Universität Konstanz
Fakultät für Verwaltungswissenschaft
Fachgruppe Informationswissenschaft

Komplexitätsreduktion bei der Anzeige von Suchergebnissen in Baumstrukturen Visualisierungs-techniken für Hypertexte

Diplomarbeit **Konstanz, im Wintersemester 1997/1998**
von Christof Ettl

Zusammenfassung

Die vorliegende Arbeit befaßt sich mit Methoden zur Visualisierung umfangreicher Hypertextstrukturen. Es werden Verfahren zur Komplexitätsreduktion und Instrumente zur Navigation in derartigen Wissenskollektionen vorgestellt. Im Rahmen des Projektes VIR (Verteilte Informationsräume) an der Universität Konstanz wird eine flexible Navigationskomponente für ein Visualisierungssystem entworfen und implementiert.

Abstract

This work deals with methods for the visualization of extensive hypertext structures. It´s an introduction to methods to reduce the complexitivity of collections of knowledge and tools to navigate in it.
As a part of the project ”Virtuell Information Spaces” at the university of Konstanz, a flexible component for navigation within a visualisation system will be designed and implemented.

Inhaltsverzeichnis

Einleitung und Motivation

Für die Benutzer von Informationssystemen wird heutzutage die Suche und der Zugriff auf relevante Daten und Informationen immer schwieriger, da sowohl die Menge an Daten, Wissen und Informationen exponentiell anwächst, als auch durch die Verfügbarkeit schneller, digitaler Netzwerke mehr Wissensquellen zugänglich gemacht werden können.

Benutzerschnittstellen zeitgemäßer Informationssysteme müssen dieser Tatsache Rechnung tragen, indem sie Werkzeuge zum effektiven Information Retrieval und zur geeigneten Aufbereitung der Informationen zur Verfügung stellen. Eine Reihe Suchdienste haben darauf reagiert und bieten ihre Dienste im World Wide Web an.

Bei den heute zur Verfügung stehenden Informationsmengen reichen jedoch die einfachen Techniken des Information Retrievals nicht mehr aus. Die Benutzer sind nicht mehr in der Lage, die für sie relevanten Informationen durch exploratives Browsing (Navigieren anhand vorgegebener Links) aufzufinden und zu nutzen.

Vielmehr entsteht die Forderung nach neuen Techniken zur Informations-aufbereitung und Recherche, zumal durch den technischen Fortschritt, der bei der Entwicklung von Massenspeichern sowie graphischer Hardware gemacht wird, auch neue Möglichkeiten zur Verarbeitung großer Datenmengen geboten werden.

Besonders die Visualisierung derartiger Informationsräume spielt eine zunehmend größere Rolle, um das Lost-in-Hyperspace-Gefühl beim Benutzer gar nicht erst aufkommen zu lassen, sondern ihm das Navigieren zu erleichtern und somit die Informationssuche effektiver zu gestalten.

Die vorliegende Arbeit wurde im Rahmen des Projektes VIR (Verteilte Informationssräume) an der Universität Konstanz durchgeführt. Dieses Projekt hat zum Ziel, das Suchen und Navigieren in elektronischen Marktplätzen zu verbessern.

In dieser Arbeit werden aktuelle Verfahren zur Komplexitätsreduktion bei der Visualisierung großer, hypertextartig organisierter Wissensbasen und Instrumente zur Navigation in solchen Systemen vorgestellt.

Im zweiten Teil wird ein eigenes flexibles Navigationssystem für die Visualisierung von Suchergebnissen als Graph beschrieben, welches vom Autor entworfen und implementiert wurde.

Überblick

In Kapitel 1 ”Informationsüberflutung und Orientierungsprobleme” werden Probleme vorgestellt, vor die sich der Benutzer bei der Informationssuche in großen, verteilten Informationsräumen gestellt sieht und die ihm von heutigen Suchmethoden abgenommen werden sollen.

Im folgenden Kapitel 2 ”Metainformation und gezielte Navigation” wird der aktuelle Stand der Technik zur Lösung dieser Schwierigkeiten beschrieben. Es werden Verfahren zur Reduzierung der Komplexität des Hypertextes, sowie Instrumente zur effektiven Navigation und Orientierung vorgestellt.

Danach werden diese Verfahren auf ihre Anwendbarkeit im konkreten Problemfall hin untersucht. Im Kapitel ”Methodenauswahl und Grobentwurf” wird begründet, welche der vorgestellten Instrumente und Verfahren beim konkreten Entwurf der Anwendung zum Einsatz kommen.

Dieser ”Entwurf der Anwendung” ist Gegenstand des Kapitels 4, hier wird das Design und die Funktionalität des vorhandenen Prototyps, für den das Navigationssystem konzipiert werden soll, vorgestellt. Es folgt die Beschreibung des Systementwurfs und ein Abschnitt über die Systemintegration.

Das letzte Kapitel ”Implementierung” enthält die Beschreibung der konkret in Java programmierten Klassen und Objekte, aus denen sich die Navigationskomponente zusammensetzt.
Anhand von Beispielgraphen wird deren Funktionsweise erläutert und der Effekt für den Benutzer dieser Navigationskomponente deutlich gemacht.

1. Informationsüberflutung und Orientierungsprobleme

Dieses Kapitel beschreibt Probleme für den Benutzer, die bei der Suche und Verwendung von Informationen, die in sehr umfangreichen Datenkollektionen vorliegen, entstehen können.

