration dieser Daten vorzunehmen.
  - Zeitersparnis, da eine Neueingabe der Daten entfällt
  - Vermeidung von Fehlern bei der Eingabe von kryptischen Texten und
  - Erhaltung der Datenkonsistenz im bereits vorhandenem Rahmen

  Durch die Integration von Datenbeständen entstehen zusätzliche Probleme, da oftmals semantisch identische Daten in verschiedenen Formaten abgelegt sind (Redundanz). Bei unterschiedlicher Interpretation dieser Daten werden im Extremfall konträre Ergebnisse erzielt.
- Maske für eine Erweiterung der Datenbasis:
  Dem Erfasser wird eine Eingabemaske zur Verfügung gestellt, mit deren Hilfe er neue Datensätze in die Datenbasis eintragen und bestehende Datensätze editieren kann.
- Vererbung:
  Durch den in Abschnitt 2.6 dargestellten objektorientierten Ansatz wird sichergestellt, daß Vererbungen vorgenommen werden können. Die Vererbung

wird eingesetzt, um neue Objektklassen aus bereits bestehenden Klassen abzuleiten und so eine Spezialisierung vorzunehmen. Die abgeleitete Klasse dabei erbt alle Eigenschaften der Superklassen.

Views für den Erfasser Um den Erfasser bei seiner Arbeit zu unterstützen, muß ihm eine ausgesuchte Menge von Views angeboten werden. Die Views sind natürlich von der jeweiligen Rahmenbetriebskonzeptsebene, die bearbeitet werden soll, abhängig.

Erfasserviews für die Dienstebene

1. Erfassung von Objekten der Dienstebene:
   Dem Erfasser muß ein View angeboten werden, der es ihm ermöglicht, Dienste, Dienstkataloge und Dienstpakete in das System einzugeben. Dazu stellt man ihm eine Maske zur Verfügung, die das entsprechende Template enthält. Mögliche Templates sind:

Templates der Dienstebene	
`Attributname`	*Typ*
Template für einen Dienst	
`Name des Dienstes:`	*String*
`Diensttyp:`	*intern/extern*
`Gehört zu Dienstpaket:`	*Dienstpaket*
`Diensterbringer:`	*OrgE*
`Dienstnutzer:`	*Liste von OrgE*
`Regulierer:`	*OrgE*
`Steuernde Aufgabe:`	*Steuernde Aufgabe*
`Dienstbeschreibung:`	*String*
`Qualität (QoS):`	*String*
`Abrechnung:`	*pauschal/variabel*
`Attributname`	*Typ*
Templates der Dienstebene	

Templates der Dienstebene	
`Attributname`	*Typ*
Template für einen Dienstkatalog	
`Organisationseinheit:` `Besteht aus Dienstpaketen:`	*OrgE* *Liste von Dienstpaketen*
Template für ein Dienstpaket	

`Gehört zu Dienstkatalog:`
`Dienstpakettyp:`
`Besteht aus Diensten:`
	Dienstkatalog
	intern/extern
	Liste von Diensten

`Attributname`	*Typ*
Templates der Dienstebene	

2. Ändern von Attributwerten:
   Diese Anforderung besteht auch in den anderen Ebenen des Rahmenbetriebskonzepts. Eine Beschreibung wird aber nur an dieser Stelle vorgenommen. Die in der oben eingeführten Datenstruktur enthaltene Information unterliegt einem stetem Wandel. Es muß daher eine Funktionalität zur Modifikation zur Verfügung gestellt werden.

3. Einhängen neuer Dienste:
   Dieser Teilaspekt ist fast identisch mit der Erfassung neuer Information. Das Einhängen von neuen Diensten kann nur dann automatisch erledigt werden, wenn der Kontext des Dienstes vollständig bekannt ist, Voraussetzung dafür ist die bereits vorgenommene Erfassung der zugrundeliegenden Objekte der Aufgaben-, Verfahrens- und Werkzeugebene.

   Wird ein neuer Dienst in die bereits bestehende Umgebung „eingehängt“, so sind die Objekte der Dienstebene automatisch zu modifizieren:

   - der *Dienstkatalog*, wenn der Dienst bisher nicht darin enthalten war (das kann über das Attribut `Diensterbringer` mit dem Typ `OrgE` geschehen, da es für jede Organisationseinheit immer nur einen Dienstkatalog gibt) und
   - das entsprechende *Dienstpaket*, falls der Dienst einem solchen zugeordnet wird. Das richtige Dienstpaket kann direkt aus dem Diensttemplate abgeleitet werden, da ein Attribut `Gehört zu Dienstpaket` existiert.

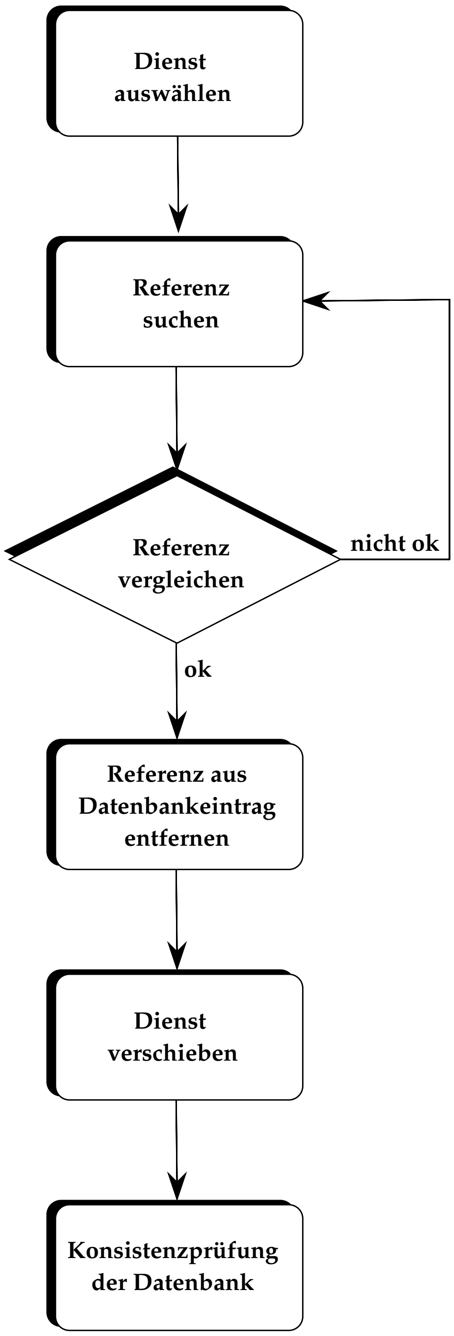

Abbildung 4.2: Löschen und Archivieren eines Dienstes

4. Löschen und Archivieren von Diensten:
   Der Erfasser wählt lediglich den zu löschenden Dienst aus, das System hat

daraufhin den Datenbestand zu durchsuchen und alle Referenzen, die auf das zu löschende Dienstobjekt zeigen, zu entfernen. Der als gelöscht markierte Eintrag sollte archiviert werden, um diesen Dienst zu einem späteren Zeitpunkt wieder anbieten zu können. Am Ende einer „Lösch- und Archivierungsaktion“ ist der Datenbestand auf seine Konsistenz hin zu überprüfen. Dadurch werden Referenzen auf ein bereits gelöschtes Objekt entdeckt und beseitigt.

Der Autor ist sich im klaren, daß die Sicherung der Datenkonsistenz einen großen Verbrauch von Systemressourcen mit sich bringt. Daher ist es nicht sinnvoll, nach jeder Löschaktion eine Konsistenzprüfung vorzunehmen. Diese Prüfung ist am Ende einer „Arbeitssitzung“ oder, falls es ausreichend ist, zu Zeiten, in denen die Systembelastung relativ gering ist, wie z.B. in der Nacht oder am Wochenende, vorzunehmen.

5. Änderung des Objekttyps von einem Dienst zu einer Aufgabe:
   Eine derartige Änderung kann z.B. durch eine Umstrukturierung der Organisation notwendig werden, da der Dienst nicht mehr explizit angeboten wird, sondern nur noch in Form einer Aufgabe existiert.

   Strategie:

   (a) Schritt 1:
   Es wird ein neues Objekt Ausführungs-Aufgabe oder Basisaufgabe angelegt; die Entscheidung des Objekttyps ist dem Anwender zu überlassen. In dieses Template werden die relevanten Attributwerte des Dienstes an der korrespondierenden Stelle eingefügt. Die Aufgabe erhält den Namen des Dienstes.

   (b) Schritt 2:
   Das Dienstobjekt und die zugehörige Steuernde Aufgabe werden gelöscht.

   (c) Schritt 3:
   Die ursprünglichen Ausführungs-Aufgaben werden zu Teilaufgaben, deren Templates identisch sind. Nur das Attribut `Aufgabentyp` muß geändert werden, die Basisaufgaben bleiben erhalten.

   (d) Schritt 4:
   Die Teilaufgaben werden in die neu angelegte Aufgabe eingehängt.

   (e) Schritt 5:
   Die darunterliegenden Ebenen bleiben unberührt.

Erfasserviews für die Aufgabenebene Auf der Aufgabenebene ergeben sich zum Teil ähnliche Views wie auf der Dienstebene.

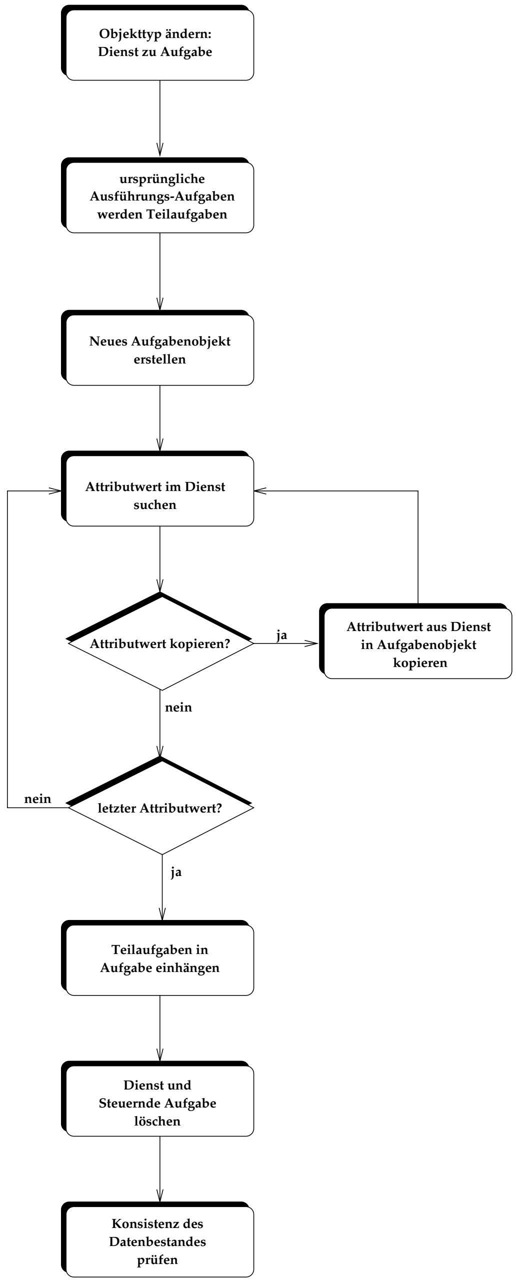

Abbildung 4.3: Umwandlung eines Dienstes in eine Aufgabe

1. Erfassen der Objekte der Aufgabenebene:
   Wie bei der Erfassung von Diensten müssen auch hier dem Erfasser Templates

zur Verfügung gestellt werden, die wie folgt aussehen könnten:

Templates der Aufgabenebene	
`Attributname`	*Typ*
Template für einen Aufgabenkatalog	
`Organisationsname:` `Besteht aus Aufgabenfeldern:`	*OrgE* *Liste von Aufgabenfelder*
Template für ein Aufgabenfeld	
`Gehört zu Aufgabenkatalog:` `Besteht aus Aufgaben:`	*Aufgabenkatalog* *Liste von Aufgaben*
Template für eine Aufgabe	

`Name der Aufgabe:`
`Aufgabentyp:`
`Regulierer:`
`Qualität:`
`Abrechnung:`
	String
	Aufgabentyp
	Rolle/OrgE
	String
	pauschal/variabel

`Typabhängige Attribute`	
`Attributname`	*Typ*
Templates der Aufgabenebene	

Es gibt zwei Klassen von Aufgaben, die mit unterschiedlichen Attributen besetzt sind. Es handelt sich dabei um

(a) *Steuernde Aufgabe* mit den Attributen:

Templates der Aufgabenebene	
`Aufgabenabhängige Attribute`	*Typ*
Spezifisches Template für eine Steuernde Aufgabe	

`Dienstzuordnung:`
`Steuernde Stelle:`
`Aufgaben:`
`Nebenbedingungen:`
	Dienst
	Rolle
	Liste von Ausführungs-Aufgaben
	String

`Aufgabenabhängige Attribute`	*Typ*
Templates der Aufgabenebene	

(b) *Ausführungs-Aufgabe* mit den Attributen:

Templates der Aufgabenebene	
`Aufgabentypabhängige Attribute`	*Typ*
Spezifisches Template für eine Ausführungs-Aufgabe	

`Ausführungs-Aufgabe:`
 	
`Aufgabenbeschreibung:`
`Aufgabenerfüller:`
`Verfahren:`
	Dienstbezogene Aufgabe/Basisaufgabe
	String
	Rolle
	Liste von Verfahren

`Aufgabenabhängige Attribute`	*Typ*
Templates der Aufgabenebene	

Hier wird schon erkennbar, wie komplex die Zusammenhänge sind, da eine Steuernde Aufgabe (nach Definition im Rahmenbetriebskonzept) immer eine Basisaufgabe darstellt. Die Basisaufgabe wird durch das Template für eine Ausführungs-Aufgabe mit abgedeckt, da eine Basisaufgabe, von der Struktur her identisch mit einer Ausführungs-Aufgabe ist. Der Unterschied zwischen beiden Aufgabentypen besteht in der Zuordnung der Ausführungs-Aufgabe zu einem Dienst, was bei einer Basisaufgabe nicht der Fall ist.

2. Einhängen neuer Aufgaben:
   Die Datenstruktur gestattet auch hier schon ein automatisches Einhängen von Aufgaben in den Kontext.

3. Ändern von Attributwerten:
   Im Grunde treten hier die gleichen Anforderungen wie in der Dienstebene auf, die durch eine weitere Ausprägung ergänzt werden: Die Umwandlung eines Aufgabentyps in einen anderen (z.B. eine Ausführungs-Aufgabe wird Basisaufgabe). Die aus der Transformation entstehenden Seiteneffekte auf die Beziehungen wurden nicht weiter untersucht.

4. Änderung des Objekttyps von einer Aufgabe zu einem Dienst:
   Bei einer solchen Änderung kann nur eine Ausführungs-Aufgabe zu einem Dienst erhoben werden. Eine Steuernde Aufgabe ist immer fest mit einem Dienst assoziiert und enthält Steuer- und Regelinformation, die zur Erbringung dieses Dienstes notwendig ist.

Strategie:

(a) Schritt 1:
Die Attributwerte der Aufgabe werden in die entsprechenden Attribute eines Diensttemplates kopiert und ggf. noch ergänzt. Der Name des neu entstandenen Dienstes ist identisch mit dem Namen der Aufgabe, die verschoben wird.

(b) Schritt 2:
Es ist eine neue Steuernde Aufgabe für diesen Dienst anzulegen, die aber nur ein Hilfsobjekt darstellt, um die Struktur des Rahmenbetriebskonzepts zu erhalten. In dieser Steuernden Aufgabe besteht die Steuer- und Regelinformation darin, daß die ursprüngliche Aufgabe auszuführen ist.

(c) Schritt 3:
Die Objekte der Verfahrens- und Werkzeugebene bleiben unangetastet.

Die unter der betreffenden Aufgabe liegenden Objekte lassen sich nicht verschieben, da sich die „Urprungsobjekte“ nicht auf die entsprechenden „Zielobjekte“ abbilden lassen. Aus einem Verfahren müßte eine Steuernde Aufgabe generiert werden, deren Steuer- und Regelinformation aus dem Verfahren abzuleiten wären. Es ist durchaus möglich, daß eine Aufgabe durch mehrere alternative Verfahren erfüllt werden kann. Dadurch entsteht ein Widerspruch zu den Definitionen des Rahmenbetriebskonzepts, die immer nur <u>eine</u> Steuernde Aufgabe pro Dienst vorsehen.

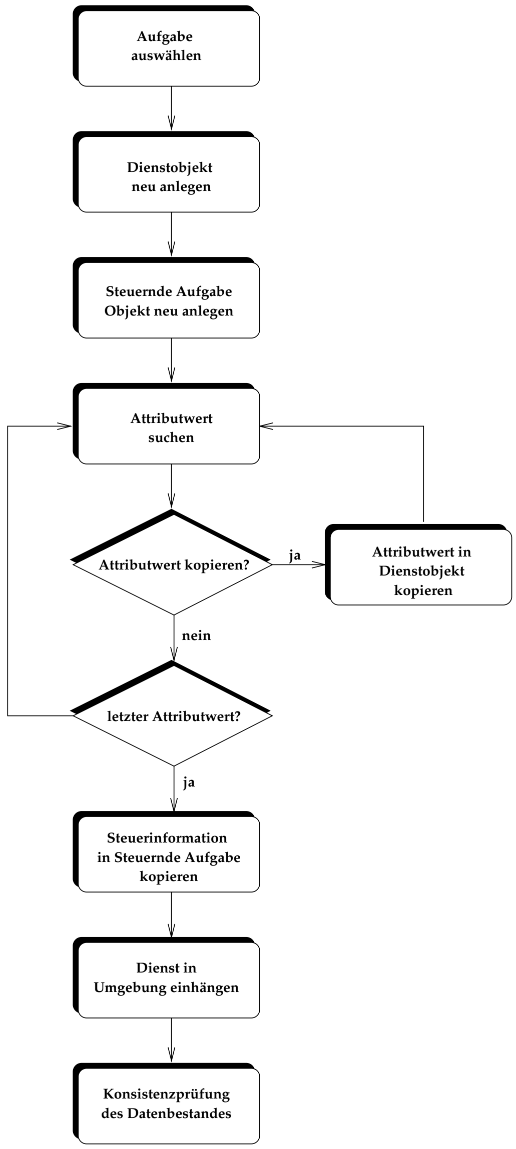

Abbildung 4.4: Umwandlung einer Aufgabe in einen Dienst

Erfasserviews für die Verfahrens- und Werkzeugebene Die Erfasserviews dieser Ebene lassen sich in zwei übergeordnete Aspekte gliedern:

1. Eingabe und Einhängen von Verfahren und Werkzeugen mit folgenden Masken:

(a) Templates der Verfahrensebene:

Templates der Verfahrensebene	
`Attributname`	*Typ*
Template für ein Verfahren	

`Name des Verfahrens:`
`Verwendet bei Diensten:`
`Nebenbedingungen:`
	String
	Liste von Diensten
	String

Template für ein Teilverfahren	

`Name des Teilverfahrens:`
`Beschreibung:`
`Werkzeugabstützung:`
`Ablauf:`
	String
	String
	Werkzeug
	attributierte Liste von Aktionen

Template für eine Aktion	

`Name der Aktion:`
`Beschreibung:`
`Werkzeugfunktion:`
	String
	String
	Funktion

`Attributname`	*Typ*
Templates der Verfahrensebene	

(b) Werkzeugebene:

Templates der Werkzeugebene	
`Attributname`	*Typ*
Template für ein Werkzeug	

`Name des Werkzeugs:`
`Einsatz bei:`
`Verwendet von:`
	String
	Liste von Verfahren
	Liste von Rollen

Template für eine Funktion	

`Name der Funktion:`
`Beschreibung:`
`Werkzeug:`
	String
	String
	Liste von Werkzeugen

`Attributname`	*Typ*
Templates der Werkzeugebene	
`Attributname`	*Typ*
Templates der Werkzeugebene	

2. Ändern von Attributwerten:
   Auch hier ergeben sich ähnlich gelagerte Probleme wie in den Ebenen zuvor.

Szenarien zum Erfassen

- Einfügen eines neuen Knotens

  1. Auswahl Knotentyp
  2. Entsprechende Maske aktivieren (vorbelegt mit gewissen Parametern für administrative Zwecke, die z.B. wegen der Wartungsproblematik benötigt werden)
  3. Aus der Maske das richtige Datenformat generieren (dem Anwender wird das Format verschattet)
  4. Graphische Struktur anzeigen

- Beziehung einfügen
  Beziehungen sollen nicht durch eine textuelle Eingabe vorgenommen werden, sondern durch ein Graphikwerkzeug. Dadurch wird dem Benutzer die Eingabe von fehleranfälligen Texten erspart.
  1. Markieren zweier Knoten
  2. Identifizieren von Vater- und Sohnknoten
  3. Prüfung, ob die gewünschte Beziehung im Rahmenbetriebskonzept zugelassen ist.
     (a) Wenn Beziehung zulässig, dann füge Beziehung in den betreffenden Datensatz ein
     (b) Wenn Beziehung nicht zulässig, dann weise „Request“ zurück
  4. Graphische Struktur anzeigen
- Einfügen eines Sohnknotens
  1. Vaterknoten identifizieren
  2. Vorauswahl für Sohnknotentyp treffen
  3. Sohnknotentyp vom Anwender auswählen lassen
  4. Entsprechende Maske aktivieren
  5. Aus Maske Datenformat generieren
  6. Beziehung in den Vaterknoten einfügen (keine Prüfung des Beziehungstyps notwendig, da diese bereits vorgenommen wurde)
  7. Ggf. Beziehung in den Sohnknoten einfügen (Stichwort: Backlinks)
  8. Graphische Struktur anzeigen
- Löschen eines Knotens
  1. Identifikation des zu löschenden Knotens
  2. Entsprechenden Eintrag aus der Datenbasis entfernen
  3. Alle Referenzen auf den gelöschten Eintrag aus der Datenbasis entfernen (Stichwort: Irrlaufende Zeiger)
  4. Gelöschten Eintrag an einen definierten Platz archivieren (ggf. sind alle Referenzen auf den entsprechenden Eintrag für eine spätere Wiederverwendung zu archivieren.).
  5. Graphische Struktur anzeigen
- Löschen einer Beziehung
  1. Identifikation der zu löschenden Beziehung

2. Entsprechenden Eintrag aus der Datenbasis auslesen und Beziehung löschen (ggf. Beziehung an einen definierten Platz archivieren)
3. Graphische Struktur anzeigen

- Editieren eines Eintrags
  1. Identifikation des Knotens
  2. Relevante Daten aus dem Eintrag extrahieren
  3. Entsprechende Maske aktivieren und mit Daten füllen
  4. editierten Eintrag wieder in die Datenbasis eintragen

4.3.2 Das Benutzerprofil des „Managers"

Der Manager verfolgt mit dem Werkzeug andere Ziele als z.B. ein Erfasser. Er wünscht eine Visualisierung des Ist-Zustandes, auf dessen Basis er Entscheidungen für das Unternehmen treffen will.

Ziele des Managers Der Manager ist hauptsächlich an einer Analysefunktionalität interessiert, da es in seinen Aufgabenbereich fällt,

- Defizite,
- Fehler und
- Optimierungsmöglichkeiten

zu erkennen. Eine derartige Aufgabenstellung ist sehr komplex und eine automatische Analysefunktion ist sicher, wenn überhaupt, nur durch ein Expertensystem zu lösen, das sich auf ein kleines Problemfeld konzentriert.

Daher wird im weiteren Verlauf auf eine reine Visualisierung des Ist-Zustandes (verschiedene Views) eingegangen, die dem Manager als Entscheidungshilfe dient. Interessant für einen Manager sind u.a.

- die Optimierung des Dienstangebots (Outsourcing?),
- die Struktur Kunde/Lieferant und
- das Work-Flow-Management.

Somit erwartet der Manager keinen „Gesamtview" auf die Betreiberorganisation, sondern vielmehr eine Einengung auf spezielle Teilaspekte, die er in Form von Views aufbereitet haben will. Informationen für solche Views lassen sich durch Filtermechanismen auf Objekte und Beziehungen innerhalb der Datenbasis gewinnen. Technisch gesehen handelt es sich um Query-Anfragen an die Datenbasis, die in einer Anfragesprache gestellt werden müssen. Durch dieses offene Design ist es möglich, flexible Anfragen an die Datenbasis zu stellen. Dabei ist zu überlegen, ob nicht ein „Grundset“ von Views zur Verfügung gestellt wird.

Die Arbeit soll sich aber hauptsächlich mit den Problemen, die der Erfasser hat, beschäftigen. Die Probleme der Analyse werden nicht weiter betrachtet.

Teil II

Konkrete Implementierung eines Werkzeugs zur Erfassung des Rahmenbetriebskonzepts

Kapitel 5

Mögliche Softwarearchitektur

Nachdem in Kapitel 3 das Werkzeug motiviert und in Kapitel 4 Anforderungen gestellt wurden, tritt die Softwareentwicklung in eine neue Phase, die Systemdesignphase. In dieser Phase wird eine Architektur entworfen, die die gestellten Anforderungen möglichst optimal erfüllt.