1.1. Stetiger Zuwachs an verfügbarem Wissen

Ein Weg Informationen in einem Informationssystem zu suchen, besteht darin alle Informationseinheiten nacheinander durchzusehen und auf ihre Relevanz hin zu prüfen. Es leuchtet ein, daß dieses Vorgehen ab einer bestimmten Menge an Daten jeglichen Anspruch an Effizienz und Effektivität verliert. Es kommt nur für kleine Informationsräume in Frage, die flächendeckend abgesucht werden können und in denen sich der Benutzer derart auskennt, daß er weiß, wo er zu suchen hat.

Diese Methode zur Informationssuche ist jedoch eher die Ausnahme. Als Faustregel gilt heutzutage, daß sich das menschliche Wissen in Form von Veröffentlichungen alle zehn Jahre verdoppelt [Nielsen95]. Die wenigsten Forscher sind noch in der Lage, sich selbst auf ihrem Spezialgebiet über den aktuellen Stand der Veröffentlichungen auf dem laufenden zu halten.

Nicht nur die Menge der verfügbaren Informationen wächst derart schnell an, auch die Anzahl der verfügbaren Informationsquellen explodiert. Über die Dienste des Internets (WWW, Gopher, News etc.) werden verschiedenste Wissensquellen zugänglich gemacht, deren Anzahl sich – wie das gesamte Internet – fast jedes Jahr verdoppelt.

Der Benutzer steht also nicht nur vor dem Problem, wie er die Informationen aus einem großen Informationssystem herausfischt. Er muß sich zuerst einmal entscheiden, welche Informationsquellen er überhaupt in Anspruch nimmt und welche er ignoriert. Schon diese Entscheidungen tragen wesentlich zum Erfolg seiner Recherche bei.

1.2. Verteilte, heterogene Informationsräume

Im Internet hat der Benutzer Zugriff auf die unterschiedlichsten Informationssystemtypen (Anwendungen), welche auf sehr vielen verschiedenen Servern abgelegt sind. Die verschiedenen Anwendungen verwenden viele unterschiedliche Benutzerschnittstellen. Dies können graphisch ansprechende, komfortable WWW-Seiten, aber auch Terminalemulationen sein, welche mittels proprietären Abfragesprachen einen Datenbankzugriff erlauben. Das Internet wird in erster Linie durch weltweit operierende Suchmaschinen erschlossen.

Betrachtet man im Gegensatz zum globalen, alles umfassenden Internet ”nur” regional oder fachlich abgegrenzte Informationsräume, so entschärft sich das Problem keineswegs, da auch diese Domänen das gleiche Maß an Heterogenität bzw. Verteiltheit aufweisen, obwohl sie von viel kleinerem Umfang sind.

Der Benutzer muß sich also in einem verteilten, heterogenen System zurecht finden, er muß die verschiedenen Dienstleistungen kennen und in Anspruch nehmen können und er darf sein eigentliches Informationsbedürfnis nicht aus den Augen verlieren. Es leuchtet ein, daß dieses Unterfangen mit Schwierigkeiten verbunden ist.

1.3. Unzulänglichkeiten heutiger Suchmaschinen

Das freie Browsing in den WWW-Seiten, bzw. die Nutzung von hierarchisch organisierten Übersichten (Katalogen) reicht nicht mehr aus, um relevante Informationen zu finden und zu nutzen. Das Angebot an Suchmaschinen ist zwar sehr umfangreich [Bekavac96], jedoch treten auch bei diesen Techniken des Information Retrieval Schwierigkeiten auf:

- Durch das periodische Durchsuchen und Erschließen vieler tausender WWW-Server durch die verwendeten Suchroboter wird eine derart hohe Netzbelastung verursacht, das diese Technik nicht mehr lange effektiv eingesetzt werden kann.

- Ergebnisse von Suchmaschinen werden aktuell als lange Listen von Trefferseiten ausgewiesen, die dann vom Benutzer einzeln geladen und überprüft werden müssen. Bei dem sehr großen Informationsangebot, welches im Internet gegeben ist, kann es vorkommen, daß auch auf sehr differenzierte Suchanfragen, lange Trefferlisten mit vielen redundanten Einträgen generiert werden. Der Benutzer erhält keinerlei Informationen über den Kontext, in dem die Seiten gefunden wurden; er muß sich diesen erst mühselig explorativ erschließen.

1.4. Orientierungslosigkeit im Hyperspace

Ist ein Informationssystem identifiziert, welches relevante Informationseinheiten enthält, beginnt der Benutzer dieses zu erforschen (Browsing).

Die Navigation in größeren Informationsräumen ist für viele Benutzer problematisch. Sie verlieren die Orientierung, können die Informationen, nach denen sie suchen nicht finden und sind schnell derartig frustriert, daß sie bei ihrer nächsten Recherche wohl eher eine andere Informationsquelle in Anspruch nehmen. Eine gute Schnittstelle, die dem Benutzer derartige Schwierigkeiten abnimmt, kann so ein entscheidendes Qualitätsmerkmal sein, welches über Erfolg und Mißerfolg eines neuen Produktes mitentscheidet.

Hypertextsysteme (z.B. das WWW) bieten die Möglichkeit, das Lesen bzw. Schreiben zu erleichtern und zu fördern, dieses Konzept birgt aber die Gefahr in sich, den Leser oder Autor zusätzlich zu verwirren.

Probleme, die beim Umgang mit Hypertext / Hypermedia auftreten können sind nach [Agosti96]:

- **disorientation**: the tendency to lose one´s sense of location and direction in a nonlinear document.
  Der Benutzer weiß nicht mehr, wo im Hypertext er sich befindet bzw. wie ihm die Navigations- und Browsingmöglichkeiten des Systems weiterhelfen können. Er hat keinen Überblick, wieviele Informationseinheiten er noch zu lesen hat, bzw. wieviele er bereits gelesen hat.

- **cognitive overhead/-load**: the additional effort and concentration necessary to maintain several tasks or trails at one time.
  Der Benutzer muß zusätzlich zu den eigentlichen Inhalten des Hypertextes auch das kognitive Modell verstehen, das die Autoren bei der Erstellung des Hypertextsystems zugrundegelegt haben. Sein Aufwand an Konzentration und Anstrengung beim Lesen erhöht sich also und muß in Relation zum Nutzen stehen, den er durch den Hypertext erfährt.

Es wird im Zusammenhang mit Hypertexten sowohl von Orientierungs- als auch von Navigationsproblemen gesprochen. Da diese beiden Phänomene jedoch eng miteinander verbunden sind, wird allgemein der Begriff der Navigationsprobleme verwendet und nur dann differenziert, wenn einer der beiden Aspekte besonders betrachtet wird.

Diese und weitere Aspekte des Orientierungsverlustes bzw. der kognitiven Überlast werden auch in [Kuhlen91 S.133] und [Utting&Yankelovich 89] ausführlich beschrieben.

1.5. Anforderungen an zeitgemäße Informationssysteme

Welche Informationen bzw. welche Instrumente sollten dem Benutzer zur Verfügung gestellt werden, um derartige Probleme zu vermeiden ?

1.5.1. Kontextinformationen

Die beschriebenen Schwierigkeiten bei der Auswertung von Trefferlisten heutiger Suchmaschinen könnten durch Darstellung der inhaltlichen Kontextinformation zumindest entschärft werden. Als Ergebnisse des Retrievals werden nicht mehr einzelne Seiten, sondern navigierbare Hypertextstrukturen nachgewiesen.

Kontextinformationen erleichtern dem Benutzer nicht nur den Überblick über einen Informationsraum, sondern sie können ihm auch Hinweise für sein weiteres Rechercheverhalten geben. Diese Eigenschaften von Kontextinformation werden im nächsten Kapitel eingehender beleuchtet.

1.5.2. Orientierungs- und Navigationshilfen

Ein Ansatz zur Vermeidung des Orientierungsverlustes wäre eine derart gute Aufbereitung der Daten, so daß der Benutzer sofort einen Überblick über Umfang und Struktur der Informationsquelle hat. Er wird so in die Lage versetzt, sich die für ihn interessanten Informationen schnell zusammenzusuchen. Ein Beispiel hierzu sind Tageszeitungen, die – in Rubriken unterteilt – selektiv gelesen werden können.

Ist jedoch eine solche Aufbereitung der Daten nicht möglich, so müssen die relevanten Informationen gezielt gesucht werden (Information Retrieval) oder der permanente Informationsfluß muß nach relevanten Elementen gefiltert werden (Information Filtering), um den Umfang des Informationsraumes in irgendeiner Form überschaubar zu halten.

Es ist jedoch oft der Fall, daß viel mehr Informationen vorhanden sind, als dem Benutzer auf einmal (z.B. auf einer Bildschirmseite) präsentiert werden können. Eine weitere Unterstützung für den Benutzer bieten Navigations- bzw. Orientierungshilfen, wie sie in verschiedenen Hypertextsystemen zu finden sind. Sie sollen dem Benutzer folgende Hilfen bieten:

- Er soll jederzeit einen Überblick über den gesamten Hypertext haben. Damit kann er die Größe und Anzahl der Informationseinheiten abschätzen. Weiterhin erleichtert es die Orientierung, wenn jederzeit eine Gesamtübersicht im Auge behalten werden kann.

- Die Strukturinformationen, also die Anordnung der Informationseinheiten, sowie ihre Verknüpfungen untereinander, tragen ebenfalls zu einer besseren Übersicht bei.

- Der Benutzer soll jederzeit in der Lage sein, seinen eigenen Pfad zurück zu verfolgen. Er gewinnt dadurch mehr Vertrauen in seine Orientierungsfähigkeiten und wird sich beim Umherstreifen im Hypertext sicherer fühlen.

Das folgende Kapitel beschäftigt sich mit Methoden und Verfahren, die dem Benutzer derartige Orientierungshilfen bieten.

2. Metainformation und gezielte Navigation

Welche Möglichkeiten können dem Benutzer an die Hand gegeben werden, um ihm seine Informationssuche zu erleichtern, bzw. erst zu ermöglichen ?
Einerseits können ihm zusätzliche (Meta-) informationen über die Datenbasis wertvolle Hinweise zum gezielten Retrieval geben, zum anderen unterstützen ihn Navigationssysteme und Orientierungshilfen beim explorativen Browsing.

2.1. Anzeige der Retrievalergebnisse mit Kontextinformation

Die Ergebnisse, die eine Suchmaschine dem Benutzer auf eine Anfrage liefert, sollen auf verschiedene Weise aufbereitet bzw. angereichert werden.

- Die Ergebnislisten sollten möglichst redundanzarm sein, d.h. daß auf eine Anfrage nicht sofort jede einschlägige Informationseinheit (z.B. WWW-Seite) nachgewiesen werden soll, sondern erst, wenn der Benutzer derart umfangreiche Antwortmengen auch anfordert, er also einen hohen Recall erreichen will.

- Die Ergebnisse sollen mit ihrer explizit im World Wide Web vorhandenen Hypertextstruktur visualisiert werden. So erhält der Benutzer wichtige Hinweise, anhand derer er die Relevanz der Dokumente besser einschätzen kann.
  Sind z.B. mehrere Trefferseiten von einer einzelnen Seite aus verlinkt, so sind mit hoher Wahrscheinlichkeit auch die übrigen Verweise dieser Seite für die Suche von Interesse.