5.1 Grundlegende Konzepte

Anhand der grundlegenden Anforderungen an das Werkzeug, wie sie in Abschnitt 4.1 geschildert werden, wird ein Konzept entworfen. Die Realisierung geschieht über verschiedene Module, die über genau zu spezifizierende Schnittstellen gekoppelt sind.

5.1.1 Verteiltes Arbeiten

Das Werkzeug muß es gestatten, daß verschiedene Benutzer gleichzeitig auf unterschiedlichen Datenbeständen arbeiten. Die angestrebte verteilte Anwendung wird durch eine Client/Server-Architektur realisiert, die einige Vorteile (vgl. [Tan92]) gegenüber einem Stand-Alone-System bietet:

- gemeinsame Nutzung von Ressourcen,
- Kommunikation zwischen verschiedenen Systemen und die
- verbesserte Ausnutzung bzgl. Funktionalität und Flexibilität von Arbeitsplatzrechnern.

In vernetzten Strukturen haben verteilte Anwendungen weitere Vorteile:

- Verbesserte Wirtschaftlichkeit

- Verbesserte Verfügbarkeit durch Replikation von Daten
- Erweiterbarkeit

Es darf jedoch nicht verschwiegen werden, daß verteilte Anwendungen auch mit Nachteilen behaftet sind. Nachfolgend sind einige davon beispielhaft aufgeführt:

- Abhängigkeit von der Leistung und Zuverlässigkeit des Netzes,
- erhöhtes Sicherheitsrisiko und die
- Komplexität der Software

Der Autor ist der Meinung, daß die Konzeption des Werkzeugs als eine Verteilte Anwendung dem Benutzer mehr Vorteile als Nachteile bietet.

5.1.2 Datenhaltung

In einer verteilten Anwendung stellt sich die Frage nach der Verwaltung des benutzten Datenbestands. Grundsätzlich gibt es zwei verschiedene Methoden der Datenhaltung, wobei jede sowohl Vorteile als auch Nachteile hat.

Verteilter Datenbestand:

- Vorteile:

  1. Geringere Netzbelastung durch eine vorteilhafte Verteilung der Daten.
  2. Jeder Anwender kann ohne große Verzögerungen auf den Datenbestand zugreifen, da die Daten nicht über das Netz geholt werden müssen.

- Nachteile:

  1. Probleme bei der Erhaltung der Konsistenz des verteilten Datenbestands
  2. Strategie zum Verteilen des Datenbestands notwendig
  3. Sicherheit in den verteilten Datenbeständen ist nicht gewährleistet. Es ist durchaus möglich, daß nicht autorisierte Personen auf einen Datenbestand Zugriff haben.
  4. Strategie für Backup des verteilten Datenbestands notwendig

Abschließend kann festgestellt werden, daß die Software für die Verwaltung eines verteilten Datenbestands sehr komplex und daher auch fehleranfällig ist.

Zentraler Datenbestand: Die Vorteile dieser Methode sind zum großen Teil komplementär zu den Nachteilen eines verteilten Datenbestands.

- Vorteile:

  1. Konsistenz der Daten kann über Sperrmechanismen (z.B. Semaphore) gewährleistet werden.
  2. Authentisierung kann zentral geschehen und ist somit eindeutig
  3. Sicherheit kann durch spezielle Verschlüsselungsalgorithmen optimiert werden.
  4. Backup repräsentiert den Gesamtdatenbestand.

- Nachteile:

  1. Erhöhte Netzbelastung
  2. evt. verlängerte Antwortzeit bei Anfragen an die Datenbank

Die erhöhte Netzbelastung stellt ein ernstzunehmendes Problem dar. Die daraus resultierenden längeren Wartezeiten bei Überlastung des Netzes führen zwangsläufig zu einem Performanceverlust und dadurch nicht selten zu einem Motivationsverlust des Anwenders. Das Design und die Implementierung einer zentralen Datenhaltung ist um ein Vielfaches unkomplizierter als bei einer verteilten Datenhaltung. Darüber hinaus existieren bereits Konzepte, wie Daten konsistent und sicher gehalten werden können. Solche Konzepte sind bei einer verteilten Datenhaltung noch Gegenstand der Forschung.

Abschließend kann festgestellt werden, daß zur Zeit eine zentrale Datenhaltung einer verteilten Datenhaltung vorzuziehen ist.

5.1.3 Anzeigen und Editieren des Datenbestands

Wie bereits im Abschnitt 4.3 dargestellt, sind auf dem Datenbestand Schreib- und Leseoperationen durchzuführen. Die Daten aus dem Datenbestand sind dem Anwender des Werkzeugs in einer Maske zu präsentieren. Mit Hilfe dieser Maske können nicht nur die bereits im Datenbestand enthaltenen Informationen angezeigt, sondern auch neue Daten in den Datenbestand eingebracht werden. Die Vorteile liegen auf der Hand:

- Durch die Erfassung der Daten mit Schablonen ist eine Interpretation der Eingabedaten verbunden, d.h. diese Daten werden schon hier mit wesentlicher Semantik versehen.

- Informationen aus dem Datenbestand werden einfacher lesbar.
- Neueingabe und Modifikationen von Einträgen in den Datenbestand werden erleichtert.
- Die Transparenz des Datenformats dem Benutzer gegenüber erlaubt die Migration zu einem neuen Datenformat, ohne daß sich die Benutzerumgebung verändert

Nachteile ergeben sich durch ein derartiges Modul nicht. Die Eingabemasken sind durch die Templates aus Abschnitt 4.3.1 bereits vorgegeben und müssen somit nur noch implementiert werden. In die Masken werden nur Attribute und keine Beziehungen eingetragen, da diese in der graphischen Darstellung nachzutragen sind.

5.1.4 Navigation

Die vermaschte Struktur des Rahmenbetriebskonzepts läßt es sinnvoll erscheinen, den Datenbestand neben der textuellen Darstellung eines Eintrags, diesen auch graphisch aufzubereiten. Der Anwender kann sich daraus den Kontext des Eintrags intuitiv erschließen und ist nicht nur auf die isolierte textuelle Sicht beschränkt, die den Eintrag aus dem Zusammenhang reißt.

Neben der graphischen Darstellung von Views auf den Datenbestand wird das Modul verwendet, um Beziehungen zwischen verschiedenen Einträgen des Datenbestands einzugeben. Dadurch wird die manuelle Eingabe anderer Objektbezeichner als Text durch den Anwender vermieden und als Fehlerquelle ausgeschlossen.

5.2 Grundlegende Architektur

Aufgrund der obigen Ausführungen wurde die in Abbildung 5.1 gezeigte Architektur entwickelt. Das System basiert auf einer Client/Server-Architektur. Der Client besteht aus drei großen Modulen:

1. dem zentralen Steuermodul,
2. einem Anzeige- und Editiermodul und
3. einem Navigationsmodul.

Zwischen diesen Modulen des Clients befinden sich auch zwei Schnittstellen, die noch genauer zu spezifizieren sind.

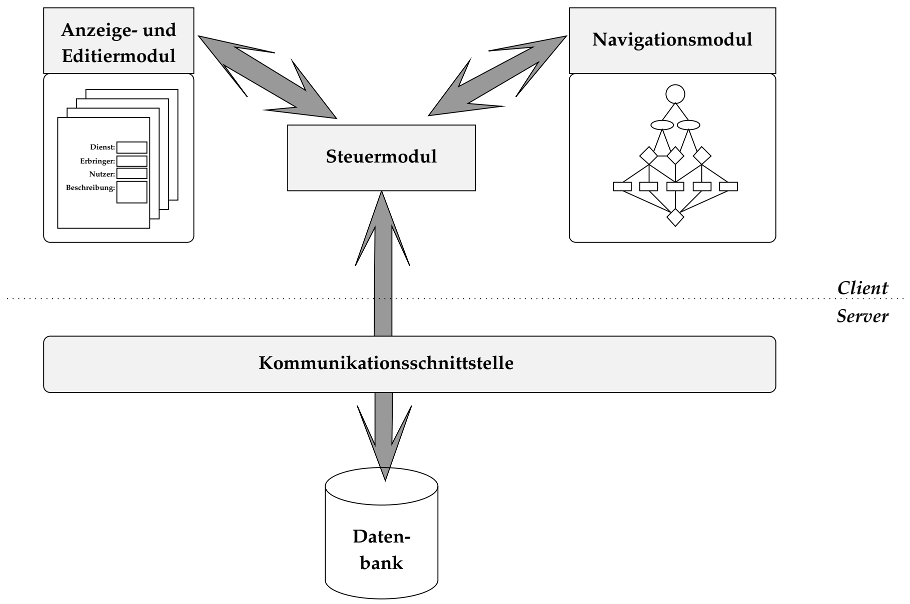

Abbildung 5.1: Grundlegende Architektur

5.2.1 Schnittstellen des Werkzeugs

Innerhalb des Werkzeugs existieren drei Arten von Schnittstellen, die in Abbildung 5.1 als Pfeile dargestellt sind;

1. die Schnittstelle zwischen dem Client und dem Server,
2. die Schnittstelle zwischen dem Anzeige-/Editiermodul und dem Steuermodul sowie
3. die Schnittstelle zwischen dem Navigations- und dem Steuermodul.

Schnittstelle zwischen Client und Server: Diese Schnittstelle wird durch ein Kommunikationsprotokoll realisiert. Aus der Vielzahl der denkbaren Kommunikationsprotokolle, die eingesetzt werden könnten (http, ftp, etc.), wird üblicherweise ein standardisiertes eingesetzt.

Schnittstelle zwischen Anzeige-/Editiermodul und Steuermodul: Das anzuzeigende Dokument wird im Navigationsmodul ausgewählt und anschließend im

Anzeige-/Editiermodul dargestellt. Das Steuermodul muß anhand eines vom Navigationsmodul gesendeten Signals den gewünschten Eintrag in der zentralen Datenbank auf der Serverseite identifizieren und diesen für das Anzeige-/Editiermodul bereitstellen. Die Schnittstelle orientiert sich am verwendeten Datenformat.

Schnittstelle zwischen Navigations- und Steuermodul: Das Steuermodul verfügt über eine Schnittstelle zum Navigationsmodul. Dadurch kann ein beliebiges Werkzeug, das zum Darstellen von Graphen geeignet ist, eingesetzt werden. Der gesamte Graph ist in einem Datenformat bereitzustellen, das durch einen Konverter in das Datenformat des Navigationsmoduls umgewandelt und vom Navigationsmodul in Form des gewünschten Graphen dargestellt wird.

5.3 Module eines Werkzeugs

Die groben Bausteinen eines Werkzeugs, wie sie unter Kapitel 5.2 beschrieben werden, lassen sich noch in kleinere Bestandteile zerlegen.

Beispiele:

- Parser, zum feststellen der Strukturinformation,
- Konsistenzprüfer,
- Benutzerverwaltung, etc.

Nachfolgend werden einige Module skizziert.

5.3.1 Steuermodul

Das Steuermodul bildet das Herzstück des gesamten Systems. Die zu erledigenden Aufgaben lassen sich durch einzelne Module realisieren.

Kriterienliste für das Steuermodul

Die Kriterienliste läßt sich direkt aus den Aufgaben, die das Steuermodul zu erfüllen hat, ableiten:

- Scheduling-Mechanismus
- Benutzerverwaltung

- Maskengenerator zum Erstellen und Editieren von Einträgen
- Konsistenzprüfer auf die Datenbank
- Bereitstellen der Kommunikationsinfrastruktur

Scheduling-Mechanismus: Der Scheduler überwacht alle innerhalb des Steuermoduls existierenden Teilmodule. Es verteilt die Aufgaben an die Module und wartet auf eine Rückmeldung. Der Scheduler hat somit einen Überblick über den Gesamtzustand des Systems.

Benutzerverwaltung: Die unterschiedliche Klassifikation von Benutzern, in Benutzertypen, fordert eine grundlegend verschiedene Behandlung. Es ist ein Modul zu integrieren, das eine Klassifikation der Benutzer in Gruppen zuläßt, wie es z.B. Unix gestattet. Diesen Klassen werden verschiedene Rechte eingeräumt, z.B.

- dem Erfasser. Er ist zuständig für die Eingabe der Daten, sowie deren Pflege. Das bedeutet, es muß ihm Schreibrecht auf die von ihm verwalteten Datenbestände gewährt werden; auf anderen Datenbeständen nur ein eingeschränktes Leserecht (abhängig von der Strategie des Datenschutzes).
- dem Manager. Ihm sind auf dem ihm zustehenden Datenbestand die vollen Leserechte zuzugestehen. Er erhält kein Schreibrecht, um die Konsistenz des Datenbestandes, für die der Erfasser zuständig ist, zu erhalten (durch weniger Schreibvorgänge auf dem Datenbestand).

Neben der dem Aspekt der Datensicherheit und der Einschränkung des Konsistenzproblems wird dadurch auch eine Pflege der Datenbestände erleichtert. Das Sicherheitsmodell nach [Mul89] teilt sich in drei Bereiche auf:

1. *Authentication:*
   For every request to an operation, the name of the user or computer system making the request is known reliably. The source of a request is called a principle.

2. *Access Control:*
   For every resource (computer, printer, file, database etc.) and every operation on that resource (read, write, delete etc.), it is possible to specify the names of the principles allowed to do that operation on that resource. Every request for an operation is checked to ensure that its principle is allowed to do that operations.

3. *Auditing:*
   Every access to a resource can be logged if desired, as can the evidence used to authenticate every request. If trouble comes up, there is a record of exactly what happens.

Dieses Sicherheitskonzept ist sehr flexibel und erlaubt die Bildung von Benutzergruppen, denen unterschiedliche, abhängig von ihren Aufgaben Rechte zugewiesen werden. So kann der Zugriff auf einen Datenbestand flexibel gehandhabt werden, durch das Mitprotokollieren der einzelnen Operationen wird die Fehlersuche wesentlich vereinfacht.

Maskengenerator: Ein Maskengenerator kann in nahezu jedem Schritt der Erfassung eingesetzt werden. Dem Anwender wird durch die Masken eine Hilfe bereitgestellt, mit welchen Inhalten die Einträge in der Datenbank zu füllen sind. Diese Masken können auch zum Editieren von Datenbankeinträgen benutzt werden.

Konsistenzprüfer: Nach jeder Schreiboperation, die auf dem Datenbestand durchgeführt wird, sollte man eine Prüfung auf Konsistenz des Datenbestandes durchführen. Es ist darauf zu achten, daß nicht gleichzeitig mehrere Nutzer auf denselben Eintrag in der Datenbank zugreifen. Diesem Umstand muß durch eine geeignete Strategie Rechnung getragen werden; denkbar wäre eine Lösung durch einen Semaphormechanismus.

Bereitstellung der Kommunikationsinfrastruktur: Das Steuermodul stellt die Kommunikationsinfrastruktur zur Verfügung. Es initiiert die Kommunikation und beendet diese gegebenenfalls, d.h. dieses Modul überwacht sowohl die Kommunikation mit den anderen Clientmodulen und die Kommunikation mit dem Server. Diese Aufgabe erfordert einen Scheduling-Mechanismus.

5.3.2 Navigationsmodul

Die komplexe Struktur des instantiierten Rahmenbetriebskonzepts wird durch eine graphische Aufbereitung für Anwender besser verständlich. Die Kriterienliste des Navigationsmoduls ist in drei Teile gegliedert:

- Systemlandschaft
- Layout
- Funktionalitätsspezifikation

Systemlandschaft:

Unix/Linux: Wird ein kommerzielles Produkt gewählt, so ist darauf zu achten, daß das Navigationswerkzeug unter dem Betriebssystem Unix lauffähig ist, da die Entwicklungsumgebung hier am Lehrstuhl unter diesem Betriebssystem läuft. Für eine Portierung auf Laptops sollte Linux unterstützt werden. Diese Laptops können zu Vorführungen außerhalb des Lehrstuhles einfacher genutzt werden.

Performance: Die Bearbeitungs- und Antwortzeiten dürfen nicht zu unangemessenen Wartezeiten führen, z.B. ist eine Antwortzeit zum Markieren eines Knotens von mehr als zwei Sekunden nicht mehr akzeptabel.

Layout:

Dieser Abschnitt stellt Kriterien auf, die sich auf das Layout beziehen. Dazu gehört die Festlegung der Schrift und die Darstellung von Knoten und Kanten.

Visualisierung: Das Navigationswerkzeug sollte in der Lage sein, jede Art von Graphen (gerichtete, ungerichtete) darzustellen.

Beschriftung: Die Beschriftung der Objekte (Knoten und Kanten des Navigationswerkzeuges) soll der ISO 8859-1 Norm entsprechen, die den deutschen Zeichensatz enthält.

Darstellung von Knoten und Kanten: Innerhalb eines Graphen existieren Knoten mit unterschiedlicher Semantik, die für den Anwender durch unterschiedliche Symbole vor den Knotennamen oder durch unterschiedliche Icons visualisiert sind. Analog zu den unterschiedlichen Knotentypen existieren auch unterschiedliche Beziehungstypen zwischen diesen Knoten. Diese werden durch unterschiedliche Kantentypen symbolisiert; denkbar wäre hier eine Pfeildarstellung, wobei die Pfeillinien unterschiedlich gestaltet werden.

Abhängig von der Sichtweise des Benutzer muß ihm eine auf ihn zugeschnittene Darstellung angeboten werden. Aus diesem Grund muß sich die Semantik der Icons flexibel anpassen.

Darstellung von Farben: Durch den Einsatz von Farben können Gruppen von Objekten gebildet werden, die sofort intuitiv zu erfassen sind.

Unterstützung von verschiedenen Schriftarten: Anhand unterschiedlicher Schriftarten ist es ebenfalls möglich, eine Einteilung der Icons in Gruppen vorzunehmen.

Markieren von Knoten und Kanten: Sind Knoten oder Kanten markiert, sind diese Objekte auch hervorgehoben darzustellen.

Funktionalitätsspezifikation:

Layout-Algorithmus: An den Layout-Algorithmus werden folgende Mindestvoraussetzung gestellt:

- **Ebenendarstellung:** Durch das Rahmenbetriebskonzept werden Knotentypen mit unterschiedlicher Semantik vorgegeben (Dienstpaket, Dienst, Aufgabenfeld, Steuernde Aufgabe, Ausführungs-Aufgabe, etc.). Knoten einer logischen Ebene sollen auch innerhalb einer Ebene des Navigationswerkzeugs dargestellt werden. Wird in einen bereits bestehenden Graphen ein neuer Knoten eingefügt, ist dieser automatisch in die richtige Ebene einzuordnen.
- **Freies Positionieren von Knoten:** Um eine flexible Darstellung zu gewährleisten, müssen Knoten frei verschiebbar sein, d.h. sowohl horizontal, als auch in Sonderfällen vertikal.
- **Zentrieren der Knoten:** Abhängig von der Anzahl der Vaterknoten sind die Sohnknoten zu positionieren:
  - *Ein Vaterknoten:* Die Sohnknoten werden unter dem Vaterknoten zentriert.
  - *Mehrere Vaterknoten:* Neben der Zentrierung der Sohnknoten soll noch eine Minimierung der Kreuzungen von Kanten erreicht werden.
- **Platzoptimierung:** Der Platzbedarf eines Graphen ist zu optimieren, z.B. durch automatisches Umbrechen der Beschriftung eines Objekts (Knoten oder Kante) oder ggf. durch Neuanordnung der Knoten (z.B. Minimieren der Kreuzungen). Die Ebenen müssen aber weiterhin erhalten bleiben.

Darstellung eines großen Graphen: Durch das Werkzeug sollen auch Graphen darstellbar sein, die größer als das definierte Fenster sind, z.B. Scrollbalken.

Kommunikationsschnittstelle: Das Navigationswerkzeug sollte eine bidirektionale Kommunikationsschnittstelle unterstützen, über die es Verbindung mit anderen Prozessen aufnimmt.

Einerseits werden vom Navigationswerkzeug Meldungen an den anderen Prozeß übergeben, andererseits werden vom Navigationswerkzeug Meldungen empfangen, die als Befehle interpretiert werden. Der minimale Befehlssatz umfaßt:

- Laden eines Graphen
- Speichern eines Graphen
- Hinzufügen von neuen Knoten oder Kanten, ohne daß ein komplett neuer Term, der den Graph repräsentiert, erzeugt werden muß (Update-Funktion).
- Sperren von jeder Benutzeraktion (wenn z.B. wenn ein neuer Graph geladen wird oder während der aktuelle Graph modifiziert wird; z.B. bei einer Datenbank-Interaktion)
- Aktivieren des Navigationswerkzeugs (Aufheben der Sperre)

"Event-Binding": Aktionen innerhalb des Navigationswerkzeugs können durch eine Meldung über die Kommunikationsschnittstelle nach aussen gegeben werden. Anhand der ausgegebenen Meldungen werden die angebundenen Applikationen getriggert.

Einhängen eigener Menüs: Das Einhängen von Menüs ermöglicht die flexible Anpassung des Navigationsmoduls an Bedürfnisse des Anwenders.

Ausblenden von Teilgraphen: Sind die angezeigten Graphstrukturen sehr groß, kann durch das Ausblenden eines Teilgraphen die Übersichtlichkeit enorm gesteigert werden.

Unterstützung einer Suchfunktion: Für große Graphen ist eine Suchfunktion, z.B. auf Knotennamen, bereitzustellen.

Meldungen: Zwei Arten von Meldungen sind zu unterscheiden:

- *Statusmeldungen:* Anhand von Statusmeldungen des Navigationswerkzeug wird dem Anwender der momentane Zustand mitgeteilt. So können Irritationen des Anwender bei längeren Wartezeiten vermieden werden (Frage: Arbeitet das Werkzeug oder ist es abgestürzty?).
- *Fehlermeldungen:* Aussagekräftige Fehlermeldungen erleichtern die Fehlersuche und -behebung (Fehlermeldungen, wie "Fehler im Programm" sind zu vermeiden!).

5.3.3 Anzeige- und Editiermodul

Aus den Aufgaben dieses Moduls läßt sich direkt der Kriterienkatalog ableiten:

- Editieren und
- Anzeigen von Informationen

Modul zum Editieren von Informationen: Dieses Modul wird zur Neuerfassung und Modifikation von Informationen eingesetzt. Die relevanten Daten werden durch die vorgegebene Benutzerführung in den Masken vom Anwender abgefragt und in die Datenbank geschrieben.

Modul zum Anzeigen von Informationen: Die Formate der Einträge in den Datensätzen sind auf die Masken abzubilden und die Daten darzustellen. In diesem Modul sind nur Leseoperationen möglich.

Maskengenerator: Um den Anwender nicht zu verwirren, werden in beiden Modulen gleiche Masken verwendet. Durch Aufrufparameter der Maske werden dem Benutzer individuell Schreib- und Leserechte eingeräumt.

5.3.4 Vererbung

Bei der Instantiierung eines Objekts sind bereits bekannte Daten und Relationen durch Vererbungsstrategien zu übernehmen. Generische Templates erleichtern dem Netzbetreiber die Anpassung auf seine spezifischen Anforderungen.

5.4 Angestrebte Architektur

Eine detaillierte Beschreibung dieser Architektur erfolgt anhand der Abbildung 5.2.

Eine verteilte Verarbeitung bedingt eine Client/Server-Architektur. Der Client besteht aus drei großen Bausteinen:

- WWW-Browser:[1]
  Die Entscheidung wurde zugunsten eines HyperText-Systems gefällt, dessen Vor- und Nachteile in Kapitel 6 behandelt werden; daher wird zur textuellen

[1] Die Begriffe HTML, HTTP, WAIS und WWW werden auf Seite 76 erklärt

Darstellung und zum Aufbau von Verbindungen zu Daten-Servern ein WWW-Browser eingesetzt. Der verwendete Browser „TkWWW“ ist vollständig in Tcl/Tk (**T**ool **C**ommand **L**anguage/**T**ool **k**it, vgl. [Ous94]) programmiert worden und bietet eine Tcl/Tk-Schnittstelle, so daß Erweiterungen problemlos vorgenommen werden können.

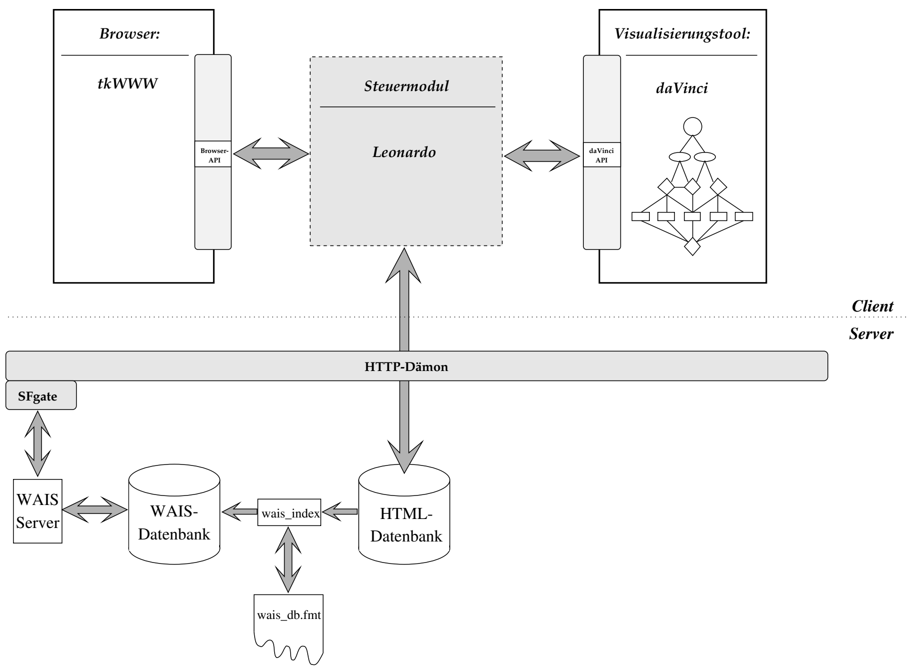

Abbildung 5.2: Angestrebte Architektur des Werkzeugs

- Navigationsmodul:
  Durch die graphische Darstellung lassen sich Zusammenhänge intuitiv und daher schneller verstehen. Dieses Modul wird durch das Werkzeug „daVinci“ realisiert, da es u.a. Graph-Strukturen darstellen und Knoten mit Semantik versehen kann.

- Steuermodul (Leonardo):
  Dieses Modul steuert die Kommunikation zwischen den Clientmodulen untereinander und zum Server.

Der Server besteht aus mehreren Teilen:

- Daten:

  - HTML-Datenbestand:
    Hierin sind die einzelnen HTML-Dokumente mit allen Beziehungen abgelegt.
  - WAIS-Datenbestand:
    Hier ist der gesamte HTML-Datenbestand nochmals abgelegt, aber in einem Format, das es erlaubt, Anfragen über die Struktur (gemeint sind viewspezifische Anfragen) zu stellen. Der WAIS-Datenbestand wird durch einen Server verwaltet.

    Die zur Viewbildung notwendige Strukturinformationen ist durch den Anwender in der Datei „wais_db.fmt“ zu spezifizieren. Bei einer Anfrage greift der WAIS-Server auf diese Datei zu. Bevor dies aber geschehen kann muß die Struktur mit Hilfe des Programms “waisindex“ aufgebaut werden.

- HTTP-Dämon:
  Der Dämon läßt eine verteilte Verarbeitung zu und stellt die Schnittstelle zwischen einem HTTP-Server und einem HTTP-Client dar. Das HTTP-Protokoll ist zustandslos.

- SFgate:
  Dieses Gateway wird an den HTTP-Dämon angegliedert und stellt eine einheitliches Protokoll an der Client/Server-Schnittstelle zur Verfügung. Das SFgate gestattet eine flexible Abfragen von Informationen aus einer WAIS-Datenbank.

Es werden hier Module eingesetzt, die zum großen Teil als Public-Domain-Software verfügbar sind und nur noch an die geforderten Gegebenheiten angepaßt oder erweitert werden müssen.

Kapitel 6

Möglichkeiten der technischen Realisierung eines Werkzeugs

6.1 Entwicklungsgeschichte des Werkzeugs Terra

Die Geschichte des Werkzeugs Terra (**T**ool zur **Er**stellung des **R**ahmenbetriebskonzepts) reicht zurück bis in den März 1994. Zu diesem Zeitpunkt wurde in Zusammenarbeit des Lehrstuhls für Rechnernetze und Rechnerkommunikation unter der Leitung von Prof. Hegering und der Firma TTI TECTRAN das Rahmenbetriebskonzept entwickelt ([AEH+94]). Es wurde schnell deutlich, daß dieses Konzept schneller in die Praxis umgesetzt werden kann, wenn ein geeignetes, unterstützendes Werkzeug zur Verfügung steht.

Ein erste Versuch („Rapid Prototyping") basiert auf den Arbeiten von [May94] und [Vog94]. Die Schwachstellen dieses ersten Prototypen ([HAS+94]) sind stichpunktartig aufgelistet:

- Keine Client/Server-Architektur:
  Das Werkzeug basiert auf einer HTML-Datenbank, die drei Werkzeuge speist:
  1. das Skript *get_struct*, das die HTML-Datenbank nach Links (Referenzen auf andere HTML-Dokumente) parst und das Ergebnis in der Datei *html_struct.dat* ablegt,
  2. das Skript *documents_tree*, das aus der Datei *html_struct.dat* eine Baumdarstellung der Links zwischen den HTML-Dokumenten erzeugt und
  3. den WWW-Browser *TkWWW*, der verwendet wird, um HTML-Dokumente anzuzeigen.

  Die Architektur (vgl. Abbildung 6.1) ist vollkommen flach, d.h. es läßt sich kein Client oder Server erkennen.

- Kein automatisches Aktualisieren der Strukturinformationsdatei:
  Nach jeder in der Datenbank vorgenommen Änderung, mußte der Anwender die Strukturdatei neu anlegen, um mit der aktuellen Struktur zu arbeiten.

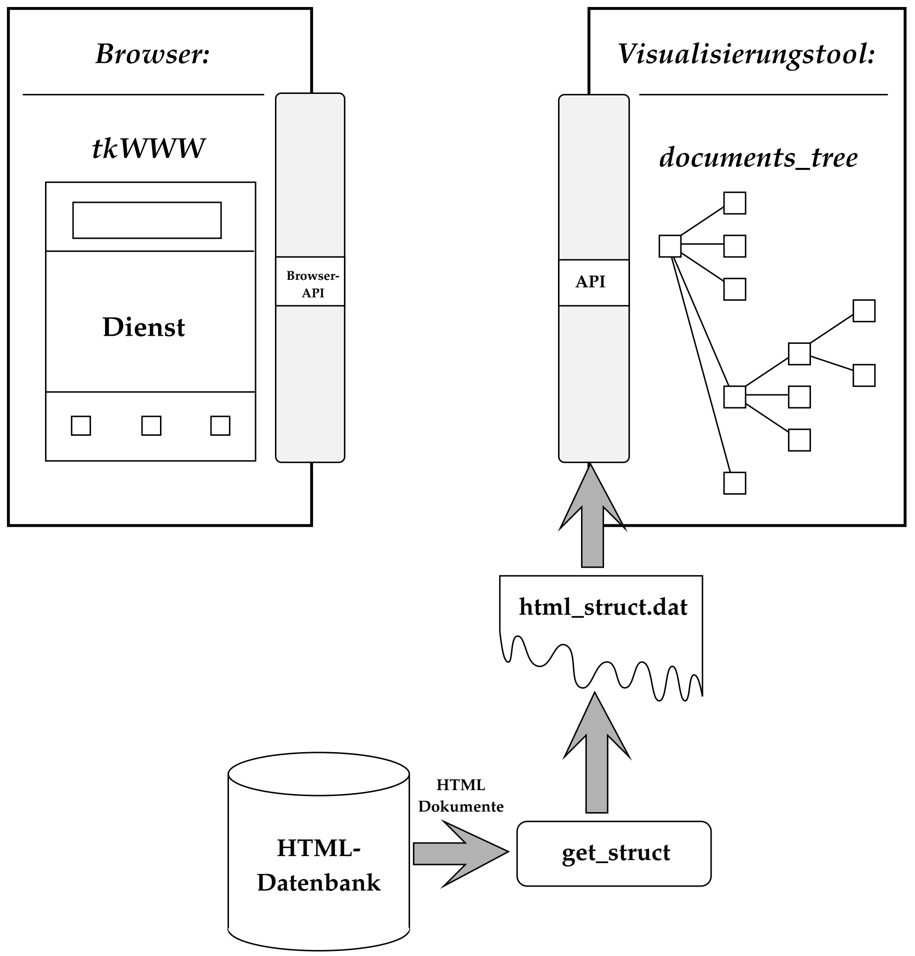

Abbildung 6.1: Aufbau der Baumstruktur

- Keine Graphenstruktur darstellbar:
  Das entwickelte Darstellungswerkzeug *documents_tree* konnte keine Graphenstrukturen sondern nur Baumstrukturen darstellen.

- Keine Klassifikation von Links möglich:
  Der Strukturparser kannte keine „Link-Klassen“.

- Nur Gesamtview darstellbar:
  Es konnte nur eine Totalansicht gezeigt werden. Außerdem konnten die einzelnen Ebenen des Rahmenbetriebskonzepts nicht eindeutig vom Benutzer identifiziert werden.

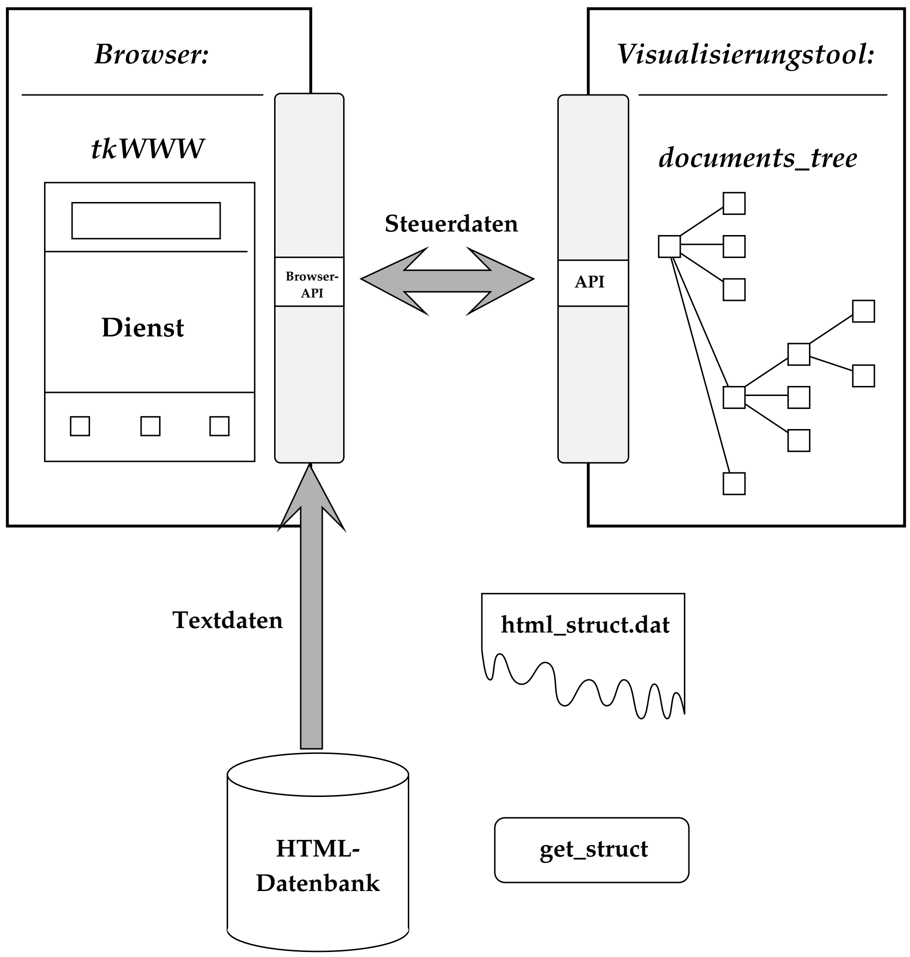

Abbildung 6.2: Abfrage von Dokumenten

Die Abbildungen 6.1 zeigt, wie die im Werkzeug *documents_tree* dargestellte Struktur aus der HTML-Datenbank erzeugt wird. Abbildung 6.2 verdeutlicht die Arbeitsweise nach der Erstellung der Strukturinformationsdatei *html_struct.dat*.

Aus diesem Prototypen wurden jedoch wichtige Erkenntnisse gewonnen, die beim Design und der Implementierung eines neuen Prototypen Verwendung fanden. Man kann auch sagen, daß dieses Vorgehen sicher der „state of the art“ im Bereich des Softwareentwicklung ist (vgl. Abb. 6.3 aus [Den91]).

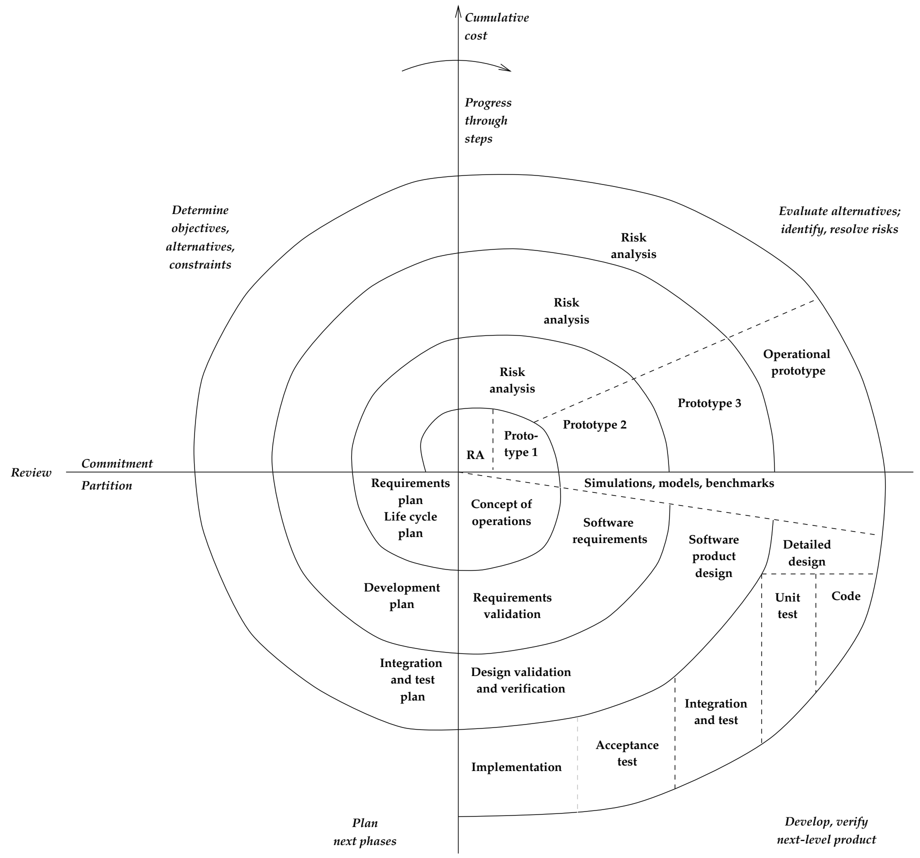

Abbildung 6.3: Spiralenmodell von Softwareentwicklungsprozessen

In vielen Fällen wird die Software an den Bedürfnissen der Anwender „vorbeientwickelt“. Die Anforderungen an ein System sind noch nicht bis ins kleinste Detail spezifiziert worden. Ist die Kommunikation zwischen dem System und dem Menschen besonders intensiv, wie es z.B. beim zu erstellenden Werkzeug Terra der Fall ist, ist eine Anforderungsspezifikation des Benutzers an das System nicht trivial. Bis zu diesem Zeitpunkt kennt er lediglich den Rahmen der zu bewältigenden Aufgaben. Daher wird ein Prototyp entwickelt, der bereits die grundlegenden Funktionen des späteren Produkts enthält. Anhand von Testläufen (Simulationen) werden sukzessive Benutzeranforderungen erarbeitet, deren Umsetzung in einer späteren Version angestrebt wird (vgl. [Bal82]). Der Softwareentwurf ist heute ein dynamischer Vorgang, im Gegensatz zu dem statischen Ansatz, der in früheren Jahren benutzt wurde.

Der statische Ansatz geht nach dem Wasserfallmodell (Abbildung 6.4) vor, während der dynamische Softwareentwurf das Spiralenmodell verwendet. Die Entwicklung des Werkzeugs Terra geht noch dem Spiralmodell vor.

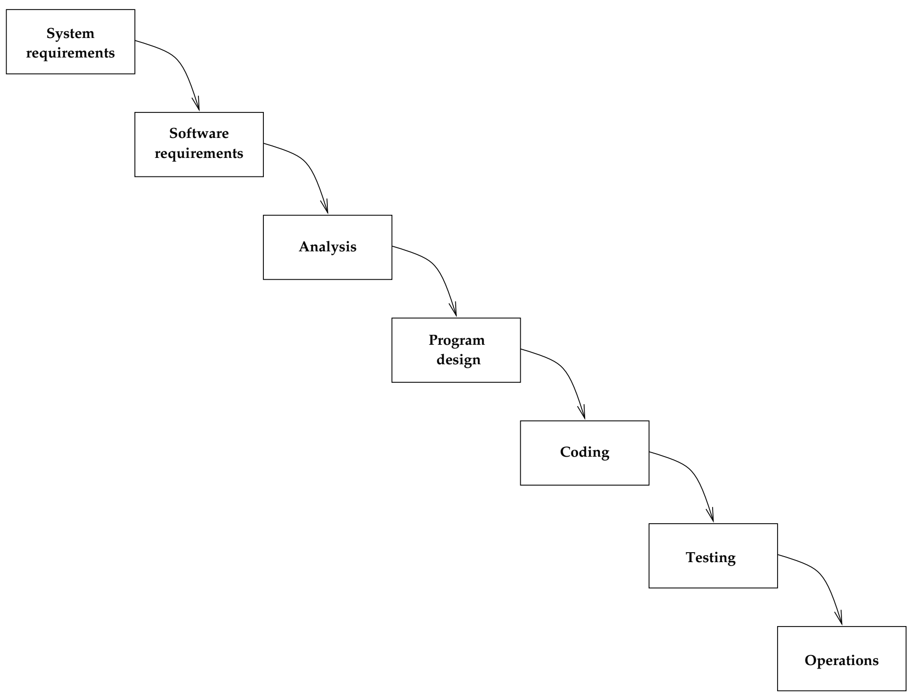

Abbildung 6.4: Klassisches Wasserfallmodell

Aufgrund der Anforderungen an ein Werkzeug muß der Entwickler entscheiden, welches System sich für diese Anforderungen am besten eignet. In diesem Fall wurde ein HyperText-System gewählt, dessen Vor- und Nachteile im Anschluß erläutert werden.

6.2 Das HyperText-System

Die Geschichte der HyperText-Systeme ist stark mit der Ausbreitung des Internets verbunden, dessen nahezu unerschöpflicher Informationsgehalt nicht mehr übersichtlich war. Die HyperText-Systeme können, diese in unterschiedlichen Formaten vorliegenden Informationen visualisieren und auf sie mit einem Werkzeug zuzugreifen – einem Browser –.

Begriffserklärungen:

Im weiteren Verlauf der Arbeit werden Begriffe verwendet, deren Kenntnis im Zusammenhang mit HyperText-Systemen als grundlegend anzusehen ist; werden sie anschließend vorgestellt und erklärt.

- *HTTP (HyperText Transfer Protocol):*
  HTTP (vgl. [BL94]) ist ein Übertragungsprotokoll, bei dessen Entwicklung besonderes Augenmerk auf Einfachheit und Schnelligkeit bei der Übertragung gelegt wurde, da es zur Übertragung von Hypermedia-Information gedacht ist. Es handelt sich um ein generisches, objektorientiertes Protokoll zur Übertragung von Hypermedia-Informationen.

- *SGML (Standard Generalized Markup Language):*
  SGML ist eine ISO-Norm (ISO 8879 [ISO87a]), die Definitionen für die Erstellung von strukturierten Dokumenttypen vorgibt.

- *HTML (HyperText Markup Language):*
  HTML (eine Spezifikation von HTML 2.0 findet sich in [BLCM94]) dient zur Erstellung von plattformunabhängigen HyperText-Dokumenten. Durch HTML kann verschiedene Typen, wie z.B.

  - HyperText news, mail, Dokumentationen, Hypermedia-Anwendungen,
  - Menüs,
  - Ergebnisse von Datenbankanfragen und
  - einfach strukturierte Dokumente mit „Inline-Graphiken“

  darstellen. HTML entspricht dem SGML-Standard.

- *URI (Uniform Resource Identifier):*
  Die URI ist ein Objektidentifikator für verteilte Daten. Dieser besteht aus einem String, der keine Leerzeichen enthält. Die Identifikation kann über zwei verschiedene Methoden erfolgen:

  - *URL (Uniform Resource Locator):*
    Diese Methode beschreibt den physikalische Ort beschrieben, an dem sich ein Objekt befindet. Eine solche URL ist folgendermaßen aufgebaut:
    `Zugriffsart://Server:Portnummer/Dokumentenpfad/Dokumentname`

    1. Zugriffsart:
       An erster Stelle steht die gewünschte Zugriffsart, anhand der erkannt wird, welches Format die ankommenden Pakete aufweisen. Zulässig sind z.B. `http`, `ftp`, `gopher` und `file`.

2. Server:
   Es wird der Server angegeben, auf dem das angewählte Dokument zu finden ist. Dieser Server muß die vom Client angeforderte Zugriffsart unterstützen, da jede Zugriffsart unterschiedliche Datenformate verwendet. Eine Besonderheit bildet die Zugriffsart `file`, da hier nicht auf einen ausgewiesenen Server, sondern auf eine lokale Datei zugegriffen wird. In einem solchen Fall lautet die URL
   `file://localhost/Dokumentenpfad/Dokumentname`
   Die Angabe der Portnummer ist optional und gibt an, über welchen Server-Port die Anfrage gestellt wird. Wird sie nicht angegeben, so wird die voreingestellte Portnummer (bei HTTP handelt es sich um den Port Nummer 80) verwendet.
3. Dokumentenpfad:
   Der im Server konfigurierte Dokumentenpfad wird angegeben; die Pfade entsprechen den in Unix verwendeten.
4. Dokumentname:
   Der physische Name der gewünschten Datei ist anzugeben.

– *URN (Universal Resource Name):*
  Diese Zugriffsmethode wurde bisher nur spezifiziert, eine Implementierung existiert zum jetzigen Zeitpunkt noch nicht. Es wird in einem Namensraum ein das Objekt repräsentierender Name definiert und über diesen auf das Objekt zugegriffen.

- *WWW (World Wide Web):*
  Bezeichnet die Gesamtheit aller Dokumente und Dienste, die im Internet über HTTP und die übrigen gängigen Internetprotokolle erreichbar sind.

6.2.1 Aufbau eines HyperText-Systems

Ein HyperText-System beruht auf einer Client/Server-Architektur. Der Client ist ein sog. „HyperText-Browser“. Die Aufgaben eines Browsers bestehen darin, eine Verbindung zu einem Server herzustellen und aus dessen Datenbasis Daten abzurufen. Diese Daten werden für den Benutzer (User), meist unter Zuhilfenahme einer graphischen Oberfläche, aufbereitet. Betrachtet man sich den Browser genauer, ist dieser nichts anderes als eine Art Dateimanager.

Die Server „verwalten“ den jeweiligen Datenbestand und stellen eine Schnittstelle zur Abfrage der Daten zur Verfügung. Der Browser benutzt diese Schnittstelle für die Kommunikation. Die Flexibilität dieses Systems ermöglicht einem HyperText-Browser eine Verbindung nicht nur zu HTTP-Servern aufzubauen, sondern zu jedem beliebigen Servertyp (WAIS-Server, Gopher-Server, etc.). Für die Kommunikation benutzt der HyperText-Browser das jeweilige Protokoll dieses Servertyps (bei einem

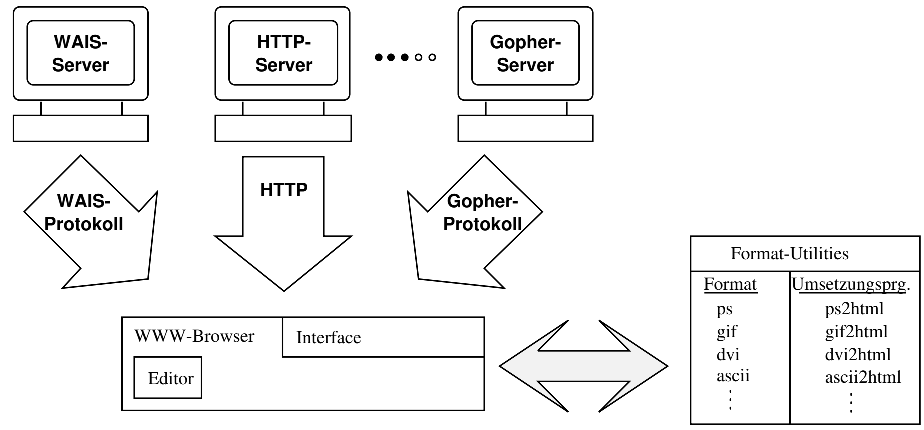

Abbildung 6.5: Arbeitsweise eines HyperText-Browsers

HTTP-Server ist es das Protokoll HTTP). Daher ist das System in der Lage, verschiedenste Datenformate (Postscript, GIF, ASCII, etc.) zu erkennen und zu verarbeiten. Ankommende Daten werden von einem Modul auf ihr Format hin untersucht und durch Aufruf einer externen Prozedur für den Browser aufbereitet. Typischerweise werden HyperText-Daten in sog. HTML-Dokumenten abgelegt. Logische Verknüpfungen zwischen den HTML-Dokumenten werden durch Links realisiert und stellen die Beziehungen zwischen den einzelnen Daten her; eine Struktur wird dabei nicht vorgeschrieben.