- Durch diese kontextualisierte Anzeige der Suchergebnisse kann der Nutzer auch Hinweise auf weiteres Navigationsverhalten oder neue Schlagworte für eine erneute Recherche erhalten. Ebenso erleichtert sie ihm die Übersicht über den Informationsraum.

Im Konstanzer Projekt ”Virtuelle Informationsräume” (VIR[1]) [Kuhlen et al. 96] wird ein Ansatz bevorzugt, bei dem die Suche in einem regional oder inhaltlich begrenzten, elektronischen Marktplatz durchgeführt wird. Ziel des Projektes ist, dem Benutzer eines solchen Marktplatzes nicht Treffer von einzelnen WWW-Seiten, sondern die Hypertextstruktur der relevanten Anwendung (z.B. eine Unternehmensdarstellung innerhalb eines elektronischen Marktplatzes) visualisiert und kontextualisiert anzuzeigen.

Zentrale Leistung der im Projekt VIR aufzubauenden Lösung ist nicht eine oder mehrere isolierte Informationseinheiten nachzuweisen, sondern die Einheiten durch Einbettung in ihren Kontext darzustellen. Dabei werden die Schwerpunkte auf die Auswahl der Informationsquellen, die dezentrale Suche, die Identifikation der Struktur, in der die relevanten Treffer eingebettet sind, sowie die Visualisierung dieser Strukturen gelegt.

2.1.1. Gewinnung und Nutzung der Strukturinformationen

Die Strukturdaten, die die Visualisierungsumgebung benutzt, werden von einem Roboter innerhalb des regional oder fachlich abgegrenzten, elektronischen Marktplatzes zur Verfügung gestellt. Der Roboter durchsucht die vorhandenen Informationsquellen systematisch, indem er von einer Startseite ausgehend den Verknüpfungen innerhalb des elektronischen Marktplatzes folgt und auf diese Weise eine Hierarchie der Informationseinheiten erstellt [Scheuch97]. Information-selemente, die nicht zum aktuellen Marktplatz gehören, werden von der Struktur-erkennung nicht erfaßt.

Ein Suchergebnis besteht also aus Treffern innerhalb mehrerer, in sich semantisch abgeschlossener WWW-Hypertexte. Solche abgeschlossenen Hypertexte sind meist ein Merkmal kontrollierter WWW-Bereiche wie z.B. elektronischer Märkte. Die WWW-Hypertexte unterliegen dabei einer hierarchischen Struktur, beginnend mit der Homepage.

[1] http://www.inf-wiss.uni-konstanz.de/FG/IV/VIR/index.html

Ein Pfad, beginnend von der Homepage zu einer beliebigen Seite innerhalb des WWW-Hypertextes ist nicht unbedingt eindeutig, jedoch ist eine Kategorisierung der Pfade durch die hierarchische Nähe zur Homepage möglich. Hier wird dabei nur der hierarchisch kürzeste Weg von einer Seite zur Homepage für die Strukturbeschreibung benutzt. Aber auch alle anderen möglichen Pfade werden über sogenannte Querverweise beschrieben, haben aber keinen Einfluß auf die Hierarchie bzw. Struktur der Hypertexte.

Das Kernziel der graphischen Darstellung der Suchergebnisse ist, dem Benutzer eine qualitativ bessere und schnellere Relevanzbeurteilung zu ermöglichen. Entscheidungsrelevante Informationen werden meist aus der Betrachtung der Gesamtheit und der Beziehungen zwischen einzelnen WWW-Trefferseiten bzw. deren Kontext erst ersichtlich. Der Einsatz geeigneter graphischer Darstellungsmittel gilt als eines der besten Verfahren zur Schaffung von räumlichem Kontext [Utting&Yankelovich 89] und wird hier zur Treffervisualisierung benutzt (aus [Bekavac&Rittberger96]).

2.2. Visualisierungstechniken für große Hypertexte

Graphen werden sowohl zur Visualisierung des dynamischen Verhaltens von Systemen (z.B. das Verhalten eines Programmes zur Ausführungszeit), als auch zur Darstellung von statischen Sachverhalten (Aufbau- oder Ablaufstrukturen) verwendet. Heutzutage auftretende Softwaresysteme sind aber bei weitem zu komplex aufgebaut, als daß ein Graph zu ihrer Visualisierung halbwegs passend und leserlich auf einem Monitor dargestellt werden kann.

Aus diesem Grunde sind Methoden zur Reduzierung der Komplexität dieser großen Informationsräume notwendig. Die Anwendung dieser Methoden ist keinesfalls auf die Behandlung von (Hypertext-) Graphen beschränkt, sie sind auf breiter Ebene anwendbar. Mit ihrer Hilfe behalten wir den Überblick über komplexe Strukturen, sei es im natur- oder sozialwissenschaftlichen Bereich oder im täglichen Leben. Auch sind sie keineswegs erst in neuester Zeit entstanden, das später erläuterte Konzept der Abstraktion findet sich bereits bei Julius Cäsar: Divide et impera – teile und herrsche.

2.2.1. Modifikation der Hypertextgraphen

Folgende Verfahren beschäftigen sich mit der Modifikation von Graphen, mit dem Ziel sie zu verkleinern und so überschaubarer zu machen. Sie sind also als Möglichkeiten zur Komplexitätsreduktion zu sehen.