6.2.2 Vorteile eines HyperText-Systems

Die Strukturen des Rahmenbetriebskonzepts und die eines HyperText-Systems sind in einigen Bereichen nahezu deckungsgleich. Es folgen einige Betrachtungen, die diesen Sachverhalt unterstreichen.

- Analogie Wurzel – Homepage:
  Die grundlegende Struktur des Rahmenbetriebskonzepts ist ein Geflecht, dessen Wurzel als die sog. Homepage dargestellt werden.

- Analogie Knoten – Dokument:
  Die Objekte des Rahmenbetriebskonzepts können auf HTML-Dokumente abgebildet werden.

- Analogie Kanten – Links:
  Die Beziehungen des Rahmenbetriebskonzepts werden durch die Links in den

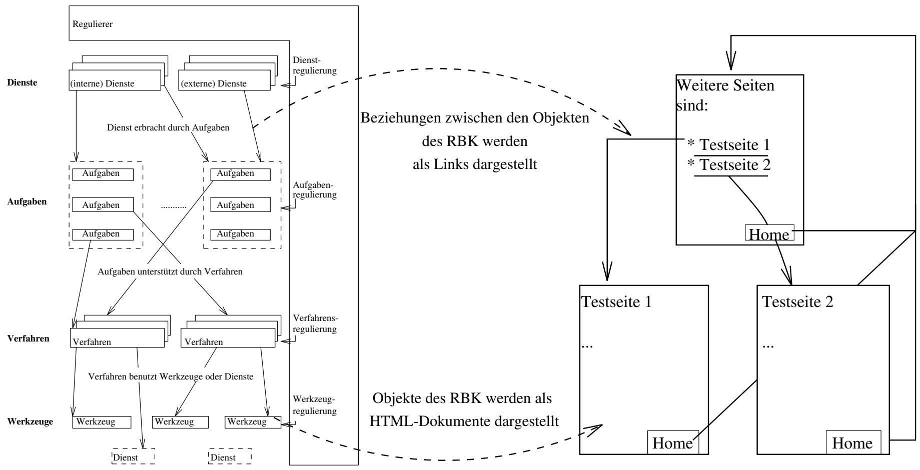

Abbildung 6.6: Analogie von Rahmenbetriebskonzept und HyperText-System

HTML-Dokumenten abgebildet.

- Schnittstelle zur Verarbeitung von verschiedenen Datenformaten:
  Wie bereits in dem allgemeinen Teil über HyperText-Systeme beschrieben, können verschiedenste Datenformate über eine Schnittstelle eingelesen werden; eine Visualisierung wird dadurch möglich. Diese Schnittstelle ist bereits für Standardformate wie Postscript, GIF, ASCII, etc. implementiert.

6.2.3 Nachteile eines HyperText-Systems

Nachteile sind:

- Keine Klassifikation von Links vorgesehen
  Nur durch eine Erweiterung von HTML 2.0 erreichbar

- Keine Echtzeit-Bearbeitung möglich:
  Der Datenbestand kann nie den zur Zeit herrschenden Ist-Zustand repräsentieren, da eine Aktualisierung schwierig ist.

- Keine Transaktionen definiert

6.3 Der HTML-Datenbestand

6.3.1 Templates zum Rahmenbetriebskonzept

In der folgenden Tabelle sind die Templates dargestellt, die als Grundlage der Testdatenbestände dienen. Es wird nach verschiedenen Objekttypen (Aufgabe, Verfahren, Aktion, Funktion und Werkzeug) unterschieden; innerhalb der Objekte wird strikt zwischen Attributen und Beziehungen getrennt.

Objekttyp	Attribut	Beziehungen
Aufgabe:	Bezeichnung Beschreibung Typ Admin_info Rolle, Zuständigkeit	Verfahren [optional] (Teil-)Aufgabe [optional] (Vater-)Aufgabe
Verfahren:	Bezeichnung Beschreibung Typ Admin_info Ablauf	Aufgabe (Teil-)Verfahren Aktion (Vater-)Verfahren
Aktion: (elementarer Arbeitsschritt)	Bezeichnung Beschreibung Funktion Information	Werkzeugfunktion
Funktion:	Bezeichnung Beschreibung Hilfe	Werkzeug
Werkzeug:	Bezeichnung Beschreibung Administrator Hotline Rechner	

In der Spalte *Attribut* stehen die Informationen, die in den Rumpf der standardisierten HTML-Datei eingefügt werden, während in der Spalte *Beziehungen* die zulässigen Linktypen zu erkennen sind.

Im Anschluß folgt ein Template für die HTML-Dokumente.

6.3.2 Aufbau eines HTML-Dokumentes

- <! DOCTYPE HTML PUBLIC "-/ /W30/ /DTD W3 HTML 2.0/ /EN">
- <! Filename: "Filename" >
- <HTML>
- <HEAD>
- <META http-equiv="Owner" content="Author: Name des Autors" >
- <META http-equiv="Reply-To" content="email-Adresse des Autors" >
- <META http-equiv="Last-Modification" content="Letzte Änderungen" >
- <META http-equiv="Node-Type" content="Typ des Dokumentes" >
- <META http-equiv="Belongs-To" content="Name des Datenbestands" >
- <TITLE> string </TITLE>
- </HEAD>
- <BODY>
- <H1> Kurzbeschreibung, Inhalt des Dokumentes </H1>
- Weiterer Inhalt, Links.
- </BODY>
- </HTML>
- Links, d.h. Verbindungen zu anderen Dokumenten, werden wie folgt beschrieben:

  <A REL="relationen_type" HREF="http://sunhegering6.informatik.
  tu-muenchen.de/Datenbestand-Directory/filename.html"
  >unterstrichener_Teil</A>

Die in den META-Zeilen gespeicherten Informationen werden benötigt, um die Konsistenz des Datenbestands zu gewährleisten. Es existiert ein Werkzeug mit Namen "MOMspider" (**M**ulti-**O**wner **M**aintenance **spider**, [lib94]), das einen HTML-Datenbestand z.B. nach Links, die ins Leere zeigen, durchsucht. Ist der Datenbestand abgearbeitet, wird dem Eigentümer ("Owner") eine email („email-Adresse") geschickt, die das Ergebnis der Untersuchung beinhaltet. Diese email kann auch an den Administrator des betreffenden Datenbestands weitergeleitet werden.

In der „Node-Type“-Zeile ist der Typ des Rahmenbetriebskonzept-Objekts einzutragen.

Objekttyp:	Einzutragender Node Type:
Dienst	Service
Aufgabe	Task
Verfahren	Procedure
Aktion	Action
Werkzeug	Tool
Funktion	Function

Zwischen Teilaufgaben und Aufgaben bzw. Teilverfahren und Verfahren wird keine Unterscheidung vorgenommen, d.h. das Attribut Node Type identifiziert die einzelnen Ebenen des Rahmenbetriebskonzepts.

Um die Relationen abbilden zu können, wurde das REL-Attribut eingeführt, welches sich aus drei Teilen zusammensetzt.

1. Der erste Teil gibt an, von welchem Dokumenttyp der Link ausgeht (`Task, aTask, Proce, aProce, Actio, Tool, Funct`). Er besteht aus einem String mit max. 5 Zeichen, die die Ebene bezeichnen. Die kleinen Buchstaben bezeichnen die weiteren Untergliederungen von z.B.Aufgabe (Task) zu einer Teilaufgabe (aTask).

2. Dieser erste Teil wird gefolgt von einer `2`. Sie stellt den Delimiter zwischen der rechten und der linken Seite des REL-Strings dar.

3. Der dritte Teil gibt an, auf welchen Typ von Dokument der Link zeigt (`Task, aTask, Proce, aProce, Actio, Tool, Funct`). Er ist genauso strukturiert wie der erste Teil.

Beispiel für eine solche Relation ist: `Task2aTask`

6.4 Das HTTP-WAIS Gateway SFgate

Das *SFgate* ist ein “Common Gateway Interface“CGI-Skript (**C**ommon **G**ateway **I**nterface-Skript). Die CGI-Skripten sind Skripten, die Anfragen eines WWW-Clients an einen WWW-Server realisieren (vgl. [Klu94b] und [Klu94a]). Das SFgate sitzt auf der Seite des WWW-Servers, nimmt die Anfrage des WWW-Clients entgegen und leiten sie an einen WAIS-Server (**W**ide **A**rea **I**nformation **S**ystem-Server) weiter.

Der WAIS-Server bearbeitet die Anfrage und schickt eine Quittung an den WWW-Server, der als WAIS-Client auftritt, zurück. In unserem Fall wird über das SFgate eine Suchanfrage an einen WAIS-Server gestellt. Dieser WAIS-Server verwaltet eine Menge von Indexen, über die effizient gesucht werden kann. Als Ergebnis liefert er alle Dokumente, die dieser Suchanfrage genügen.

WAIS ist ein Werkzeug, das eine inhaltsorientierte Suche gestattet. Unter anderem wird das WAIS-System auch im Internet verwendet, um aus dem reichhaltigem Datenangebot die relevanten Informationen zu extrahieren. In Terra wurde es verwendet, um Strukturinformationen abzuspeichern, die in den HTML-Dokumenten in Form von Links enthalten sind. Diese Strukturinformationen werden zur Viewbildung herangezogen. WAIS ist ein bereits fertiges Public-Domain-Softwareprodukt, das von der Firma Thinking Machines seit 1989 immer weiter entwickelt wurde. Es bietet einige Vorteile anderen Systemen gegenüber. WAIS basiert auf einer Client/Server-Architektur, die das zustandslose Protokoll ANSI z3950 verwendet. Darüber hinaus wird nicht das allgemein übliche Modell der boolschen Suche, sondern das Vektorraummodell unterstützt. Beim boolschen Ansatz wird die Eingabe durch eine formale Anfragesprache beschrieben. Das System berechnet die Dokumente der ausgewählten Datenbank, auf die das Suchkriterium zutrifft. Der boolsche Ansatz hat zwei entscheidende Nachteile:

1. Der Benutzer muß sich in eine komplexe Befehlssprache einarbeiten, die sich von Datenbank zu Datenbank unterscheiden kann.

2. Die Größe der Antwortmenge läßt sich nur schwer abschätzen.

Das Vektorraummodell wurde Ende der 60er Jahre von Gerard Salton an der Universität Harvard entwickelt ([Pfe95]). Das Modell gestattet dem Benutzer, Anfragen in „natürlicher“ Sprache an das System zu stellen. Zudem läßt sich auch das Problem der Antwortgröße elegant lösen.

Bei diesem Modell versucht man, sowohl die Anfrage des Benutzers als auch die Dokumente einer Datenbank auf sog. Feature-Vektoren abzubilden. Features korrospondieren meist mit Schlüsselworten innerhalb eines Textes. Der Anfrageprozeß vergleicht den Vektor der Abfrage mit den Vektoren der verschiedenen Dokumente. Die Ähnlichkeit eines Anfragevektors und eines Dokumentenvektors ergibt sich im einfachsten Fall als das Produkt der beiden Vektoren. Aus der mathematischen Sicht stellt sich dieses Produkt als das Skalarprodukt der beiden Vektoren dar. Die unterschiedlichen Dokumente können noch anschließend nach ihrer Ähnlichkeit zu der ursprünglichen Anfrage sortiert werden (Ranking).

Beispiel einer Suche anhand des Vektorraummodells Als Beispiel soll eine Datenbank dienen, die aus zwei Dokumenten besteht. Die beiden Dokumente sind mit den Features „Information“ und „Retrieval“ belegt worden. Es entstehen also Vektoren mit zwei Komponenten, wobei die erste Komponente die Häufigkeit des Features „Information" und die zweite Komponente die Häufigkeit des Features „Retrieval" darstellt. Die Vektoren lauten $d_1 = (2,3)$ und $d_2 = (0,2)$. Die Suchanfrage lautet „Information Retrieval", der dazugehörige Vektor $q = (1,1)$. Legt man das Skalarprodukt als Metrik zugrunde, so erhält man die Ähnlichkeit $sim(q, d_1) = q * d_1 = 5$ und $sim(q, d_2) = q * d_2 = 2$. d_1 wäre in dieser sehr einfachen Metrik die bessere Antwort und würde natürlich im Ranking vor d_2 liegen.

6.5 Der Server-Dämon für HTTP

Bei HTTP ([BL94]) handelt es sich um ein Protokoll, das bei verteilten, aus unterschiedlichen Bestandteilen zusammengesetzten, Hypermedia-Informationssystemen zum Einsatz kommt; daher wurde besonderes Augenmerk auf die Schnelligkeit der Übertragung gelegt. HTTP ist ein zustandsloses Protokoll und beruht auf einem objektorientierten Ansatz.

Eine Transaktion besteht aus einer

Connection	Es wird eine TCP/IP-Verbindung defaultmäßig zum TCP/IP Port Nummer 80 aufgebaut. Diese kann durch die Angabe eines URL aber auch auf einen anderen Port umgelenkt werden.
Request	Der Client schickt eine Anfrage an den Server.
Response	Der Server schickt eine Antwort an den Client.
Close	Die TCP/IP-Verbindung wird durch den Client oder den Server beendet.

HTTP stellt die Methoden für die Requests an den Server zur Verfügung. Nachfolgend werden die wichtigsten Methoden kurz skizziert, die restlichen sind in [BL94] aufgeführt und erläutert.

GET-Methode: Diese Methode schickt ein Objekt – es kann sich um ein HTML-Dokument oder ein auszuführendes Skript handeln – an den Client zurück. Dieses Objekt wird durch eine URI exakt spezifiziert.

POST-Methode: Der Client legt durch diese Methode ein neues Objekt an, der Server speichert es ab und liefert dem Client die zugewiesene URL als Response zurück.

PUT-Methode: Die PUT-Methode ist vergleichbar mit der POST-Methode, es können aber keine neuen Objekte angelegt werden. Daher eignet sich diese Methode gut, um bereits bestehende Objekte zu aktualisieren, da sich der Datenbestand nicht unkontrollierbar vergrößert.

6.6 TkWWW-HyperText-System

Für das Werkzeug wurde das TkWWW-HyperText-System ausgewählt. Gegenüber Mosaic oder anderen Systemen besitzt es den Vorteil einer Schnittstelle zu der Sprache Tcl/Tk (**T**ool **C**ommand **L**anguage/**T**ool **k**it, vgl. [Ous94]). Tcl/Tk ist eine Skriptsprache, die es erlaubt, X-Window-Applikationen ohne größeren Aufwand zu programmieren. Dem Programmierer wird durch Tcl ein Programmrahmen vorgegeben, der die Arbeit im Vergleich zur herkömmlichen X-Window-Programmierung wesentlich vereinfacht. Nach einer relativ kurzen Einarbeitungszeit können komplexere X-Window-Applikation geschrieben werden.

Das TkWWW-System ist vollständig in Tcl programmiert und stellt den WWW-Browser „TkWWW“ zur Verfügung. Er übernimmt die Aufgabe, Informationen von Servern, die an das WWW angeschlossen sind, aufzubereiten und darzustellen.

6.6.1 Der modifizierte TkWWW-Browser

Im ersten Prototyp des Werkzeugs Terra wurde der WWW-Browser TkWWW verwendet. Er bietet gegenüber dem wohl am meisten verwendeten WWW-Browser “Mosaic“ den Vorteil der Tcl/Tk-Schnittstelle. Tcl/Tk besteht aus zwei Teilen:

- Tcl:
  Mit Tcl können Anweisungen programmiert werden. Zur Verfügung stehen hier eine Vielzahl von Konstrukten, wie sie auch z.B. C bereitstellt.
- Tk:
  Tk besteht aus einer Sammlung von sog. Widgets. In diesen Widgets sind bereits vordefinierte Komponenten enthalten, die zum Aufbau einer X-Window-Oberfläche benötigt werden, z.B. Knöpfe, Eingabefelder, Scrollbars etc.

Neben dem Vorteil einer leichten Programmierung der X-Window-Oberfläche ist die Prozeßkommunikation zwischen Tcl/Tk-Applikationen verhältnismäßig einfach zu bewerkstelligen. Tcl/Tk stellt ein Kommando „send“ zur Verfügung, das X-Events für die Interprozeßkommunikation verwendet.

6.7 Das Graphvisualisierungswerkzeug *daVinci*

DaVinci ist ein an der Universität Bremen entwickeltes Werkzeug ([FW94]) zur

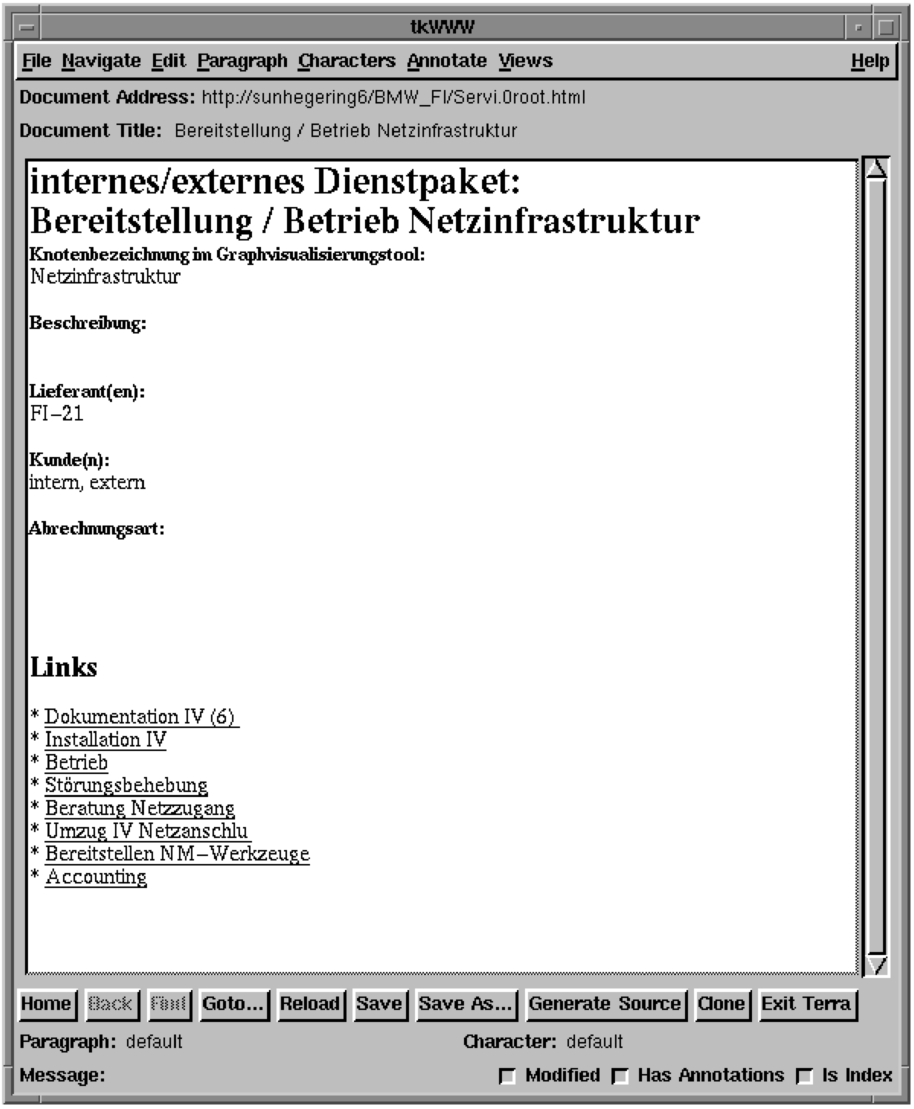

Abbildung 6.7: Modifizierter TkWWW-Browser

Darstellung von gerichteten Graphen. Es ist in der funktionalen Sprache ASpecT (**A**lgebraic **Spec**ification **T**ransformer; vgl. [vHS93], ebenfalls an der Universität Bremen entwickelt) geschrieben und unterstützt eine ASpecT-Benutzerschnittstelle. Über diese kann eine Applikation an daVinci angebunden werden. Diese Schnittstelle wird im Rahmen des Projekts Terra benutzt, um daVinci darzustellende Graphen in ASpecT-Notation zu schicken. Diese Graphen repräsentieren Views auf den in Abschnitt 6.3.2 gemachten Konventionen unterliegenden HTML-Datenbestand, die durch eine Anfrage an die WAIS-Datenbank generiert werden. Die Graphen werden möglichst optimal dargestellt, d.h. die Anzahl der Kantenkreuzungen wird durch den Algorithmus von Kozo Sugiyama ([SM91], [STT81]) minimiert.

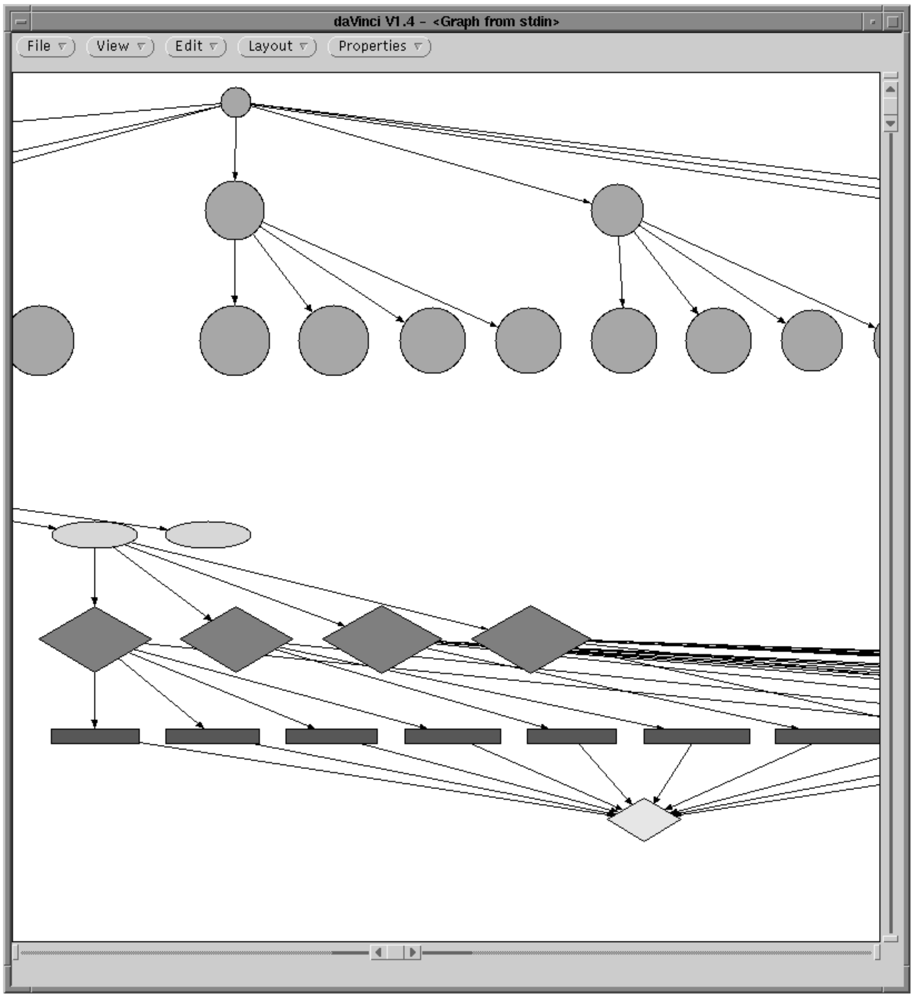

Abbildung 6.8: Darstellung in daVinci

daVinci bietet dem Anwender verschiedene Funktionen an, auf die zunächst nur stichpunktartig eingegangen wird; wichtige Punkte werden später gesondert betrachtet.

- Das Menü *File*:
  In diesem Menü sind alle Funktionen zusammengefaßt, die
  - das Laden und Speichern von Graphen und deren Status (unter Status

wird hier ein vom Anwender erstelltes Layout eines Graphen verstanden)
  - das Speichern des aktuell dargestellten Graphen als Postscript-Datei und
  - das geregelte Verlassen von daVinci ermöglichen.

  Zusätzlich kann in diesem Menü eine Applikation in daVinci eingebunden werden.

- Das Menü *View*:
  Hier werden die Funktionen zusammengefaßt, die mit der Präsentation des Graphen zusammenhängen. Unter anderem stehen hier die Funktionen
  - Skalieren,
  - allgemeine Informationen über den Graphen, z.B. die Anzahl der Knoten oder der Kantenkreuzungen und
  - allgemeine Informationen über daVinci selbst

  zur Verfügung.

- Das Menü *Layout*:
  Mit den folgenden Funktionen kann der Anwender das Layout das Graphen beeinflussen:
  - durch Ausblenden von Teilgraphen und Kanten und
  - durch Neupositionierung aller Knoten, wodurch sich evt. die Kantenkreuzungen minimieren lassen.

- Das Menü *Properties*:
  Die Funktionen, die dieses Menü anbietet, beschäftigen sich mit grundlegenden Eigenschaften des gezeigten Graphen, z.B. wie groß der Abstand eines Knotens zum nächsten Knoten in der Horizontalen und Vertikalen sein darf, oder bis zu welcher Rekursionstiefe der Layout-Algorithmus zu berechnen ist. Je größer die Rekursionstiefe ist, umso weniger Kantenkreuzungen treten auf, aber die benötigte Rechenzeit steigt überproportional an. Anwendungen, die durch daVinci angesprochen werden sollen, werden in diesem Menü spezifiziert.

6.7.1 Das Menü *Edit*

Das Menü *Edit* nimmt eine Sonderstellung in daVinci ein, da nur dieses Menü vom Anwender manipuliert werden kann. Um mit daVinci über dessen API zu kommunizieren, muß eine Applikation angebunden werden, die die Kommunikationskanäle initialisiert (zur Kommunikation werden Unix-Pipes verwendet). Erst jetzt können Kommandos durch die Applikation an daVinci geschickt werden. Hier ein kurzer Auszug, der zeigt, wie ein Menü in daVinci aufzubauen ist:

```
command  ::=  create_menu(menu)
menus    ::=  [menu]
menu     ::=  menu_entry(menu_id,string) |
              submenu_entry(menu_id,string,menus) |
              blank
menu_id  ::=  string
```

Beispiel für ein eingehängtes Menü:

```
create_menu(submenu_entry("Insert_Node","Knoten einfuegen",
[blank,menu_entry("Insert_Node:Service","Dienst"),
menu_entry("Insert_Node:Task","Aufgabe"),
menu_entry("Insert_Node:Procedure","Verfahren"),
menu_entry("Insert_Node:Action","Aktion"),
menu_entry("Insert_Node:Tool","Werkzeug"),
menu_entry("Insert_Node:Function","Funktion")]))
create_menu(submenu_entry("Insert_Child","Kindknoten einfuegen",
[blank,menu_entry("Insert_Child:Service","Dienst"),
menu_entry("Insert_Child:Task","Aufgabe"),
menu_entry("Insert_Child:Procedure","Verfahren"),
menu_entry("Insert_Child:Action","Aktion"),
menu_entry("Insert_Child:Tool","Werkzeug"),
menu_entry("Insert_Child:Function","Funktion")]))
create_menu(menu_entry("Insert_Edge","Beziehung einfuegen"))
create_menu(blank)
create_menu(menu_entry("Delete_Node","Knoten loeschen"))
create_menu(menu_entry("Delete_Edge","Beziehung loeschen"))
create_menu(blank)
create_menu(menu_entry("Show_Attributes","Attribute anzeigen"))
```

Wird ein Knopf des neu eingehängten Menüs „Edit“ betätigt, schickt daVinci über die Pipe an die Applikation eine Meldung, in der mitgeteilt wird, welcher Menüknopf gedrückt wurde. Diese Meldung kann die Applikation auswerten und ggf. anwenderdefinierte Aktionen auslösen.

Darstellungsmöglichkeiten

Die unterschiedlichen Knotentypen, die sich durch die Einteilung der Rahmenbetriebskonzept-Objekte in Objektklassen ergeben, werden durch eindeutige Symbole dargestellt. So werden z.B. Kreise, Rauten, Vierecke, und ab der Version 1.4 auch anwenderdefinierte Formen, unterstützt. Die Anzeige verschiedener Beziehungsklassen erfolgt durch eindeutige Visualisierung der Kanten. DaVinci bietet verschiedene

Stile – gepunktete, gestrichelte oder durchgezogene Linien – für die Kanten an. Die Unterscheidung von Knoten bzw. Kanten kann durch eine unterschiedliche Farbgebung noch erhöht werden.

6.8 Das Steuermodul *Leonardo*

Das Steuermodul hat folgende Aufgaben zu übernehmen:

- Sicherstellung der Kommunikation zum Server,
- Bereitstellen der Schnittstelle zum Navigationswerkzeug und
- Bereitstellung der Schnittstelle zum Browser.

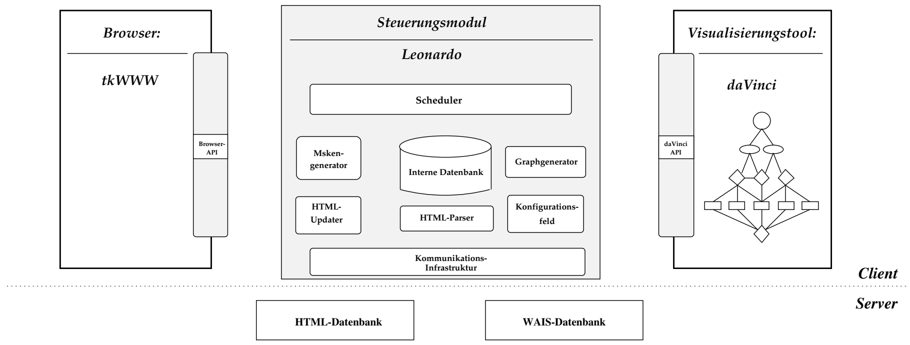

Abbildung 6.9: Schema des Steuermoduls Leonardo

Innerhalb dieses Steuermoduls, das auf den Namen *Leonardo* getauft wurde, existieren verschiedene Module:

- der Scheduler,
- die Kommunikationsinfrastruktur,
- der HTML-Parser,
- die interne Datenbank,
- Konfigurationsfeld,
- der Graphgenerator,

- das HTML-Update-Modul und
- der Eingabemaskengenerator für die Rahmenbetriebskonzept-Objekte.

Die Module, die bis auf den Eingabemaskengenerator in *Perl* (**P**ractical **E**xtraction and **R**eport **L**anguage, vgl. [WS91]) geschrieben sind, werden zuerst isoliert dargestellt und daran anschließend in Form von Szenarien die Wechselbeziehungen zwischen den einzelnen Modulen.

6.8.1 Der Scheduler

Der Scheduler ist die „Schaltzentrale“ des Steuermoduls Leonardo und arbeitet in zwei verschiedenen Phasen.

Initialisierungsphase: In der Initialisierungsphase werden die Kommunikationskanäle installiert, dies wird auf der Client-Seite durch Unix-Pipes realisiert. Die beiden anderen Module, daVinci – das Navigationsmodul – und der TkWWW-Browser – das Anzeigemodul – werden als Sohnprozesse von Leonardo gestartet. Danach tritt der Scheduler in die Event-Handling-Phase ein.

Event-Handling-Phase: In der Event-Handling-Phase überwacht Leonardo die angeschlossenen Pipes auf das Eintreffen von „Events“. Registriert Leonardo einen Event, so wertet ihn der Scheduler aus und startet die entsprechende Aktion.

Aufbau des Schedulers: Der Scheduler stellt den Programmrahmen zur Verfügung; er besteht aus einer Endlosschleife, die die initialisierten Pipes überwacht, und einer Sammlung von Perl-Prozeduren, die die einzelnen Module repräsentieren.

Zugrundeliegender Algorithmus: Nachdem der Scheduler die Kommunikationskanäle etabliert und die beiden Sohnprozesse in der Initialisierungsphase abgespalten hat, tritt er in eine Event-Handling-Phase ein, in der er die einzelnen Unix-Pipes überwacht. Das Verfahren des „busy waiting“ eignet sich für diese Aufgabe nicht, da sonst in der Pipe eintreffende Events „vergessen“ werden, wenn ein anderer Event behandelt wird. Zur Verwendung kommen Unix-Signale, die gesendet werden, wenn in einer Pipe eine Meldung eintrifft; das hat neben dem Performancegewinn auch noch den Vorteil, daß kein Event vergessen werden kann. Für die verschiedenen Pipes werden unterschiedliche Signale eingesetzt, so daß der Scheduler die Pipe, in der ein Event angekommen ist, eindeutig erkennen kann. Die Meldung wird anschließend geparst und ausgewertet. Anhand des Auswertungsergebnisses stößt der Scheduler die zugeordnete Prozedur an. Danach wartet der Scheduler auf den

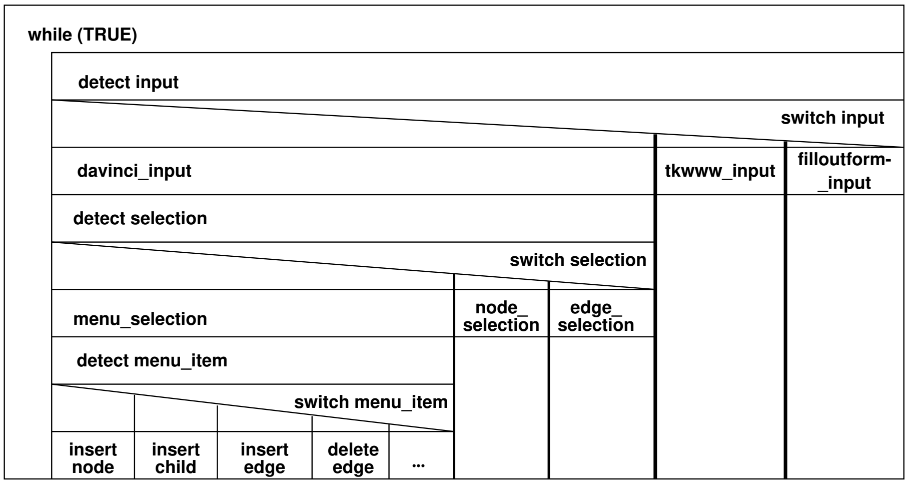

Abbildung 6.10: Implementierung des Schedulers

nächsten eintreffenden Event. Ein kleiner Ausschnitt der Arbeitsweise des Schedulers findet sich in Abbildung 6.10.

6.8.2 Die Kommunikationsinfrastruktur

Die Kommunikationsinfrastruktur besteht aus einer Sammlung von Perl-Routinen (vgl. [WS91]) des Public-Domain-Softwarepakets [lib94]. Dieses Paket stellt „GET"- und „PUT"-Anweisungen zur Verfügung, die an den HTTP-Dämon geschickt werden, der diese an den Server weiterreicht.

- „GET" führt eine Leseoperation auf dem HTML-Datenbestand aus. Als Ergebnis liefert es das über die URL (=Uniform Resource Locator) angeforderte HTML-Dokument.
- „PUT" schreibt ein HTML-Dokument in den HTML-Datenbestand auf der Serverseite. Das Ziel wird wieder in Form einer URL angegeben.

6.8.3 Der HTML-Parser

Der Scheduler stellt über das Severscript *SFgate* an den WAIS-Server eine Anfrage, welche über die SFgate-eigene Methode „direct_get" ausgeführt wird. Der WAIS-Server wertet die Anfrage aus und liefert daraufhin über die Kommunikationsschnittstelle alle Dokumente, die der Anfrage entsprechen, an den Client (Leonardo) zurück. Der HTML-Parser verwendet diese HTML-Dokumente als Eingabe und extrahiert daraus den Titel des Dokuments und die Links. Aus diesen Angaben wird die Interne Datenbank aufgebaut.

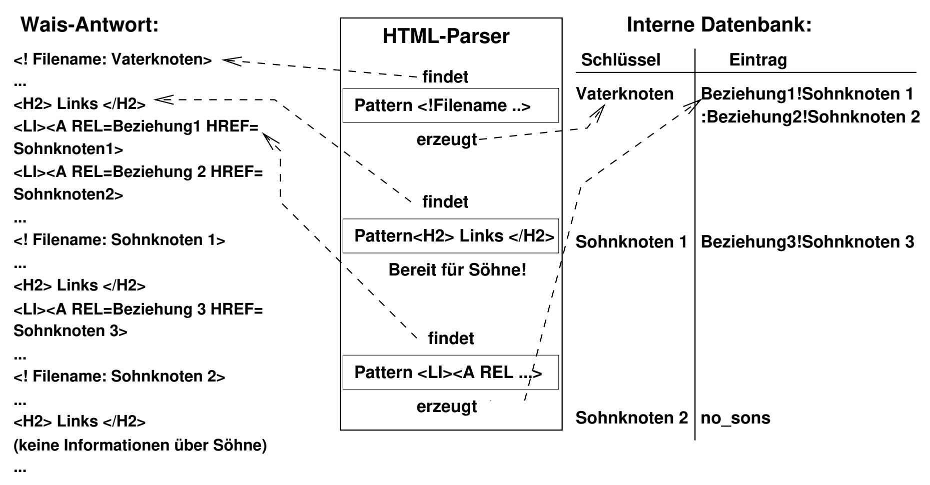

Abbildung 6.11: Parsen und Aufbau der Internen Datenbank

6.8.4 Die Interne Datenbank

Die Interne Datenbank stellt die Schnittstelle zu daVinci dar und enthält eine Liste von Vätern, den dazugehörigen Söhnen und den Beziehungen. Aus dem einheitlichen Datenformat der Internen Datenbank kann ein ASpecT-Graphenterm generiert werden, so daß daVinci diesen Graphen anzeigen kann.

Datenformat der internen Datenbasis: Die Datenbank besteht aus einer Hash-Tabelle, über die eine effiziente Suche in der Datenbank möglich ist. Als Schlüssel für die Tabelle dient der Name des Vaterobjekts.

Vaterobjekt	Beziehung_1:Sohn_1!...!Beziehung_n:Sohn_m
Steuernde Aufgabe	Steuernde Aufgabe nach Ausführungs-Aufgabe: Aufgabe_1! Steuernde Aufgabe nach Basisaufgabe: Basisaufgabe_1! ...! Steuernde Aufgabe nach Ausführungs-Aufgabe: Aufgabe_m
Verfahren	Verfahren nach Teilverfahren:Teilverfahren_1! ... Verfahren nach Teilverfahren:Teilverfahren_n

Die Interne Datenbank wird durch den HTML-Parser als „Associated Array" aufgebaut. Als Schlüssel werden die Väterknoten angesehen, über welche der Index der Hash-Tabelle angelegt wird. Jeder Knoten tritt einmal als Vaterknoten auf; hat ein Knoten keine Referenz (Link) auf einen Sohnknoten, so wird in die Interne Datenbank anstelle einer Referenz *no_sons* eingetragen. Ein Auszug aus dieser Datenbank sieht wie folgt aus:

Vaterknoten (Schlüssel)	Beziehung:Sohnknoten
Task.Hub_in_Use	Task2Proc:Proc.Hub_in_Use

6.8.5 Das Konfigurationsfeld

Das Konfigurationsfeld ist ein „Associated Array", in dem alle zulässigen Beziehungen und Objekttypen stehen. Die für die Graphdarstellung notwendigen Attribute sind enemfalls hier abgespeichert.

6.8.6 Der Graphgenerator

Der Graphgenerator ist für den Aufbau des Graphen aus der Internen Datenbank zuständig und orientiert sich am verwendeten Graphvisualisierungswerkzeug. Die von daVinci für den Aufbau von Graphen verwendeten ASpecT-Terme sind stark geklammerte Ausdrücke, die beim Parsen große Probleme aufwerfen.

```
                                                     | Definition eines/einer:
graphtrees  ::= [graphtree]                          |
graphtree   ::= n(string,attributes,edge) |          | node
                l(string,graphtree) |                | labeled node
                r(string)                            | referenced node
edges       ::= [edge]                               |
edge        ::= e(string,attributes,graphtree) |     | edge
                l(string,edge)                       | labeled edge
attributes  ::= [attribut]                           |
attribut    ::= a(string,string)                     | attributes
```

Mit `a` für „attributes“ können einem Graphobjekt (Knoten oder Kante) Attribute wie die Form oder Farbe mit übergeben werden.

```
l(''node_b'',n('''',[a(''OBJECT'',''node_b''),a(''_GO'',''circle'')],
[l(''node_b2node_d'',e('''',[a(''_DIR'',''normal'')],r(''node_d'')))]))
```

Algorithmus-Idee: Das Problem, einen Graphen aufzubauen, ist sehr komplex und erfordert meist Backtracking-Algorithmen, die eine hohe Zeit- und auch Speicherkomplexität aufweisen. Dies wird durch die Verwendung von ASpecT-Termen noch zusätzlich erschwert. Ein Trick löst das Problem auf einfache Weise. Der Graphgenerator erzeugt „einelementige Teilbäume“, die über Referenzen verpointert werden. Diese Teilbäume werden aneinandergereiht und an daVinci geschickt, das sich aus diesen Teilbäumen selbständig den Graphen aufbaut. In den Algorithmus muß keine „Intelligenz“ gesteckt werden, die die Positionierung der Knoten vornimmt.

Der Graphgenerator filtert Beziehungen des Linktyps „BackLink“ heraus, da sonst die Ebenendarstellung des Rahmenbetriebskonzepts nicht gewährleistet werden kann. Treten Zyklen in einem Graphen auf, d.h. ein Knoten ist sowohl Vater- als auch Sohnknoten eines anderen Knotens, kann es vorkommen, daß daVinci die Knoten nicht in der richtigen Rahmenbetriebskonzept-Ebene anordnet, wenn sich dadurch die Kantenkreuzungen minimieren lassen.

Algorithmus zum Generieren des Graphenterms: Ausgegangen wird von einem folgendermaßen aussehendem Term:
`l(''nodelabel'',n('''',[attributpart_n],[childpart]))`
Als `nodelabel` wird der Dateiname des entsprechenden Dokuments verwendet, der aus der *Internen Datenbank* extrahiert wird. Der Dateiname enthält auch den jeweiligen Knotentyp, so daß die im *Konfigurationsfeld* enthaltenen Attribute in den String eingesetzt werden können. Sind Referenzen auf andere Knoten vorhanden, wird der Teil `childpart` generiert. Anderenfalls wird `childpart` durch einen leeren

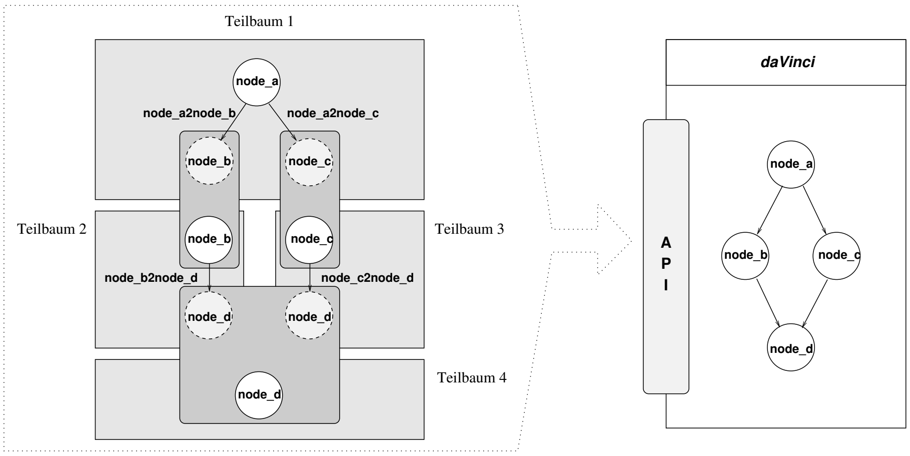

Abbildung 6.12: Algorithmus-Idee für den Graphgenerator

String ersetzt. Der neu generierte Teilbaum wird an den bisher erstellten Graphen angehängt und der Ablauf für jeden Eintrag der Internen Datenbank wiederholt.

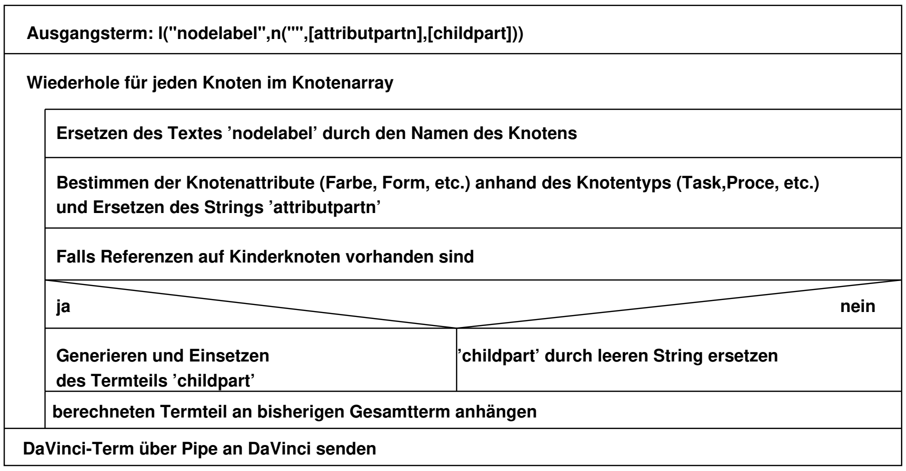

Abbildung 6.13: Implementierung des Graphgenerators

Ist der Term vollständig generiert, wird er über die durch den Scheduler initialisierte Pipe-Verbindung an daVinci zur Darstellung geschickt.

6.8.7 Das HTML-Update-Modul

Das HTML-Update-Modul trägt HTML-Links in die Dokumente ein bzw. löscht aus einem Dokument Links heraus.

Einfügen eines Links: Das Modul wird vom Scheduler angestoßen und erhält als Argument das Label des HTML-Dokuments, in das ein Link eingefügt werden soll

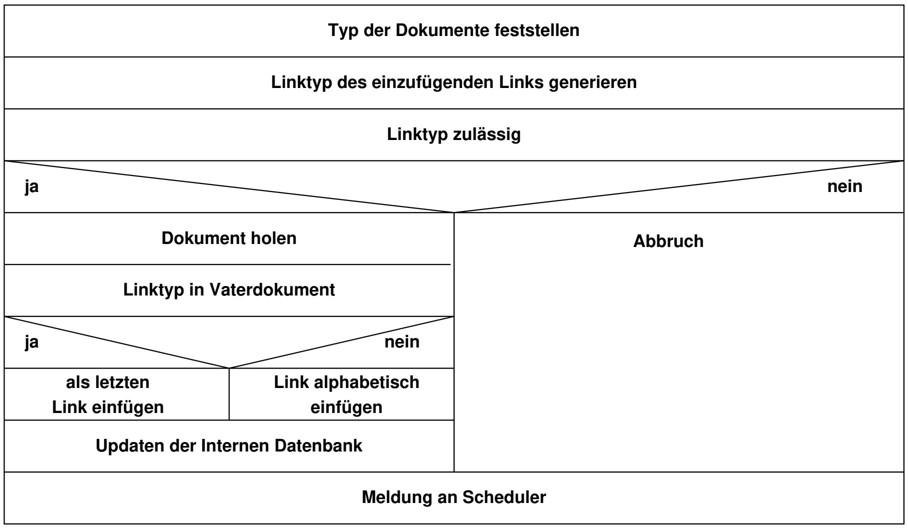

Abbildung 6.14: Implementierung des HTML-Update-Moduls: Einfügen eines Links

und das des dazugehörigen „Sohndokuments". Anhand der Labels generiert das Modul den Beziehungstyp, der zwischen den Objekten eingefügt werden soll. Anhand der im Konfigurationsfeld abgespeicherten zulässigen Beziehungstypen kann das Modul mit einem einfachen Vergleich feststellen, ob die gewünschte Beziehung eingefügt werden darf. Ist der Vergleich negativ, wird mit einer Fehlermeldung abgebrochen, die vom Scheduler ausgegeben wird; ist der Vergleich aber positiv, werden mit einer „GET"-Anweisung „Vater-" und „Sohndokument" aus dem Server in den Client geladen. Das Vaterdokument wird nach Beziehungen gleichen Typs durchsucht und alphabetisch eingeordnet; wird kein gleichartiger Beziehungstyp gefunden, wird die Beziehung als letzter Link eingefügt. Im Sohnknoten wird automatisch ein Link vom Typ „BackLink" auf den Vaterknoten eingefügt. Die entstandenen Änderungen sind in die Interne Datenbank einzutragen, danach ist der Scheduler zu unterrichten.

Löschen eines Links: Nachdem das HTML-Update-Modul den Auftrag zum Löschen eines Links vom Scheduler erhalten hat, werden die HTML-Dokumente (Vater- und Sohndokument) über die Kommunikationsinfrastruktur mit einer

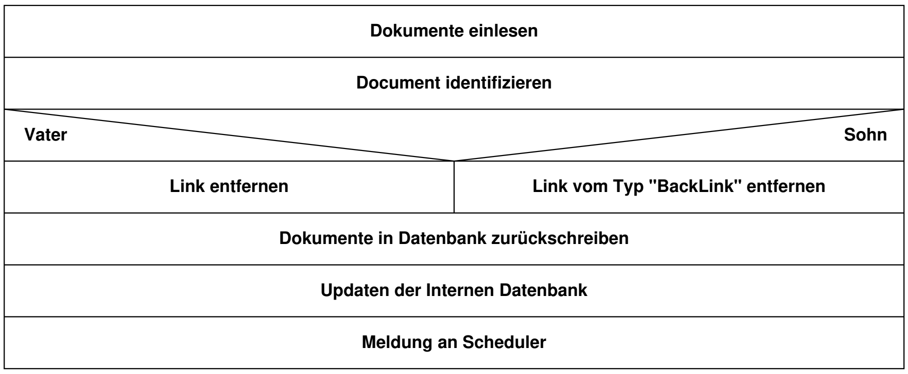

Abbildung 6.15: Implementierung des HTML-Update-Moduls: Löschen eines Links

„GET"-Anweisung geholt. Das Dokument wird geparst und der entsprechende Link gelöscht. Aus dem „Sohndokument" wird der entsprechende Eintrag mit dem Linktyp „BackLink" entfernt. Anschließend werden die modifizierten Dokumente mit einer „PUT"-Anweisung wieder in die HTML-Datenbank zurückgeschrieben. Die Interne Datenbank, die die Schnittstelle zum Navigationsmodul daVinci darstellt, wird aktualisiert, indem aus dem Eintrag des Vaterknotens der entsprechende Sohneintrag gelöscht wird.