Dies hat nicht nur den Vorteil, daß der ursprüngliche – eben oft zu komplexe – Graph so in einer für den Benutzer übersichtlicheren, besser lesbareren Form dargestellt wird. Die Algorithmen zum Layout und zur Anzeige des Graphen erfahren eine deutliche Performancesteigerung, was zu einer spürbaren Reduzierung der Antwortzeiten des Systems führt und so zu einer erhöhten Benutzerakzeptanz beitragen kann.
Werden dynamische Systeme visualisiert, bei denen eine definierte Antwortzeit gefordert ist (z.B. in der Automatisierungstechnik), können solche Verfahren für den Erfolg eines Informationssystems entscheidend sein.

Abstraktion

Eine Möglichkeit zur Reduktion der Komplexität eines Graphen heißt Abstraktion. Es wird ein abstrakter Graph erzeugt, der nur noch relevante Knoten enthält, alle weiteren werden ausgeblendet. Die sowohl in den Knoten als auch in der Struktur enthaltene Information wird so vor irrelevantem ”Nebenrauschen” geschützt und dem Benutzer präsentiert. Dieser kompakte Graph hat den weiteren Vorteil, daß er nur bei Änderungen in den relevanten Knoten neu visualisiert werden muß, andere Änderungen können zurückgestellt werden.

[Kimelman et al. 94] beschreiben mehrere Verfahren für das ”Verstecken” irrelevanter Knoten:

- Ghosting:
  Die Koten werden in einer der Hintergrundfarbe ähnlichen Farbe gezeichnet, so daß der Benutzer sie zwar noch erkennen kann, sie somit aber deutlich als nicht relevant gekennzeichnet sind.

- Hiding:
  Hier werden die irrelevanten Knoten vollständig ausgeblendet.

- Grouping:
  Mehrere Knoten werden unter einem ”Meta-Knoten” zusammengefaßt.

Clustering:

Unter Clustering versteht man das Anordnen der Knoten eines großen Graphen in Gruppen, indem man bestimmte strukturelle Gleichheiten von Knoten identifiziert und sie anhand dieser Eigenschaften zusammenfaßt. Diese strukturellen Gleichheiten sind anwendungsabhängig und können frei definiert werden.

[Sablowsky&Frick96] beschreiben einen Algorithmus, der dieses Verfahren auf einen Graphen von WWW-Seiten anwendet.

Partitionierung

Im Gegensatz zur Abstraktion wird hier der Graph nicht mehr als Ganzes dargestellt, sondern er wird in Teilgraphen zerlegt, die dann einzeln visualisiert werden. Dieser sogenannte ”Divide-and-Conquer”-Ansatz wird in [Wang&Miyamoto95] vorgestellt.

Folding

Hinter diesem Begriff verbirgt sich eine Art Zusammenfalten (to fold) des Graph. Es lassen sich Teilbäume per Mausklick aus- und wieder einblenden. Der Benutzer ist also in der Lage selbst den Graphen auf eine für ihn handliche Größe zu reduzieren, indem er sich für einen bestimmten Bereich im Graph entscheidet und die restlichen Knoten und Kanten der Übersichtlichkeit halber ausblendet. Es kann so eine kompaktere Ansicht des Graphen geschaffen werden.

Unterschiedliche Layouttypen

Je nach Anwendungsgebiet, in dem Graphen visualisiert werden müssen, können unterschiedliche Layouttypen sinnvoll zur Verwendung kommen. Auch die Größe oder die Form eines Graphen kann entscheidend für ein Layout sein. [Madden et al. 95] stellen vier unterschiedliche Layouttypen vor, die in dem Softwaresystem ”Graph Layout Toolkit[2]” realisiert wurden.

[2] http://www.tomsawyer.com

Die Abbildungen 1 bis 3 zeigen diese verschiedenen Layouttypen; es wird beschrieben, welche dieser Layouts sich zur Darstellung bestimmter Typen von Graphen besonders eignen.

- Kreisförmige Anordnung der Knoten:

Dieses Layout eignet sich zur Darstellung von ring- oder sternförmigen Netzwerken. Die Knoten werden in logische Gruppen eingeteilt, sie können z.B. nach IP-Adressen gruppiert werden (Clustering). Diese Gruppen werden in Abhängigkeit der zwischen ihnen bestehenden Verbindungen auf einem Kreisbogen plaziert.