6.8.8 Der Eingabemaskengenerator für die Rahmenbetriebskonzept-Objekte

Dieses Modul ist im Gegensatz zu den anderen Modulen von Leonardo nicht in der Scriptsprache *Perl*, sondern in der Scriptsprache *Tcl/Tk* geschrieben worden, da diese das Schreiben von X-Window Applikationen sehr gut unterstützt. Den Eingabemasken liegen die in Abschnitt 6.3.1 vorgestellten Templates zugrunde, d.h. nur Attribute werden durch die Eingabemaske erfaßt, da die Links in daVinci einzugeben sind. Es existiert für jeden Knotentyp eine Eingabemaske, die aus den durch den Anwender eingegebenen Daten ein standardisiertes HTML-Dokument erstellt, das den Konventionen von Abschnitt 6.3.2 entspricht.

Dieses Modul wird als Sohnprozeß des Moduls Leonardo gestartet, wobei die Kommunikation über Unix-Pipes realisiert ist. Während eine Eingabemaske aktiv ist,

Abbildung 6.16: Eingabemaske für einen Dienst

wird daVinci für alle Eingaben des Anwenders gesperrt, damit es keine Kollisionen mit anderen anwenderausgelösten Aktionen geben kann. Wird die Eingabemaske geschlossen, wird eine Meldung an den Scheduler geschickt, der erkennt, ob die Eingabemaske ausgefüllt wurde und ggf. eine „PUT“-Operation anstößt, damit die erzeugte HTML-Datei in die HTML-Datenbank geschrieben wird. Außerdem nimmt der Scheduler eine Aktualisierung der Internen Datenbank vor und fügt in sie einen neuen Knoten ein.

6.9 Verdeutlichung des Zusammenspiels der Module in Form von Szenarien

Anhand von zwei Szenarien wird beispielhaft das Zusammenwirken der verschiedenen Bestandteile des Steuermoduls Leonardo aufgezeigt. Zu diesem Zweck wurde der Aufbau eines Graphen und das Einfügen einer Beziehung ausgewählt. Weitere

Szenarien sind in [FBE+94] enthalten.

6.9.1 Aufbau eines Graphen

Durch das Menü „Views“ im WWW-Browser TkWWW wird ein View spezifiziert, der im Graphvisualisierungswerkzeug daVinci dargestellt werden soll. Diese Anforderung wird über die Browser-API an den Scheduler innerhalb Leonardos geschickt (**Schritt 1** in Abbildung 6.17). Der Scheduler stellt die Anfrage über die Kommuni-

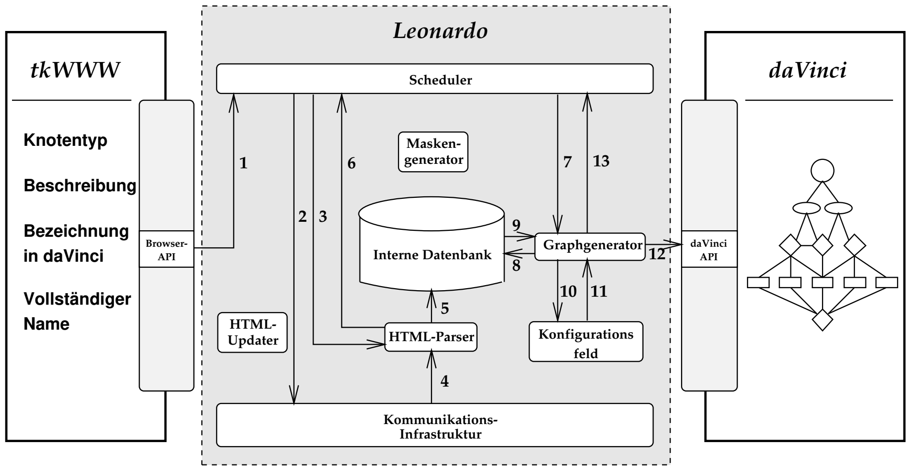

Abbildung 6.17: Szenario: Aufbau eines Graphen in Terra

kationsinfrastruktur an das SFgate (**Schritt 2**), welches die Anfrage bearbeitet und das Ergebnis wieder über die Kommunikationsinfrastruktur an den HTML-Parser weiterreicht (**Schritt 4**). Während die Anfrage vom Server bearbeitet wird, wird der HTML-Parser durch den Scheduler aktiviert und wartet auf die Antwort der Anfrage (**Schritt 3**). Trifft die Antwort ein, so wird diese geparst, die Interne Datenbank generiert und eine Meldung an den Scheduler geschickt (**Schritt 5** und **6**). Daran anschließend wird der Graphgenerator vom Scheduler aktiviert (**Schritt 7**), der die Daten der Internen Datenbank und des Konfigurationsfelds ausliest, daraus einen ASpecT-Term aufbaut und diesen daVinci über seine Schnittstelle schickt (**Schritt 8** bis **13**).

6.9.2 Einfügen einer Beziehung

Im Graphvisualisierungswerkzeug wird zunächst der Vater- und anschließend der Sohnknoten mit der Maus markiert. Durch das Auswählen des Menüpunkts „Kante einfügen“ im Menü „Edit“ wird die auszuführende Aktion spezifiziert. Eine Meldung,

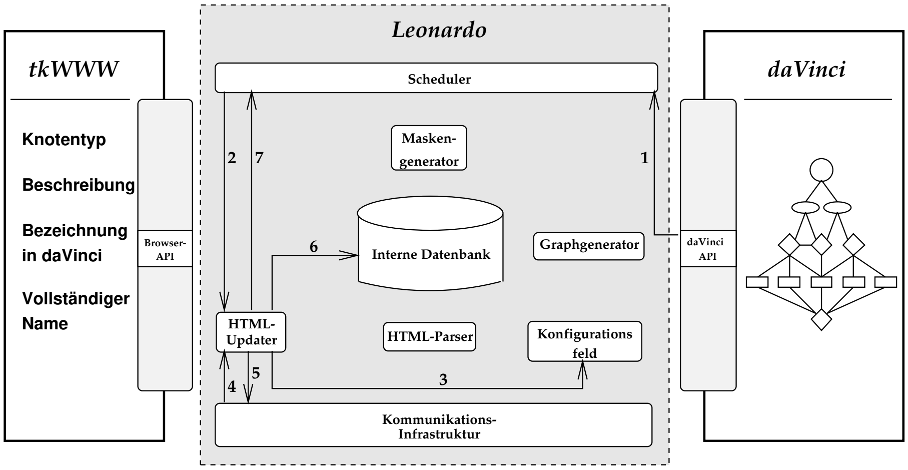

Abbildung 6.18: Szenario: Einfügen einer Beziehung

die diese Aktionen beinhaltet, wird über die daVinci-API an den Scheduler geschickt (**Schritt 1** in Abbildung 6.18), der daraufhin das HTML-Update-Modul aktiviert (**Schritt 2**). Dieses ermittelt den Typ der einzufügenden Kante und überprüft anhand des Konfigurationsfelds, ob dieser Typ überhaupt zulässig ist (**Schritt 3**). Ist diese Prüfung positiv verlaufen, werden beide HTML-Dokumente über die Kommunikationsinfrastruktur geladen und geparst (**Schritt 4**). In das Vaterdokument wird die gewünschte Referenz und in das Sohndokument ein Link vom Typ „BackLink“ eingetragen. Anschließend werden beide Dokumente wieder über die Kommunikationsinfrastruktur in die HTML-Datenbank zurückgeschrieben (**Schritt 5**). Zuletzt wird die Interne Datenbank noch aktualisiert (**Schritt 6**) und eine Meldung an den Scheduler abgesetzt (**Schritt 7**).

Kapitel 7

Diskussion über die technische Umsetzung der Anforderungen an das Werkzeug

In Kapitel 6 wurde ein erster Lösungsversuch unternommen, den Erfasser bei der Erfassung von Daten gemäß dem Rahmenbetriebskonzept zu unterstützen. Die Erprobung des Werkzeugs *Terra* durch unabhängige Personen förderte zusätzliche Erkenntnisse zutage, die gemäß den in Kapitel 4 und 5 aufgestellten Anforderungen klassifiziert und im Anschluß diskutiert werden. Das Ergebnis dieser Diskussion ist eine Aufstellung über die erfüllten und nicht erfüllten Anforderungen und dient als Anforderungskatalog für zukünftige Terra-Versionen.

7.1 Diskussion der Anforderungen an das Anzeige- und Editiermodul

Diese Funktionen wurden in Abschnitt 5.3.3 als ein Modul behandelt, in der praktischen Umsetzung jedoch in zwei verschiedene Programmodule aufgeteilt:

- zum Anzeigen der WWW-Browser TkWWW und
- zum Erfassen der Daten, ein vom Autor entwickelter Maskengenerator.

7.1.1 Das Anzeigemodul TkWWW

Als Basissystem wird das TkWWW-HyperText-System verwendet, um so Entwicklungszeit zu sparen. Theoretisch eignet sich jeder WWW-Browser, der über eine

Kommunikationsschnittstelle verfügt, für diese Aufgabe. Der TkWWW-Browser bietet eine Tcl/Tk-Programmierschnittstelle an, über die zusätzliche Funktionalität eingebracht werden kann. Diese Schnittstelle wird von *Terra* benutzt, um neue Menüs in den Browser einzuhängen und die Prozeßkommunikation mit dem Steuermodul *Leonardo* vorzunehmen.

Neu eingehängtes Menü: Es handelt sich um das Menü *Views*, mit dessen Hilfe vordefinierte Anfragen über das Serverskript SFgate an den WAIS-Server gestellt werden. Aus den gelieferten Daten werden Views auf den HTML-Datenbestand generiert. Bisher wurde aber erst ein View spezifiziert, der eine Gesamtsicht bietet.

Schnittstelle zum Steuermodul Leonardo: Die Prozeßkommunikation geschieht durch zwei Tcl-Skripten, eines für jede Richtung der Kommunikation. Durch das in den Skripten verwendete „send"-Kommando werden dem TkWWW-Browser Kommandos geschickt; das Prinzip beruht auf dem Austausch von X-Events.

Der Einsatz des TkWWW-Browsers erfordert eine etwas unsaubere Architektur, da ein angestrebter einheitlicher Kommunikationsweg nicht mehr möglich ist. Da jeder WWW-Browser über einen eigenen Protokollstack verfügt, werden daher die HTML-Dokumente nicht über das Steuermodul, sondern direkt vom Browser geladen.

7.1.2 Das Eingabemodul

Für die Eingabe von Daten wurde eine Reihe von Tcl-Skripten geschrieben, die dem Erfasser bereits eine Richtlinie der einzutragenden Daten gibt. Aus den in der Maske erfaßten Daten werden die standardisierten HTML-Dokumente automatisch erzeugt; nicht möglich ist es, diese Dokumente wieder in die Masken zu laden, anzuzeigen, oder zu editieren. Die Erfasuung von Neudaten wird durch diese Masken nicht besonders gut unterstützt, da weder der deutsche Zeichensatz unterstützt wird, noch die Eingabe eines Feldes mit dem Betätigen der Returntaste beendet werden kann. Der hohe Zeitaufwand resultiert aus der Tatsache, daß nur Attribute und keine Beziehungen eingegeben werden können. Um diese Problematik zu umgehen und die bereits bestehenden LaTeX-Tabellen einzubinden, wurden Konvertierungsskripten geschrieben, die aus diesen LaTeX-Tabellen HTML-Dateien generieren. Diese Tabellen haben das Format, das [Wei94] im Anhang seiner Arbeit verwendet. Im Navigationswerkzeug daVinci müssen noch die Beziehungen eingetragen werden.

7.1.3 Zusammenfassung

Die Eingabemasken erfüllen im Grundsatz die gestellten Anforderungen, ebenso wie das Anzeigewerkzeug TkWWW, das aber die Systemarchitektur durch den gesonderten Kommunikationsweg verwässert.

Das größte Defizit dieses Moduls liegt darin, daß das Editieren von Datenbankeinträgen nicht unterstützt wird. Muß infolge betrieblicher Veränderungen ein Datenbankeintrag modifiziert werden, ist das derzeit nur mit einem „normalen“ Texteditor möglich; dies fördert aber den „Wildwuchs“ im Datenbestand und kann nicht gewährleisten, daß das Standardformat der HTML-Dokumente erhalten bleibt.

7.2 Diskussion der Anforderungen an das Navigationswerkzeug daVinci

DaVinci wird eingesetzt, um die Struktur des HTML-Datenbestands zu visualisieren und damit dem Anwender ein intuitives Begreifen der Zusammenhänge zu erleichtern. Anhand des in Abschnitt 5.1.4 vorgestellten Anforderungskatalogs wird daVinci bewertet.

7.2.1 Systemlandschaft

Dieser Bereich teilt sich in zwei Bereiche:

- das eingesetzte Betriebssystem und
- die Performance.

Betriebssystem: Das Werkzeug daVinci ist auf den unterschiedlichsten Plattformen erhältlich und somit auch auf den gängigen Betriebssystem Unix und Linux. Die Hardwarevoraussetzungen sind unterschiedlich: für Linux wird ein PC vorausgesetzt, für Unix dagegen wird eine Workstation gefordert. DaVinci ist für verschiedene Unix-Systeme erhältlich, für

- HP-9000/7xx-Workstations mit dem Betriebssystem HP-UX
- IBM-RS6000-Workstations mit dem Betriebssystem AIX
- Sun-SPARCstations mit den Betriebssystemen Solaris 1 und Solaris 2 und
- alle anderen Plattformen, die das OPEN LOOK Toolkit XView unterstützen.

In dieser Arbeit wurde daVinci auf einer Sun-SPARCstation IPX mit 32 MB Hauptspeicher installiert.

Performance: Die Entwickler empfehlen, daVinci auf Sun-SPARCstations 10/20 mit 32 MB Hauptspeicher zu betreiben, da Performanceverluste sonst nicht vermeidbar sind. Die Testumgebung bestand aus einer Sun SPARCstation IPX mit 32 MB Hauptspeicher. Beobachtet wurde, daß daVinci in der Anfangsphase nach dem Aufruf eine gute, den Erwartungen entsprechende, Performance aufwies. Der Aufbau eines Graphen mit 70 Knoten dauerte unter zehn Sekunden; bei längerem Einsatz (Laufzeit) wurde daVinci immer langsamer. Nach einer Stunde war die Performance so weit gesunken, daß allein das Markieren eines Knotens bis zu 20 Sekunden beanspruchte. Durch die Laufzeitabhängigkeit liegt die Vermutung nahe, daß der belegte Speicher nicht wieder vollständig von daVinci freigegeben wird und dadurch der verfügbare Hauptspeicher stetig abnimmt. Ob ein solches Problem bei größer dimensionierten Installationen ebenfalls auftritt, kann vom Autor nicht beantwortet werden.

7.2.2 Layout

In diesem Punkt wird die Erfüllung der Anforderungen an die graphische Darstellung durch daVinci kritisch betrachtet.

Visualisierung: daVinci ist der Lage, sowohl gerichtete als auch ungerichtete Graphen darzustellen und erfüllt damit die gestellte Anforderung.

Beschriftung: Ab der Version 1.4 unterstützt daVinci den deutschen Zeichensatz, die Beschriftung im Inneren eines Knotens kann nun umgebrochen werden.

Darstellung von Knoten und Kanten: daVinci unterstützt unterschiedliche Formate der Darstellung der Icons; ab Version 1.4 sind selbstdefinierte Icons möglich. Der innere Bereich von Icons läßt sich mit einer Beschriftung versehen und fördert dadurch das Verständnis beim Anwender. Die Darstellung der Beziehungen zwischen den verschiedenen Knoten erfolgt durch Pfeile, die durch unterschiedliche Gestaltung der Linien – durchgezogen, gepunktet oder gestrichelt – verschiedene Beziehungen repräsentieren.

Darstellung von Farben: daVinci unterstützt eine Palette von 24 Farben zum Füllen der Icons und zum Zeichnen der Pfeile.

Unterstützung von verschiedenen Schriftarten: Diese Anforderung wird durch daVinci, Version 1.4, erfüllt, da es sechs Schriftarten darstellen kann.

Markieren von Knoten und Kanten: Die Markierung der Knoten und Kanten durch schwarze Schatten, ist für den Anwender gut erkennbar. Markierungen permanent zu halten und eine andere Farbe als schwarz zu verwenden ist nicht möglich; eine „History-Funktion“ würde aber diese Option voraussetzen.

7.2.3 Funktionalitätsspezifikation

Es werden alle Aspekte behandelt, die die Funktionen von daVinci behandeln, wie z.B. der Layout-Algorithmus.

Layout-Algorithmus: DaVinci verwendet den Darstellungsalgorithmus von Sugiyama, der die Kantenkreuzungen minimiert.

> **Ebenendarstellung:** Die Ebenen des Rahmenbetriebskonzepts werden durchaus erkannt. Beim Aufbau des ASpecT-Terms ist darauf zu achten, daß ein Knoten immer nur einmal als Vaterknoten eines anderen Knotens auftritt. Ist dies nicht garantiert, ist daVinci zu frei in der Anordnung der Knoten, da es den Vaterknoten immer in einer höheren Ebene als den Sohnknoten positioniert. Durch diesen Trick werden die Knoten meistens in die richtigen Ebenen eingeordnet, da Knoten, die keine Väter haben, in der obersten Stufe[1] stehen. An sie werden, eine Stufe tiefer, die Söhne gehängt. Treten in einer Ebene, z.B. Verfahren, mit den dazugehörigen Aktionen und auch Teilverfahren auf, so kommt es zu einer Verschiebung der Ebenen des Rahmenbetriebskonzepts; neben einem Teilverfahren wird, in derselben Stufe eine Aktion des Verfahrens angezeigt.
>
> **Freies Positionieren von Knoten:** DaVinci gestattet, Knoten innerhalb einer Stufe, in die sie positioniert wurden, zu verschieben. Eine vertikale Verschiebung ist aber nicht möglich, um das automatisch erstellte Layout an die Bedürfnisse des Netzbetreibers anzupassen.
>
> **Zentrieren der Knoten:** Das Zentrieren eines Vaterknotens, der mehrere Söhne besitzt, funktioniert nur in der obersten Ebene. In allen anderen Stufen werden die Väter stets linksbündig über den Söhnen angeordnet.
>
> **Platzoptimierung:** Da daVinci automatisch keine Zeilenumbrüche vornimmt und die Anzahl der Objekte eines instantiierten Rahmenbetriebskonzepts mit dem Grad der Erfassung wächst, wird der Graph stark in

[1] Der Autor versteht unter dem Begriff Stufe eine Schicht, in die daVinci die Knoten einordnet; mit dem Begriff Ebene sind die Ebenen des Rahmenbetriebskonzepts gemeint.

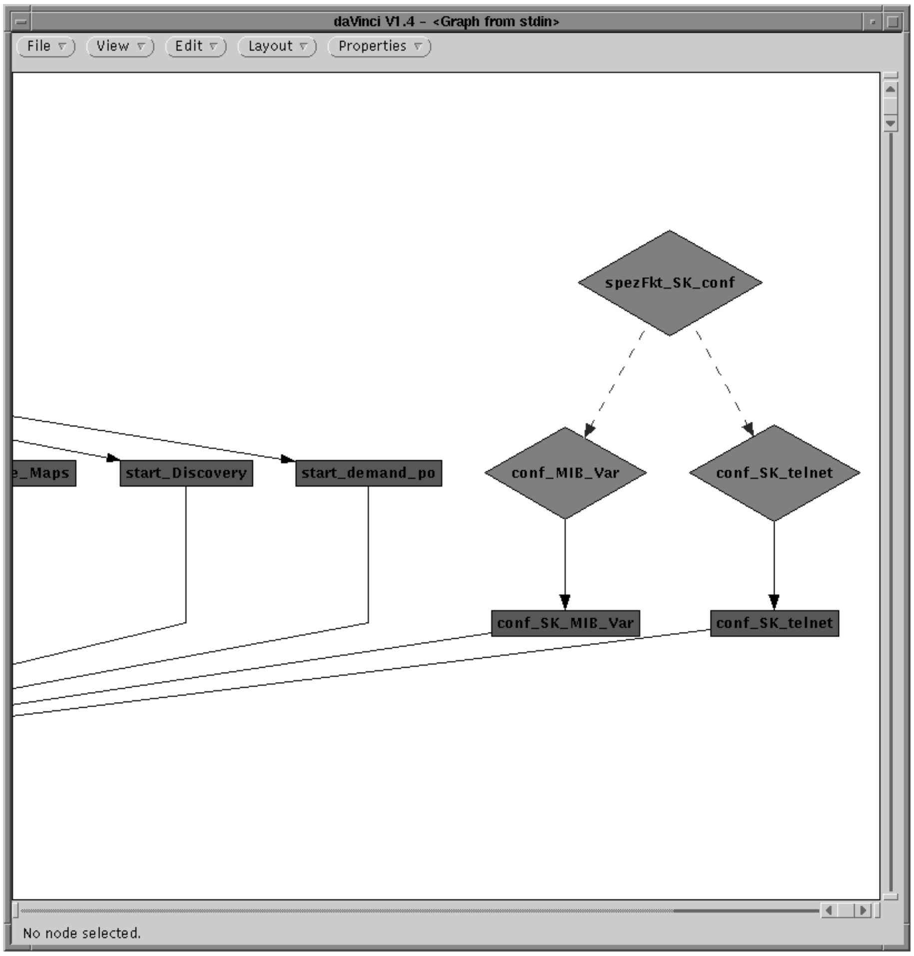

Abbildung 7.1: Ebenendurchmischung von daVinci

die Breite gezogen. Eine Beschriftung wird nur dann umgebrochen, wenn dies bereits beim Generieren des Graphenterms spezifiziert wurde.

Darstellung eines großen Graphen: da Vinci erlaubt die Generierung von Graphen, die größer sind als das auf dem Bildschirm angezeigte Fenster. Dieses Kriterium ist damit erfüllt.

Kommunikationsschnittstelle: daVinci stellt eine definierte Benutzerschnittstelle zur Verfügung, die

- das Laden eines Graphen,
- das Speichern eines Graphen,
- das Sperren von daVinci für Anwenderaktionen und
- die Freigabe für die Anwenderaktionen

unterstützen. Zu dargestellten Graphen können keine Teilgraphen hinzugeladen werden (Append-Funktion); es muß stets ein neu generierter Graph an daVinci geschickt werden. Neben den oben erwähnten Funktionen werden weitere angeboten, wie z.B. das Positionieren und Einstellen der Fenstergröße.

„Event-Binding“: daVinci meldet jede Aktion, die der Anwender ausführt, über die Kommunikationsschnittstelle nach außen, z.B. das Markieren eines Knotens oder das Aktivieren eines Menüpunkts. Durch ein geeignetes Parsing kann eine externe Aktion initiiert werden.

Einhängen eigener Menüs: Das *Edit*-Menü kann an die spezifischen Bedürfnisse des Anwenders angepaßt werden. Leider läßt sich die Beschriftung des Buttons nicht ändern, wodurch es zu Verwirrungen kommen kann, da die neuen Menüpunkte nicht unbedingt mit der Buttonbeschriftung korrespondieren.

Ausblenden von Teilgraphen: Diese Funktion wird von daVinci angeboten und verwendet, um große Graphstrukturen übersichtlicher zu gestalten.

Unterstützung einer Suchfunktion: daVinci bietet diese Funktion nicht an. In großen Graphstrukturen werden häufig bestimmte Knoten gesucht, ohne eine Suchfunktion ist dies mühevoll.

Meldungen:

> **Fehlermeldungen:** Die Fehlermeldungen von daVinci sind oft nicht sehr hilfreich; z.B. wird bei einem Kommunikationsfehler die Meldung ausgegeben: `communication crash - panic syntax error in incoming command`, obwohl ein Kommando von daVinci nach außen gegeben wurde.
>
> **Statusmeldungen:** Statusmeldungen werden von daVinci nicht angeboten.

7.2.4 Zusammenfassung

DaVinci erfüllt den Großteil der gestellten Anforderungen, weist aber auch einige Defizite auf, die bei einem zukünftig eingesetzten Produkt nicht mehr auftreten dürfen. Der gravierendste Nachteil ist der zu komplizierte Layout-Algorithmus zur Minimierung der Kantenkreuzungen. Er liefert zwar ein gutes Ergebnis, jedoch ist die Zeitkomplexität zu hoch. In unserem Fall ist ein Algorithmus ausreichend, der die Objekte in die richtigen Ebenen einordnet. Von dort aus soll der Anwender die Objekte selbst an die seiner Meinung nach richtige Stelle setzen. Dies ist für die Wahrung der Reihenfolge wichtig, wie etwa bei Verfahrensabläufen. Der Algorithmus von Sugiyama verändert aber die Reihenfolge der Objekte, wenn sich dadurch der Gesamtgraph optimieren läßt.