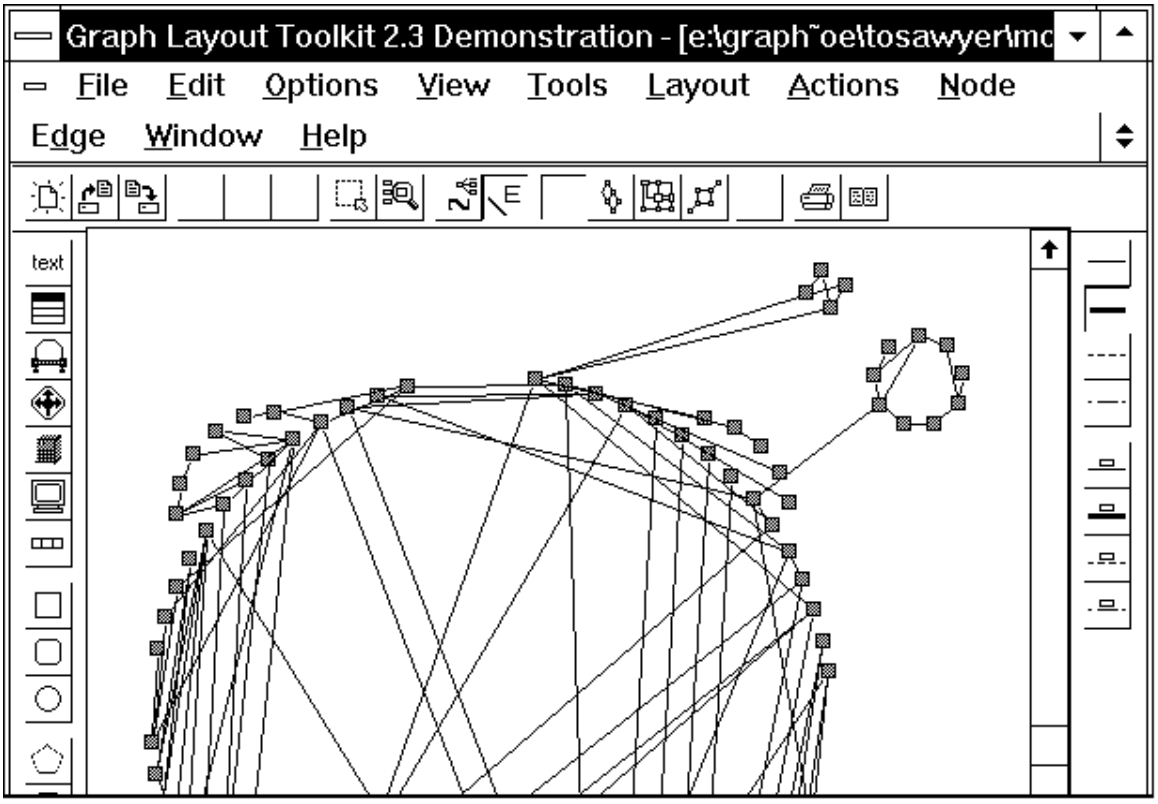

Abbildung 1: Kreisförmiges Graphenlayout

- Hierarchisches Layout:
  Hiermit werden gerichtete Graphen dargestellt, wie sie z.B. bei der Softwareentwicklung oder im Projektmanagement vorkommen, also zur Visualisierung von Aufbau- und Ablaufstrukturen.

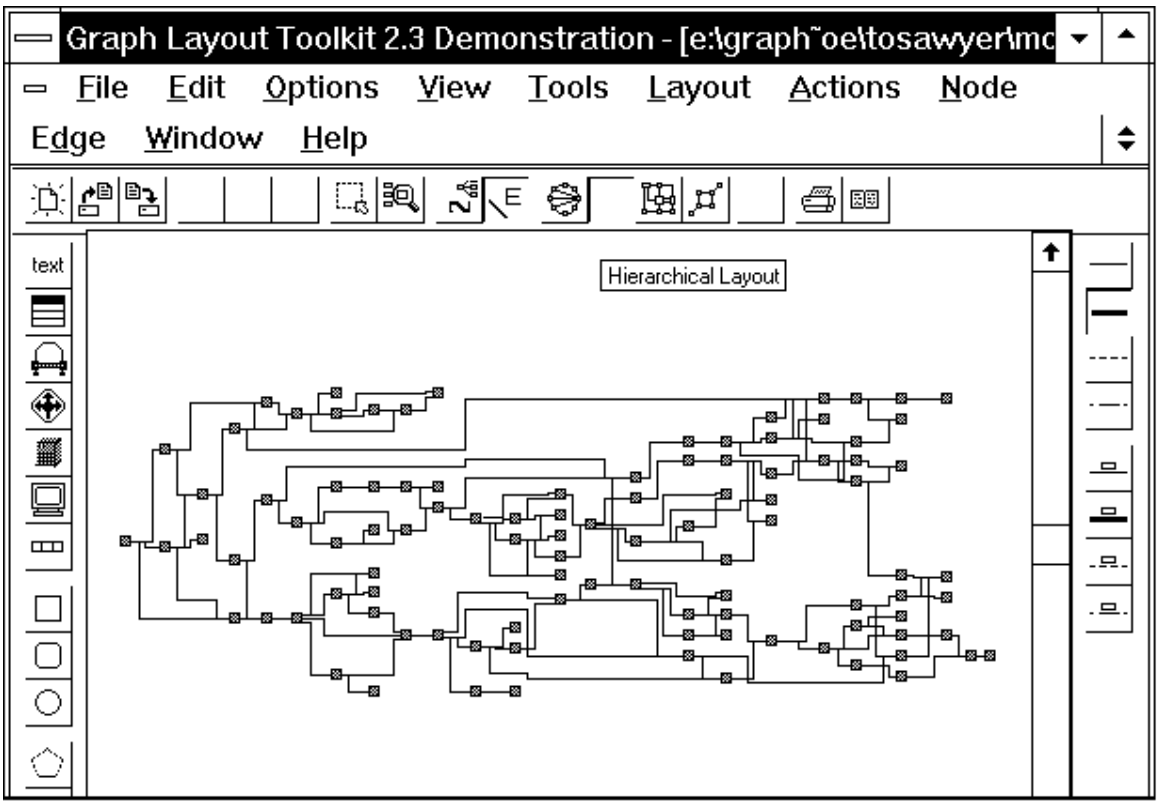

Abbildung 2: Hierarchisches Graphenlayout

- Orthogonales Layout:
  Es eignet sich zur Darstellung von Graphen beim objektorientierten Softwareentwurf, beim Datenbankdesign oder beim Computer Aided Design, also immer dann, wenn es darum geht, Komponenten mit ihren wechselseitigen Beziehungen im Gesamtsystem anzuzeigen.

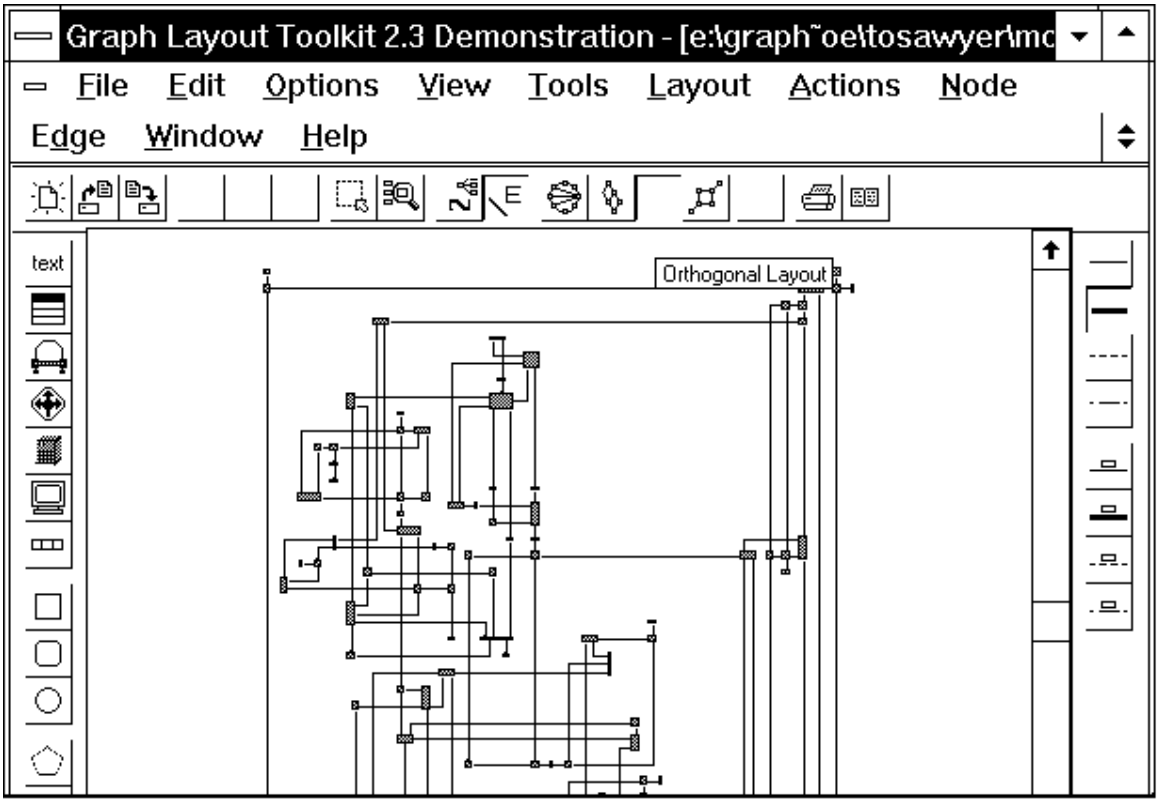

Abbildung 3: Orthogonales Graphenlayout

- Symmetrisches Layout:
  In dieser Form können alle Arten von Netzwerken visualisiert werden. Es wird durch eine möglichst gleichmäßige Verteilung der Knoten und Kanten ein symmetrisches bzw. gleichförmiges Aussehen des Graphen erreicht.

Eine Abbildung zur Darstellung des symmetrischen Layouts war leider nicht zu bekommen, da das Graph Layout Toolkit der Firma TomSawyer immer bei der Anwendung dieser Darstellungsart abstürzte.

Die in diesem Abschnitt beschriebenen Konzepte sind zum größten Teil allgemein gültige Prinzipien. Jetzt werden anwendungsneutrale Methoden zur Navigation und konkrete Verfahren zu deren Umsetzung beschrieben.

Die begriffliche Unterscheidung zwischen Prinzip, Methode und Verfahren ist an [Balzert96] angelehnt, konsequent wird auch auf die Unterscheidung zwischen den Begriffen Methode und Verfahren verzichtet.

2.2.2. Navigationsinstrumente bzw. Orientierungshilfen

Ist der Graph immer noch zu groß, als daß er als Ganzes auf eine Bildschirmseite paßt, braucht der Benutzer eine Möglichkeit zur Navigation, also sich innerhalb des Informationsraumes kontrolliert bewegen zu können (freies Browsing). [Kuhlen91] definiert Navigation als Oberbegriff für das für Hypertext typische ”Wandern” in Hypertexträumen, das durch das im Information Retrieval übliche gezielte Suchen ergänzt wird.

Navigationsinstrumente müssen einfach zu erlernen und zu bedienen sein, denn auch sie bedeuten für den Benutzer eine kognitive Überlast. Von dem Zeitpunkt ab, an dem der Benutzer zu einer bestimmten Position im Hypertext navigieren will, bis er diese Position erreicht, ist alles kognitive Überlast.

Nachfolgend werden Systeme und Konzepte zur schnellen und effektiven Navigation sowie zur besseren Orientierung in großen Hypertexten vorgestellt. Es wird besonderer Wert auf Techniken und Verfahren für Übersichtsdiagramme gelegt, aber auch allgemeine Navigationsmechanismen, die auch ohne Übersicht über den gesamten Hypertext verwendet werden können, werden erwähnt.

Scrollbars und Zoom

Scrollbars stellen das einfachste und bekannteste Instrument zur Navigation dar. Mit ihrer Hilfe blättert man leicht in Texten oder überblickt Zeichnungen, welche einen Umfang von mehreren Bildschirmseiten haben können.
Sie werden allerdings bereits bei mäßig großen Datensammlungen unhandlich und sind somit nicht als ”Navigations” - instrumente zu bezeichnen, da der Benutzer immer ungefähr wissen muß, wo er sich im Informationsraum befindet, um weiter navigieren zu können.