Ein weiterer Nachteil ist der Performanceverlust: Dieses Problem kann auch durch die Konfiguration der Testumgebung hervorgerufen worden sein.

Die Fehlerbehandlung weist ebenfalls noch einige Schwächen auf, so werden Menüfenster nicht mehr ordnungsgemäß geschlossen, wenn dieses mit einer falschen Maustaste aktiviert wurde. Dieser Fehler wird durch das verwendete „OPEN LOOK“-Toolkit hervorgerufen.

Bewertende Zusammenfassung für *daVinci*	
Kriterium	Bewertung
Systemlandschaft	
Betriebssystem	$\surd$
Performance	($\surd$)
Layout	
Graph	$\surd$
Beschriftung	$\surd$
Darstellung von Knoten und Kanten	$\surd$
Darstellung von Farben	$\surd$
Unterstützung verschiedener Schriftarten	$\surd$
Markieren von Knoten und Kanten	($\surd$)
Funktionalitätsspezifikation	
Layout-Algorithmus	
Ebenendarstellung	($\surd$)
Freies Positionieren von Knoten	($\surd$)
Zentrieren von Knoten	$\times$
Platzoptimierung	($\surd$)
Darstellung großer Graphen	$\surd$
Kommunikationsschnittstelle	$\surd$
„Event-Binding"	$\surd$
Einhängen eigener Menüs	$\surd$
Ausblenden von Teilgraphen	$\surd$
Unterstützen einer Suchfunktion	$\times$
Meldungen	
Fehlermeldungen	($\surd$)
Statusmeldungen	$\times$
Kriterium	Bewertung
Legende	
erfüllt	$\surd$
bedingt erfüllt	($\surd$)
nicht erfüllt	$\times$

7.3 Diskussion der Anforderungen an das Steuermodul Leonardo

Scheduling-Mechanismus: Der Scheduler wurde durch eine Endlosschleife implementiert, die den Status der an Leonardo angeschlossenen Unix-Pipes überwacht und anhand der eintreffenden Meldungen in Prozeduren verzweigt. Tritt ein Fehler in dieser Prozedur auf und blockiert dadurch das System, ist der Scheduler nicht in der Lage, dies zu erkennen. Mit einer dem Round-Robin-Verfahren ähnlichen Zuteilungsstrategie könnte dieses Problem beseitigt werden.

Benutzerverwaltung: Eine Benutzerverwaltung wurde bisher nicht implementiert, ebensowenig wie ein Zugriffsschutz auf verschiedene Datenbestände.

Maskengenerator: Ein Maskengenerator wurde für die Eingabe von Daten entworfen, seine Defizite werden im Abschnitt 7.1.2 besprochen.

Konsistenzprüfer: Ein Konsistenzprüfer wurde nicht in das System eingebunden, jedoch existiert ein Public-Domain-Werkzeug mit Namen *MOMspider*, das den Konsistenzcheck von HTML-Datenbeständen durchführt. Am Ende der Bearbeitung wird der Erzeuger oder Administrator der HTML-Datei über email unterrichtet.

Bereitstellung der Kommunikationsinfrastruktur: Das Steuermodul besteht aus einer Sammlung von Perl-Routinen, mit denen alle wichtigen Operationen über das HTTP-Protokoll auf dem Datenbestand ausgeführt werden. Über diesen Weg sollte die gesamte Kommunikation zwischen dem Client, also dem Steuermodul und dem Server stattfinden, jedoch wird dieses Konzept durch den gesonderten Protokollstack des TkWWW-Browser durchbrochen.

Bewertende Zusammenfassung für *Leonardo*	
Kriterium	Bewertung
Scheduling-Mechanismus	($\surd$)
Benutzerverwaltung	$\times$
Maskengenerator	($\surd$)
Konsistenzprüfer	$\times$
Bereitstellen der Infrastruktur	$\surd$
Kriterium	Bewertung
Legende	
erfüllt	$\surd$
bedingt erfüllt	($\surd$)
nicht erfüllt	$\times$

7.4 Diskussion der Anforderungen an die Datenbasis

7.4.1 Performance

Bei unseren Testläufen stellten wir lange Antwortzeiten des WAIS-Servers fest. Der Grund liegt in der langen Dauer des Verbindungsaufbaus zum WAIS-Server. Die Bearbeitung der Anfrage selbst ist ausreichend schnell; eine Untersuchung, wie sich aber die Größe der Datenbank auf die Suchdauer auswirkt, wurde nicht vorgenommen. Bevor jedoch eine Anfrage an die WAIS-Datenbank gestellt werden kann, muß der HTML-Datenbestand indexiert werden. Das kann zu Wartezeiten bis zu mehreren Minuten führen, da für jeden einzelnen Index der gesamte HTML-Datenbestand durchlaufen werden muß. Die Indexierung wird entweder beim Start von Terra, oder bei dessen Beendigung durchgeführt; nie aber während der Arbeitsphase. Die aktuellen Veränderungen innerhalb eines Views befinden sich in der Internen Datenbank des Clients. Der aktuelle View muß dadurch nicht jedesmal neu vom Server geholt werden.

7.4.2 Datenhaltung

Durch die Wahl eines HyperText-Systems ergeben sich Probleme, die nicht auftreten würden, wenn eine relationale oder objektorientierte Datenbank zum Einsatz käme, da die Problemlösung durch das Datenbankdesign vorgenommen wird.

Doppelte Datenhaltung: Nur durch das Anlegen eines WAIS-Datenbanksystems ist es möglich, flexible Abfragen auf die Struktur der HTML-Dokumente vorzunehmen. Als Nachteil erweist sich der zusätzliche Verwaltungsaufwand, aus den HTML-Dokumenten die Struktur zu extrahieren und diese in der WAIS-Datenbank in indexierter Form abzulegen. Dadurch wird die Performance stark gesenkt.

Kein Sperrmechanismus: Greifen gleichzeitig mehrere Benutzer auf ein HTML-Dokument zu und schreiben es anschließend wieder zurück, überschreibt die letzte Schreiboperation auch die Änderungen der anderen Anwender. Um dies zu verhindern muß ein Sperrmechanismus, wie ihn z.B. Semaphore darstellen, implementiert werden. Dieser sperrt während der Editierung ein Dokument für andere Anwender.

7.5 Diskussion der grundlegenden Anforderungen

Neben den Anforderungen an einzelne Module ergeben sich auch modulübergreifende Anforderungen, die nicht einem einzelnen Modul zugeordnet werden können.

7.5.1 Verteiltes Arbeiten

Durch die Client/Server-Architektur ist ein verteiltes Arbeiten grundsätzlich möglich. Es können mehrere Anwender gleichzeitig auf einen gemeinsamen Datenbestand zugreifen, dabei Views bilden und Veränderungen vornehmen. Diese Modifikationen werden erst den anderen Anwendern mitgeteilt, wenn

1. die Modifikationen in der WAIS-Datenbank, die die Struktur des HTML-Datenbestands beinhaltet, aufgenommen wurden und
2. der Anwender einen neuen View anfordert, so daß über das SFgate eine Anfrage an den WAIS-Server gerichtet wird.

7.5.2 Konsistenz des Datenbestands

Die Konsistenz eines Datenbestands wird gewährleistet, solange ein Anwender ausschließlich die durch das Werkzeug zur Verfügung gestellten Funktionen benutzt, z.B. die Eingabemasken zum Anlegen neuer HTML-Dokumente oder das Graphvisualisierungswerkzeug daVinci zum Einfügen von HTML-Links. Muß ein Dokument editiert werden, ist darauf zu achten, daß

1. die Konventionen (vgl. Abschnitt 6.3.2) für ein HTML-Dokument eingehalten werden und daß

2. bei gleichzeitigem Editieren von verschiedenen Anwendern keine Modifikation überschrieben wird, bzw. verloren geht.

Eine zentrale Datenbank läßt sich wesentlich einfacher konsistent halten, da die Daten immer nur an einem Ort stehen und somit nur ein Datenbestand gepflegt werden muß. Aus diesem Grund wurde in *Terra* eine zentrale Datenhaltung gewählt.

7.5.3 Vererbung

Die eingesetzten Programmiersprachen (Perl und Tcl) basieren nicht auf einem objektorientierten Ansatz. Daher konnten auch keinerlei Vererbungsmechanismen realisiert werden.

Kapitel 8

Fazit und Ausblick

Die Entwicklung und Realisierung des Werkzeugs *Terra* hat entscheidende Erkenntnisse für die Werkzeugunterstützung des Rahmenbetriebskonzepts geliefert. Tests mit Beispieldaten, die beim Netzbetreiber BMW gewonnen wurden, führten zu wichtigen Erkenntnissen über die Anforderungen an ein solches Werkzeug.

Die Erfassung einer Betreiberorganisation gemäß dem Rahmenbetriebskonzept liefert eine große Anzahl von Objekten, die zur Zeit in einem „Gesamtview" dem Anwender zur Verfügung gestellt werden. Das Werkzeug Terra ist so konzipiert, daß eine Erweiterung um Views einfach ist, jedoch existieren bisher keine Erkenntnisse, welche Views für einen Netzbetreiber interessant sind. Die in Terra gewählte Darstellungsform ist für die Dienst- und Aufgabenebene ausreichend. Bei der Darstellung von Verfahren traten jedoch Defizite auf:

- Steuer- und Regelinformationen lassen sich nur in einem Beschreibungsfeld angeben
- zeitliche Abhängigkeiten werden nicht dargestellt
- durch das verwendete Graphenwerkzeug daVinci wurde das Layout automatisch generiert und so die Positionierung der Verfahren-Icons verhindert
- die Dienste und Aufgaben, denen ein Verfahren zugeordnet ist, lassen sich nicht ausblenden, dadurch kann die Sicht nicht auf die technischen bzw. betriebswirtschaftlichen Aspekte begrenzt werden

Die gewonnenen Erkenntnisse führten zu einem neuen Anforderungskatalog an ein Werkzeug - *Terra 2*. Dieses kann Abläufe auf der Verfahrens- und Werkzeugebene durch graphische Animationen darstellen (siehe [Fis95]) und bietet eine Möglichkeit, die Funktionalität der benutzten Werkzeuge über eine Schnittstelle zu integrieren. In einer weiteren Diplomarbeit (siehe [Hae95])werden alternative Darstellungsformen von Verfahrensabläufen erörtert.

Der in Kapitel 2 aufgezeigte Ansatz für eine Abbildung bzw. Umsetzung des Rahmenbetriebskonzepts in ein objektorientiertes Modell soll durch [Ewa95] weiter untersucht werden.

Das hier konzipierte Werkzeug ermöglicht die strukturierte und umfassende Dokumentation von Betreiberabläufen, wie sie auch von der ISO-9000-Norm gefordert werden. Zusammenfassend ist zu sagen, daß das hier vorgestellte Werkzeugkonzept durch die Festlegung der ISO-9000-Norm an Bedeutung gewonnen hat.

Teil III

Anhang

Anhang A

Benutzerhandbuch zum Tool zur Erstellung des Rahmenbetriebskonzepts (Terra)

Benutzerhandbuch zum Tool zur Erstellung eines Rahmenbetriebskonzepts (Terra)

MNM TEAM

Münchner Netzmanagement Team

Zusammenfassung

Die bei der Implementierung eines Werkzeuges für das Rahmenbetriebskonzept nötige Beschreibung des Inhalts erfolgt in Form von Hypertext-Dokumenten in der Sprache **HTML** (Version 2.0). Das vorliegende Dokument ist ein Benutzerhandbuch, in dem einleitend die aktuelle Architektur des Werkzeuges dargestellt wird, um dann dem Anwender eine Einführung in die Benutzung des Werkzeuges zu geben.

A.1 Architektur von Terra

In diesem ersten Abschnitt soll die Architektur des **T**ools zur **Er**stellung eines **Ra**hmmenbetriebskonzepts Terra kurz vorgestellt werden. Dadurch erhält der Anwender einen Überblick über das Gesamtsystem.

Das Werkzeug Terra ist vom Grundkonzept her eine verteilte Anwendung. Sie basiert auf einer Client/Server-Architektur. Die einzelnen Bestandteile von Terra werden anschließend kurz erläutert.

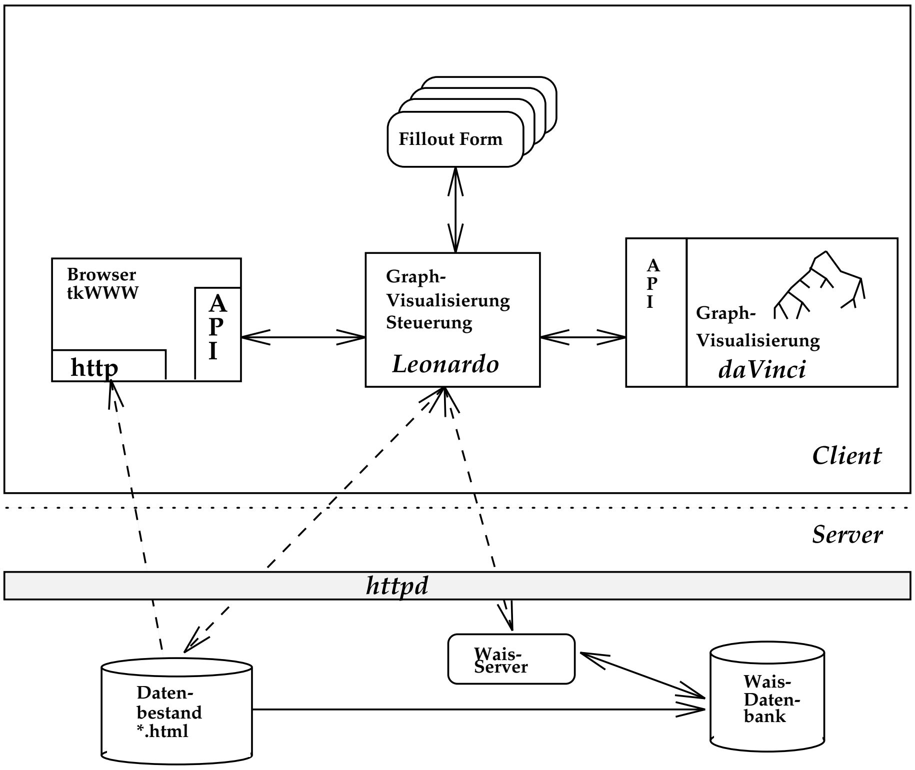

Abbildung A.1: Grobe Architektur

A.1.1 Der Client

Die Bestandteile des Clients sind diejenigen Teile des Gesamtsystems, von deren Arbeitsweise der Anwender am meisten beeinflußt wird. Der Client umfaßt drei Module.

Das Anzeigewerkzeug: Der WWW-Browser „TkWWW“

Wie wir in Abschnitt A.1.3 noch sehen werden, basiert Terra auf einem Hypertextsystem. Um die einzelnen Dokumente adäquat darstellen zu können, wird ein WWW-Browser verwendet. In diesem Fall wurde das graphikfähige WWW-Frontend *TkWWW* gewählt, da es im Gegensatz zu dem derzeit wohl am weitest verbreiteten WWW-Frontend Mosaic eine definierte Programmierschnittstelle bietet, mit deren Hilfe eine Erweiterung bzw. eine Modifikation problemlos vorgenommen werden kann.

Das Steuermodul Leonardo

Das Modul *Leonardo* wurde hier am Lehrstuhl entwickelt. Es bildet die Schnittstelle zwischen dem TkWWW-Browser und dem Graphvisualisierungswerkzeug daVinci, welches in Abschnitt A.1.1 noch genauer beschrieben wird. Darüberhinaus realisiert es die Kommunikation mit dem Server über das Protokoll HTTP (siehe A.1.2). Im Grunde stellt Leonardo das Herzstück von Terra dar, da es die problemspezifische Funktionalität enthält.

Das Graphvisualisierungswerkzeug „daVinci“

Das Graphvisualisierungswerkzeug *daVinci* wurde an der Universität Bremen entwickelt. Es bietet die Möglichkeit, gerichtete Graphenstrukturen darzustellen. In Terra wird daVinci eingesetzt, um die Beziehungen, die zwischen den Ojekten in der Datenbank bestehen, dem Anwender graphisch anzeigen zu können. Von daVinci aus werden alle Benutzeraktionen, die z.B. das Manipulieren der Daten betreffen, ausgelöst.

A.1.2 Das Transportprotokoll HTTP

HTTP ist das Transportprotokoll, das im World Wide Web (WWW) hauptsächlich eingesetzt wird. Es ist zustandslos, d.h. es wird kein Status im HTTP-Server abgelegt.

A.1.3 Der Server

Der Server beinhaltet die Datenbanken sowie die Strukturinformation, über die die einzelnen Objekten miteinander verbunden sind.

Der HTML-Datenbestand

Auf diesem Datenbestand basiert Terra. Es ist keine Datenbank im herkömmlichen Sinne, sondern eine Sammlung von Dateien, die logisch geordnet in verschiedenen Verzeichnissen abgelegt sind. Aus diesem Grund ist eine Struktur zwischen diesen einzelnen Dateien nicht ohne weiteres erkennbar. Daher wird zusätzlich eine WAIS-Datenbank verwendet.

Der WAIS-Server

Der *WAIS-Server* bearbeitet Anfragen, die an die Datenbank gestellt werden.

Die WAIS-Datenbank

Die *WAIS-Datenbank* wird benötigt, um Abfragen auf die Strukturinformation durchführen zu können. Dies wird in einem späteren Stadium wichtig, da es vorgesehen ist, verschiedene Views auf den Datenbestand zu gestatten. Die WAIS-Datenbank enthält Indexe über solche Strukturinformationen und Verweise in den HTML-Datenbestand.

A.2 Grundlegende Funktionen

In diesem Kapitel werden die grundlegenden Funktionen erklärt, die für die Arbeit mit Terra Mindestvoraussetzungen sind. Es lassen sich drei Klassen von Funktionen unterscheiden:

- Funktionen, die sowohl Auswirkungen auf den Browser als auch auf das Graphvisualisierungstool haben
- Funktionen, die nur im Graphvisualisierungstool Auswirkungen haben
- Funktionen, die nur auf den Browser wirken

Im folgenden werden die einzelnen Klassen genauer beschrieben.

A.2.1 Toolübergreifende Funktionen

Unter diese Klasse von Funktionen fallen z.B. das Starten und Beenden von Terra.

Starten des Werkzeugs Terra

Der Nutzer muß zu Beginn sicherstellen, daß der HTTP-Dämon läuft, da sonst die Kommunikation zwischen Client und Server nicht möglich. Weiterhin muß er sich in die Gruppe `hyper` eintragen lassen, da nur diese Zugriff auf die Daten in den Datenbanken haben.

Zum Starten von Terra:

1. Wechsel in das Verzeichnis `/proj/hyper/Browser/bin`

2. Aufruf `Terra` [`Datenbestand`]
   Wird kein Datenbestand im Aufruf mit angegeben, wird ein Default-Datenbestand bearbeitet (im Augenblick ist dies der Datenbestand `ILAN`). Als Datenbestandsname ist das jeweilige Verzeichnis anzugeben.

   Derzeit sind nur zwei Datenbestände verfügbar:

   - der Datenbestand `ILAN`
   - der Datenbestand `BMW_FI`

Es erscheinen zwei Fenster:

- Auf der rechten Seite das Visualisierungwerkzeug „daVinci“

- Auf der linken Seite das Anzeigewerkzeug „TkWWW“

Das „Hochfahren“des gesamten Systems dauert ca. zwei Minuten. Der Grund liegt in Anfrage an den WAIS-Server. Ein Verbindungsaufbau zu diesem dauert verhältnismäßig lang. Danach werden alle HTML-Dokumente, die eine Antwort der Anfrage darstellen, an Leonardo geschickt, der diese Informationen für daVinci aufbereiten muß. Der Hauptgrund für die schwache Performance ist aber eindeutig dem Datenbankzugriff zuzuordnen.

In daVinci wird ein Gesamtgraph des aktuellen Datenbestandes angezeigt, während im TkWWW das Wurzeldokument des Graphen angezeigt wird.

Beenden von Terra

Im TkWWW (linkes Fenster) existiert rechts unten ein Button **Exit Terra**. Dieser wird zum Verlassen von Terra benutzt, denn nur dieser Button gewährleistet ein geregeltes Beenden. Wird der Prozeß nur abgebrochen, so ergeben sich schwerwiegende Probleme beim nächsten Terra-Start, die daVinci mit sofortigem Absturz quittiert. Daher ist auch von dem Kommando „Quit“ unter dem Menüpunkt „Files“ in daVinci Abstand zu nehmen, da dieses Quit nur daVinci beendet.

A.2.2 Funktionen im TkWWW

Der TkWWW ist ein auf X-Window basierender WWW-Browser. Er hat alle grundlegenden Funktionen, die von einem WWW-Browser verlangt werden. So kann natürlich auch innerhalb eines Dokuments auf eine Referenz geklickt werden und das entsprechende Dokument wird angezeigt. Er stellt allerdings keine Inline-Images und keine Fillout Forms dar.

Der TkWWW-Browser besitzt eine Vielzahl von Knöpfen, wobei in Terra nur der Knopf „Exit Terra“ benutzt wird.

A.2.3 Funktionen in daVinci

Das Graphvisualisierungstool daVinci selbst bietet bereits eine große Menge von Funktionen an, wobei in diesem Abschnitt erst einmal auf die grundlegendsten eingegangen werden soll.

Scrollen

Soll der in daVinci dargestellte Graph gescrollt werden, so stehen dem Benutzer mehrere Möglichkeiten offen:

- Den dreigeteilten Scrollknopf in der Mitte mit der linken Maustaste nehmen und verschieben. Der aktuelle Graph wird verschoben, wobei der im Fenster sichtbaren Bereich im Scrollbalken dunkelgrau unterlegt ist. Ist der Graph sehr groß, kann das Scrollen auf diese Art sehr lange dauern.

- Die zweite Möglichkeit besteht darin, nicht direkt auf den Knopf, sondern neben den Knopf zu drücken. Dadurch wird der Graph nicht geshiftet, sondern ein neuer Ausschnitt gebildet und neu dargestellt. Diese Methode ist bei großen Graphen wesentlich schneller als die oben vorgestellte.

Der Graph kann sowohl in x- als auch in y-Richtung verschoben werden.

Markieren von Objekten

Ist nur ein Objekt (eine Kante oder ein Knoten) zu markieren, so ist die rechte Maustaste zu benutzen. Das hat den Vorteil, daß alle bisher markierten Objekte demarkiert werden und nur das eine gewünschte Objekt markiert wird.

Sollen mehrere Objekte markiert werden, so ist das erste Objekt mit der linken Maustaste und alle weiteren Objekte mit der mittleren Maustaste zu markieren.

Menübedienung

Alle Menüs in daVinci sind grundsätzlich mit der **rechten** Maustaste anzuklicken, da sonst Probleme mit dem Widow-Manager auftreten können. Die Menüfenster lassen sich nicht mehr schließen und werden vom Window-Manager nicht mehr als solche erkannt. Dieser Zustand läßt sich nur durch das Beenden von Terra beseitigen.

Doppelklick

Wird in daVinci ein Knoten mit Doppelklick markiert (linke Maustaste), wird das Dokument, das hinter dem Knoten steht, im Browser dargestellt.

Horizontales Verschieben von Knoten

DaVinci bietet von sich aus zwei verschiedene Funktionen an, um Knoten in der Horizontalen zu verschieben:

- Verschieben einer ganzen Ebene
  Voraussetung ist, daß im daVinci Menü **Properties** unter dem Unterpunkt *Layout Algorithm* der Button „Preserve Order“ ausgewählt wurde. Der Anfangsknoten, von dem aus verschoben werden soll, ist mit der linken Maustaste zu markieren. Die linke Maustaste gedrückt halten und die Maus in die gewünschte Richtung ziehen.

- Verschieben eines Knotens innerhalb einer Ebene
  Der einzige Unterschied zum Vorgehen zum Punkt „Verschieben einer ganzen Ebene“ liegt in der Belegung des Untermenüpunktes *Layout Algorithm* des Menüs **Properties**. Hier ist der Button „Modify Order“ zu drücken. Wird der Knoten in der Ebene bewegt, behalten die anderen Knoten ihre Reihenfolge und der zu bewegende Knoten kann über die anderen Knoten springen.

A.3 Funktionen des Menüs „Edit“

Das **Edit**-Menü ist der einzige Menüknopf von daVinci, der vom Anwender verändert werden kann. Für das Projekt TERRA wurden hier die folgenden Menüpunkte eingerichtet:

- *Knoten einfügen*

- *Kindknoten einfügen*

- *Beziehung einfügen*
- *Knoten löschen*
- *Beziehung löschen*
- *Attribute anzeigen*

Die einzelnen Punkte werden in Abhängigkeit von der Anzahl der markierten Objekte (ein Objekt ist entweder ein Knoten oder eine Kante) aktiviert oder deaktiviert.

A.3.1 Der Menüpunkt „Knoten einfügen"

Dieser Menüpunkt wird nur dann freigegeben, wenn weder ein Knoten noch eine Kante selektiert wurde. Wird dieser Menüpunkt ausgewählt, wird damit ein weiteres Untermenü geöffnet, das den Anwender nach dem Knotentypen fragt, der angelegt werden soll. Diese Knotentypen wurden direkt aus den Rahmenbetriebskonzept abgeleitet. Folgende stehen zur Auswahl:

- Dienst
- Aufgabe
- Verfahren
- Aktion
- Werkzeug
- Funktion

Die dem ausgewählten Knotentyp entsprechende Maske wird vom System hochgefahren. Genaueres zu Fillout Forms folgt in Abschnitt A.4.

A.3.2 Der Menüpunkt „Kindknoten einfügen"

Dieser Menüpunkt wird nur dann von Leonardo freigegeben, wenn genau ein Knoten markiert ist. Leonardo fragt den Knotentypen dieses Knotens ab und aktiviert die Knotentypen im Untermenü, die vom Rahmenbetriebskonzept zugelassen werden. Aus dieser Menge kann der Benutzer dann einen auswählen. Leonardo startet selbständig die richtige Fillout Form (vgl. Abschnitt A.3.1).

A.3.3 Der Menüpunkt „Beziehung einfügen“

Der Menüpunkt wird nur dann aktiviert, wenn zwei Knoten selektiert wurden. Zwischen diesen beiden Knoten wird eine Beziehung eingefügt. Es ist wichtig, daß der Vaterknoten und der Sohnknoten identifizierbar sind, daher ist der Vaterknoten als erstes mit der linken Maustaste und der Sohnknoten als zweites mit der mittleren Maustaste zu markieren. Der Rest wird automatisch von Terra erledigt. Ist die Arbeit durchgeführt, so wird ein neuer Graph generiert, der diese neue Kante bereits enthält. Bei großen Graphen kann der Aufbau des Graphen mehrere Sekunden betragen (Erfahrungswerte fehlen hier leider noch; bei 40 Knoten dauert es ca. drei bis vier Sekunden).

A.3.4 Der Menüpunkt „Knoten löschen“

Der zu löschende Knoten ist mit der linken Maustaste zu markieren. Ist dies geschehen, kann dieser Unterpunkt angeklickt werden. Der Knoten wird aus dem Graphen entfernt. Alle Kanten auf diesen Knoten werden eliminiert. Das Dokument wird auf der Server-Seite aus dem aktuellen Verzeichnis in das Unterverzeichnis `delete` geschoben. So ist es für einen späteren Zeitpunkt weiter verfügbar, da es aus dem `delete`-Verzeichnis problemlos in das Arbeitsverzeichnis zurückgeschoben werden kann.

Befindet sich der Knoten mitten im Graphen, so werden die Knoten, auf die Kanten des gelöschten Knoten gezeigt haben, als „Root-Knoten“ dargestellt. D.h. diese Knoten werden auf die oberste Ebene des Graphen (also in die Root-Ebene) gestellt; sie stellen die Wurzel für den darunterliegenden Teilgraphen dar. Der Benutzer muß ggf. Beziehungen zwischen diesen Teilgraphen und anderen Knoten selbst eintragen. Dies kann er mit der Funktion „*Beziehung einfügen*“ die unter Abschnitt A.3.3 beschrieben ist, tun. Gelöschte Beziehungen können nicht wieder zurückgeholt werden.

A.3.5 Der Menüpunkt „Beziehung löschen“

Die zu löschende Beziehung ist mit der linken Maustaste zu markieren. Sie wird in der neuen Version 1.4 von daVinci auch hervorgehoben. Erst dann ist der Menüpunkt freigegeben und kann aufgerufen werden. Es wird die Kante gelöscht und ein neuer Graph generiert, in dem diese Kante natürlich nicht mehr vorkommt.

A.3.6 Der Menüpunkt „Attribute anzeigen“

Beim Anklicken dieses Punktes geschieht derzeit noch nichts, da die Funktionalität noch nicht implementiert wurde.

A.3.7 Knoten im „TkWWW"darstellen

Durch einen Doppelklick mit der linken Maustaste auf einen Knoten, wird dieses Dokument in den tkWWW-Browser geladen und kann dort betrachtet werden. Es ist ebenso möglich, einen Link in TkWWW anzuklicken, um so zum nächsten Dokument zu kommen. Dieses Dokument wird in daVinci markiert.

A.3.8 Horizontales Verschieben von Knoten

DaVinci bietet von sich aus zwei verschiedene Funktionen an, um Knoten in der Horizontalen zu verschieben:

- Verschieben einer ganzen Ebene
  Voraussetzung ist, daß im daVinci Menü **Properties** unter dem Unterpunkt *Layout Algorithm* der Button „Preserve Order" ausgewählt wurde. Der Anfangsknoten, von dem an verschoben werden soll, ist mit der linken Maustaste zu markieren. Die linke Maustaste gedrückt halten und in die Maus in die gewünschte Richtung ziehen.

- Verschieben eines Knotens innerhalb einer Ebene
  Der einzige Unterschied zum Vorgehen zum Punkt „Verschieben einer ganzen Ebene" liegt in der Belegung des Untermenüpunktes *Layout Algorithm* des Menüs **Properties**. Hier ist der Button „Modify Order" zu drücken. Wird der Knoten in der Ebene bewegt, behalten die anderen Knoten ihre Reihenfolge und der zu bewegende Knoten kann über die anderen Knoten springen.

A.4 Die Fillout Forms

Dem Benutzer wird die Eingabe von Knoten dadurch erleichtert, daß ihm Masken zur Verfügung gestellt werden, die nur mit entsprechender Information zu fillen sind. In diese Masken sind

- administrative und
- inhaltliche

Angaben einzutragen. Zu den adminstrativen Angaben gehören neben den Namen auch die email-Adresse des Erstellers/Verfassers. Diese beiden Felder werden schon mit Defaultwerten besetzt. Die Defaultwerte sind der Name und die email-Adresse des Benutzers, welcher TERRA gestartet hat. Diese Angaben sind notwendig, damit bei Problemen in der Datenbank dem Ersteller des Eintrages eine Nachricht geschickt werden kann.

Als weiteres muß ein *Titel* für den Knoten eingegeben werden. Die Knoten werden derzeit noch in Form von HTML-Dokumenten abgelegt und der hier eingegebene Titel ist der Titel des neu erstellten Dokuments. Die *Kurzbezeichnung* ist die Bezeichnung des Knotens in daVinci. Des weiteren wird diese Kurzbezeichnung zum Generieren der Filenamen verwendet; daher sind auch nur solche Zeichen zugelassen, die auch das File-System des entsprechenden Betriebssystems zuläßt. Das Feld *Beschreibung …* besitzt auch jede der Masken. In sie ist eine Beschreibung des jeweiligen Objekts einzutragen (handelt es sich z.B. um ein Verfahren, so sind die Zielsetzung dieses Verfahrens und die Voraussetzungen, die gültig sein müssen, einzugeben). Neben den Eingabefeldern existieren noch zwei Knöpfe, die ein Abschicken der Fillout Form (*Generiere Knoten*) oder ein Verwerfen (*Exit (kein Knoten anlegen)*) veranlassen.

Ist die Fillout Form abgeschickt worden, wird ein neuer aktueller Graph generiert. Der Aufbau kann mehrere Sekunden dauern (Erfahrungswerte für große Graphen fehlen leider noch). Weiterhin wird der neu generierte Knoten immer ganz links in den neuen Graphen eingeordnet. Es kann passieren, daß außer dem neuen Punkt nichts auf dem Bildschirm zu sehen ist. Der Rest des Graphen ist ganz rechts angeordnet. Das Problem liegt im Layout-Algorithmus von daVinci, d.h. wir haben darauf keinen Einfluß.

Achtung:

1. Der Wechsel von einem Eingabefeld zum nächsten geschieht über einen Mausklick mit der <u>linken</u> Maustaste in das gewünschte Eingabefeld.

2. Es können derzeit keine deutschen Umlaute (ä, ö, ü, ß, …) umgesetzt werden. Auf keinen Fall dürfen Umlaute und Leerzeichen in der *Kurzbezeichnung* vorkommen, da es sonst Probleme mit dem späteren Zugriff auf diese Datei gibt. Diese Probleme können so weit gehen, daß das System Terra abstürzt.

3. Wird eine leere Fillout Form zurückgegeben, so wird daVinci derzeit nicht mehr aktiviert. Zu debugging-Zwecken gibt es die Möglichkeit, an daVinci auch Befehle über die Tastatur zu schicken. DaVinci kann wieder aktiviert werden, indem im debugging-Fenster `activation` eingetippt wird, d.h. daVinci ist wieder bereit, Eingaben vom Anwender entgegenzunehmen. Über diese debugging-Fenster kann daVinci jeder Befehl, den die API unterstützt, geschickt werden.

Im weiteren Verlauf wird nur noch auf die speziellen Eigenheiten einer jeden Fillout Form eingegangen.

A.4.1 Die Fillout Form für einen Dienst

Eine Fillout Form für einen Dienst besteht aus mehreren Komponenten.

- checkbuttons
- einzeiligen Eingabefeldern
- mehrzeiliges Eingabenfeld
- Knöpfe

Durch die checkbuttons wird spezifiziert, um welche Unterart von Dienst es sich handelt (internes/externes, internes oder externes Dienstpaket bzw Dienst).

Danach folgen die vier Felder für den Namen, die email-Adresse, den Titel des Dokuments und der Kurzbezeichnung für das Graphvisualisierungwerkzeug daVinci.

Die drei folgenden Eingabefelder spezifizieren die Lieferanten für einen Dienst, die Kunden dieses angebotenen Dienstes sowie die Abrechnungsart des angebotenen Dienstes.

A.4.2 Die Fillout Form für eine Aufgabe

Neben den bereits erwähnten Komponenten existieren noch

- radiobuttons, um den Aufgabentyp festzulegen und
- ein Eingabefeld für die Rolle bzw. Zuständigkeit. Einzutragen ist hier derjenige, der die Aufgabe administriert, bzw. überwacht.

Es wurden radiobuttons (1:n Auswahl) verwendet, da eine Aufgabe immer nur einen Typ besitzen kann.

A.4.3 Die Fillout Form für ein Verfahren

Diese Fillout Form ist von der Grundstruktur identisch mit der Fillout Form für eine Aufgabe. Der einzige Unterschied besteht in der Bezeichnung für das Eingabefeld. Hier soll ein Ablauf genauer spezifiziert werden.

A.4.4 Die Fillout Form für eine Aktion

Eine Aktion ist ein elementarer Arbeitsschritt, der mit dieser Fillout Form erstellt werden kann. Die Grundstruktur ist identisch zu den vorhergegangenen Masken, nur daß es keine radio- oder checkbuttons gibt. Statt dessen werden zu den Grundelementen noch zwei einzeilige Eingabefelder ausgegeben, in die

1. die *Funktion* des angewendeten Werkzeugs und

2. die *Information* (z.B. MIB), auf die sich diese Funktion abstützt,

einzugeben sind.

A.4.5 Die Fillout Form für ein Werkzeug

Diese Maske dient der Instantiierung eines Werkzeuges einzugeben. Verlangt werden einerseits die administrativen Angaben und andererseits Angaben über den *Administrator*, dessen *Hotline* (anzugeben ist hier z.B. seine Telefonnummer) und der *Rechner*, auf dem das Werkzeug installiert wurde.

A.4.6 Die Fillout Form für eine Funktion

Neben den Standardeingaben wird dem Benutzer ein Feld *Hilfe* zur Verfügung gestellt. Darin können Hilfestellungen, die diese Funktion betreffen, angeboten werden.

A.5 Menüs in „daVinci“

DaVinci dient in Terra zur Benutzerführung. Von hier aus kann der Benutzer die gesamte Funktionalität, die von Terra angeboten wird, anfordern. Ein Teil der Funktionalität wird von daVinci direkt angeboten. Die neu integrierte Funktionalität wird unter dem Menüpunkt **Edit** sichtbar (siehe Abschnitt A.3). Dieser Punkt wurde von uns in daVinci neu eingehängt.

Achtung: Alle Menüs in daVinci sind mit der <u>*rechten*</u> Maustaste anzuklicken, da sich sonst die Menüfenster nicht mehr schließen lassen. Die Menüfenster erscheinen im weiteren Verlauf der Terra-Sitzung für den Window-Manager durchsichtig. Dieser Zustand läßt sich nur durch das Abbrechen der Terra-Sitzung (siehe Abschnitt A.2.1) beenden.

A.5.1 Das Menü „Files“

Klickt der Benutzer diesen Menüpunkt mit der rechten Maustaste an, so wird ein Fenster mit Unterpunkten geöffnet:

- *Load Graph*
  Es kann ein ASpecT-Term geladen werden, der einen Graph repräsentiert. Wie ein ASpecT-Term aufgebaut ist, kann ggf. im daVinci-Handbuch nachgelesen werden.

- *Save Graph*
  Der aktuell angezeigte Graph wird als ASpecT-Term abgespeichert.

- *Load Status*
  Es wird ein Graph mit einem Status geladen. Ein solcher Status ist meist ein vom Benutzer erstelltes Layout.

- *Save Status*
  Der aktulle Status wird in einer Datei abgelegt.

- *Save Postscript*
  Ein unter dem Menüpunkt **Properties** (siehe Abschnitt A.5.5) erzeugter Postscriptcode des aktuellen Graphen wird in einer Datei abgelegt.

- *Connect Application*
  Dieser Unterpunkt gestattet es, an daVinci eine Applikation anzubinden. In Terra ist diese angebundene Applikation das Steuermodul Leonardo. An daVinci kann immer nur eine Applikation angebunden werden.

- *Quit*
  Dieser Menüpunkt gestattet es, daVinci zu verlassen. Doch in unserem Fall soll er nicht benutzt werden, da das Beenden vom tkWWW aus geschieht. Gründe hierzu sind unter Abschnitt A.2.1 zu finden.

A.5.2 Das Menü „Views“

Wird dieser Menüpunkt angeklickt, so öffnet sich folgendes Untermenü:

- *Scale*
  Hiermit läßt sich die Größe des Graphen von null bis hundert Prozent der Ursprungsgröße stufenlos verändern. Es werden dabei aber die Knoten- und Kantenbezeichnungen nicht dargestellt, so daß ein bestimmter Knoten nur anhand seiner Lage im Graphen identifiziert werden kann. Wird ein Knoten angeklickt, so wird der Knotenname in der Statuszeile von daVinci (linke untere Ecke) angezeigt.

- *Fit Scale to Window*
  Der Gesamtgraph wird so scaliert, daß er komplett im aktuellen Fenster betrachtet werden kann. Mit dem *Scale*-Kommando kann er wieder auf die Ursprungsgröße gebracht werden.

- *Graph Info*
  Es werden Informationen zum aktuellen Graphen in einem Fenster ausgegeben, z.B. die Anzahl der, Knoten und Dummy-Knoten, der Abstand zwischen den einzelnen Knotenebenen etc.

- *daVinci Info*
  Dem Benutzer werden allgemeine Informationen über daVinci in einem Fenster angeboten.

  Achtung: Dieses Fenster stellt keine Hilfefunktion zur Verfügung!

A.5.3 Das Menü „Edit“

Auf diesen Menüpunkt wird in Abschnitt A.3 auf Seite 125 genauer eingegangen, da er die Funktionalität, die vom Rahmenbetriebskonzept verlangt wird, bereitstellt.

A.5.4 Das Menü „Layout“

Mit diesem Menüpunkt hat der Anwender Einfluß auf das Layout des aktuelle Graphen. Klickt man mit der <u>rechten</u> Maustaste auf diesen Button, so öffnet sich folgendes Untermenü:

- *Place all Nodes*
  Der Gesamtgraph wird komplett neu berechnet. Ziel ist ein optimierter Graph, wobei die Anzahl der Kantenüberschneidungen minimal sein soll. Die Parameter, nach denen das Layout neu berechnet werden soll, können im Menü **Properties** (vgl. A.5.5 Seite 134) unter dem Punkt **Layout Algorithm** genauer spezifiziert werden.

- *Place visible Nodes*
  Es wird hier nur der Teil des Graphen optimiert, der in dem aktuellen Fenster sichtbar ist. Daraus soll sich ein optimaleres Layout ergeben, da alle außerhalb liegenden Knoten nicht beachtet werden. Es können ebenfalls Parameter im Menü **Properties** spezifiziert werden.

- *Compact Graph*
  Der Platzbedarf des Gesamtgraphen wird komprimiert. Diese Funktion ist sinnvoll, wenn z.B. mit **Hide Subgraph** ganze Teilgraphen „versteckt“ werden und so große Löcher innerhalb des Graphen entstehen würden.

- *Hide (oder Show) Subgraph*
  Ist ein Knoten selektiert, wird beim Aufruf dieser Funktion der gesamte Teilgraph unter diesem Knoten „versteckt“.

- *Restore all Subgraphs*
  Alle mit **Hide Subgraph** geschlossenen Teilgraphen werden wieder geöffnet.

- *Hide (oder Show) Edges*
  Es werden alle Kanten eines Knotens für den Betrachter unsichtbar gemacht und der betreffende Knoten wird durch eine schwarze Box markiert.

- *Restore all Edges*
  Alle unsichtbaren Kanten werden wieder dargestellt.

A.5.5 Das Menü „Properties“

- *Layout Algorithm*
  Hier können Parameter an den Layout-Algorithmus übergeben werden. Für genauere Informationen sei hier auf das daVinci-Handbuch verwiesen.

- *Layout Dimensions*
  Hiermit kann man die Dimensionen des Gesamtgraphen verändern, wie z.B. Abstände der Knoten oder Fonts

- *Application*
  Es ist möglich, an daVinci über Unix-Pipes eine Applikation anzubinden. In Terra wurde die Kommunikation so realisiert. Es kann immer nur eine einzige Applikation angebunden werden. Da dies bereits geschehen ist, wird dieser Menüpunkt immer deaktiviert sein.

- *Postscript*
  Es werden die Parameter des aktuellen Graphen bestimmt, die zum Generieren eines Postscript-Codes gebraucht werden.

Anhang B

Abkürzungsverzeichnis

ASpecT	Algebraic Specification Transformer
BMW	Bayerische Motorenwerke
CGI	Common Gateway Interface
FTP	File Transfer Protocol
GUI	Graphical User Interface
HP	Hewlett Packard
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IBM	International Buiseness Machines Inc.
IEEE	Institute of Electrical and Electronic Engineers
IMS	Information Management System
IP	Internet Protocol
ISO	International Standards Organization
IV	Informationsverarbeitung
LAN	Local Area Network
LRZ	Leibniz Rechenzentrum
MAN	Metropolitan Area Network
MPF	Management-Plattform
MVS	Multiple Virtual Storage
NM	Netzmanagement
OSI	Open Systems Interconnection
oo	objektorientiert
Perl	Practical Extraction and Report Language
RBK	Rahmenbetriebskonzept
RMON	Remote Network Monitoring
SGML	Standard Generalized Markup Language
SNA	Systems/Structured Network Architecture
SNMP	Simple Network Management Protocol

TCP	Transmission Control Protocol
Tcl	Tool Command Language
Terra	Tool zur Erstellung des Rahmenbetriebskonzepts
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Universal Resource Name
VM	Virtual Machine
WAIS	Wide Area Information System
WAN	Wide Area Network
WS	Workstation
WWW	World Wide Web

Literaturverzeichnis

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

[Bal82] Helmut Balzert. *Die Entwicklung von Software-Systemen: Prinzipien Methoden, Sprachen, Werkzeuge*. Wissenschaftsverlag, Mannheim, Wien, Zürich, 1982.

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

[BL94] Tim Berners-Lee. *Hypertext Transfer Protocol - A Stateless Search, Retrieve and Manipulation Protocol*, Mai 1994.

[BLCM94] T. Berners-Lee, D. Connolly und K. Muldrow. *HyperText Markup Language (HTML) - Version 2.0*, Februar 1994.

[Bro92] Manfred Broy. *Algorithmische Sprachen und Methodik des Programmierens I und II*, May 1992.

[Den91] E Denert. *Software Engineering*. Springer Verlag, 1991.

[Ege94] Rudolf Egerer. *Entwurf eines Modells für die Erstellung einer Verfügbarkeitsdokumentation und deren effektive Nutzung im Bereich des integrierten Netzmanagements*. Diplomarbeit, Technische Universität München - Institut für Informatik, November 1994.

[Ewa95] Karl Ewald. *Entwicklung einer Methodik zur systematischen Gewinnung generischer Konfigurationsvorgänge am Beispiel ausgewählter LAN-Komponenten*. Diplomarbeit, Technische Universität München - Institut für Informatik, May 1995.

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

[Fis95] Bernhard Fischer. *Bewertung des Programmiersystems KHOROS hinsichtlich der Beschreibung und Anwendung von Verfahren.* Fortgeschrittenenpraktikum, Technische Universität München - Institut für Informatik, voraussichtlich März 1995.

[FW94] Michael Fröhlich und Mattias Werner. *The Interactive Graph Visualization System daVinci V1.4*, Oktober 1994.

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

[Hab94] Cornelia Haber. *Entwicklung einer Methodik zur systematischen Analyse von Objektkatalogen ausgewählter LAN-Komponenten.* Diplomarbeit, Technische Universität München - Institut für Informatik, August 1994.

[Hae95] Christina Haegele. *Bewertung und Analyse alternativer Darstellungsm glichkeiten f r Verfahren aus dem Netz- und Systemmanagement.* Diplomarbeit, Technische Universität München - Institut für Informatik, August 1995.

[HAS+94] Heinz-Gerd Hegering, Sebastian Abeck, Stefan Schwerdtner, Peter Segner, Rene Wies, Oliver Blume, Christian Fischer, Cornelia Haber, Andreas von der Heyde, Christian Mayerl, Thorsten Vogs und Hans-Peter Weiß. *Projekt Terra -Tool zur Erstellung eines Rahmenbetriebskonzepts*, Juni 1994.

[Hei91] Edmund Heinen. *Industriebetriebslehre - Entscheidungen im Industriebetrieb.* Gabler-Verlag, 9. Auflage, 1991.

[Hol94] Ralf Holighaus. *Zwischen den Welten.* *iX*, Seiten 30–36, Oktober 1994.

[ISO87a] DIN: ISO 8879, 1987.

[ISO87b] DIN: ISO 9000, 1987.

[Jag94] August-Wilhelm Jagau. *Mainframes neue Wege - TCP/IP unter MVS im Einsatz bei der BASF.* *iX*, Seiten 46–55, Oktober 1994.

[Klu94a] Rainer Klute. *Generischer Generator - Dynamische Dokumente mit dem CERN-WWW-Server.* *iX*, Seiten 178–187, September 1994.

[Klu94b] Rainer Klute. *Zweiter Gang - Dynamische Dokumente mit dem CERN-WWW-Server.* *iX*, Seiten 140–146, August 1994.

[lib94] libwww-perl: Distribution Information, 1994. ftp://liege.ics.uci.edu/pub/arcadia/libwww-perl/README.html.

[May94] Christian Mayerl. *Entwurf und prototypische Umsetzung eines Beschreibungsrahmens für Funktionen im Netz- und Systemmanagement unter Verwendung einer hypertext-basierten Schnittstelle.* Fortgeschrittenenpraktikum, Technische Universität München - Institut für Informatik, Mai 1994.

[Mul89] Sape Mullender. *Distributed Systems.* Addison Wesley (ACM Press), 1989.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[Pfe95] Ulrich Pfeifer. *HTTPs älterer Bruder - WAIS: Inhaltsorientierte Suche im Internet.* *iX*, Seiten 120–126, Januar 1995.

[RBP+91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy und William Lorensen. *Object-Oriented Modelling and Design.* Prentice Hall International, Inc, 1991.

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

[SM91] Kozo Sugiyama und Kazuo Misue. *Visualization of Structural Information: Automatic Drawing of Compound Digraphs. IEEE Transactions on Systems, Man and Cybernatics*, SMC-21(4):876–892, Juli/August 1991.

[STT81] Kozo Sugiyama, Shojiro Tagawa und Mitsuhiko Toda. *Methods for Visual Understanding of Hirarchical System Structures. IEEE Transactions on Systems, Man and Cybernatics*, SMC-11(2):109–125, Februar 1981.

[Tan92] Andrew S. Tanenbaum. *Modern Operating Systems.* Prentice Hall, 1992.

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

[vHS93] Jörn von Holten und Richard Seifert. *ASpecT 2.0 User Manual*, Februar 1993.

[Vog94] Thorsten Vogs. *Entwurf und prototypische Umsetzung eines Beschreibungsrahmens für Funktionen im Netz- und Systemmanagement unter Verwendung einer hypertext-basierten Schnittstelle.* Fortgeschrittenenpraktikum, Technische Universität München - Institut für Informatik, Mai 1994.

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

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