Einfaches Zoomen in ein Informationsobjekt, also eine Vergrößerung eines bestimmten Bereichs, ist eine notwendige Maßnahme, um in komplexen, umfangreichen Strukturen wesentliche Details überhaupt erkennen zu können. Je größer der Text oder die Zeichnung, desto schneller verliert man sich in der vergrößerten Ansicht. Es muß ”zurückgezoomt” werden, um sich wieder einen Überblick zu verschaffen, wo man sich eigentlich befindet und wohin man weiter navigieren will.

Beiden Techniken haftet also der Makel einer mangelnden Übersicht über den gesamten zu betrachtenden Informationsraum an, es sind komfortablere und effektivere Verfahren gefragt.

Roaming und Zooming mittels MapWindows

In einem MapWindow wird - zusätzlich zur normalen Visualisierung - eine Miniaturansicht des gesamten Informationsraumes (Hypertextes) angezeigt. Der Benutzer sieht also gleichzeitig einen Ausschnitt aus dem zu betrachtenden Informationselement in Originalgröße und einen Gesamtüberblick.

In diesem MapWindow ist mit Hilfe eines Rahmens (wire frame box) die aktuelle Position der Normalansicht auf dem Informationselement gekennzeichnet. Diese Position kann nun einfach per direkter Manipulation mit der Maus in dem MapWindow verändert werden. Der Benutzer kann also schnell im Informationsraum umherstreifen (Roaming) und behält immer den Überblick, wo er sich aktuell befindet.

Weiterhin besteht die Möglichkeit diesen Rahmen im MapWindow in neuer Größe und Form neu aufzuziehen (rubber banding), dessen Inhalt wird darauf in eventuell anderer Größe im Anzeigefenster dargestellt. Liegt der neue Rahmen innerhalb des alten Rahmens, so wird ein Ausschnitt daraus vergrößert, umgekehrt wird verkleinert.

In [Beard&Walker90] wird ein Experiment beschrieben, welches nachweist, daß diese Roaming- und Zoomingtechnik mit Hilfe des MapWindows den Benutzer effektiver bei der Erforschung eines unbekannten Informationsraums unterstützt als einfache Scrollbars.

Anzumerken bleibt jedoch, daß das Konzept eines MapWindows nur funktioniert, wenn die zu visualisierende Hypertextstruktur nicht zu groß wird. Sind mehr Knoten/Kanten darzustellen, als innerhalb des MapWindows bei einer gegebenen Bildschirmauflösung Pixel vorhanden sind, scheitert der Ansatz.

Eine Implementierung dieses Konzeptes stellt das Programm VGJ[3] (Visualizing Graphs with Java) dar.
Im MapWindow kann vom Benutzer ein sog. Viewing Offset gesetzt werden, eben das Positionieren des Rahmens zur Navigation in der Übersicht. Ein weiteres Feature ist das Konzept der Viewing Angles, bei dem der Benutzer in der Lage ist, den Graph um beliebige Achsen rotieren zu lassen, also eine dreidimensionale Ansicht. Sämtliche Quelltexte stehen im Netz frei zur Verfügung.

Fisheye-Konzepte

In zweidimensionalen Informationsräumen ist sowohl eine detaillierte Ansicht eines speziellen Ausschnittes, als auch eine Gesamtübersicht für den Benutzer von Interesse. Im Gegensatz zu dem bereits vorgestellten Konzept der MapWindows, bieten sogenannte Fish-Eye-Views eine Integration dieser beiden Ansichten innerhalb eines Fensters. Ein Fish-Eye-View erlaubt eine vergrößerte Ansicht eines kleinen Teils des Graphen, während der gesamte Graph auf dem Bildschirm sichtbar ist.

”Unter dem Begriff Fish-Eye-View lassen sich Darstellungsformen verstehen, die sowohl lokale Details als auch globale Zusammenhänge gleichrangig berücksichtigen. Die globalen Zusammenhänge sollen dem Betrachter zeigen, welche Alternativen ihm für die weitere Interaktion zur Verfügung stehen, bzw. wie diese zu erreichen sind. Die lokalen Details dienen zur lokalen Interaktion. Bei der Beschäftigung mit den lokalen Details kann der globale Zusammenhang im übrigen wichtig sein” (aus [Aßfalg92]).

[3] http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.html

Ein Beispiel für Fish-Eye-Views sind Tageszeitungen, die eine Art ”Fish-Eye”-Sicht auf die Nachrichtenwelt bieten. Lokales Geschehen wird in gleicher Ausführlichkeit behandelt, wie global wichtiges Weltgeschehen. Je weiter ein lokales Ereignis vom Verteilungsgebiet der Zeitung entfernt ist, um so weniger umfangreich wird es behandelt werden.

Fish-Eye-Views zeigen viele Einzelheiten der Informationselemente, die nahe bei dem momentanen Interessenschwerpunkt des Benutzers liegen, und weniger Details von Gebieten des Informationsraums, die weiter entfernt vom momentanen Aufmerksamkeitszentrum liegen. Dazu muß der Informationsraum zwei Eigenschaften aufweisen: Erstens muß es möglich sein, die Distanz zwischen einem Informationselement und dem Aufmerksamkeitsschwerpunkt des Benutzers zu bestimmen. Zweitens muß die Information auf verschiedenen Abstraktionsebenen darstellbar sein (vergleiche auch [Nielsen95]).

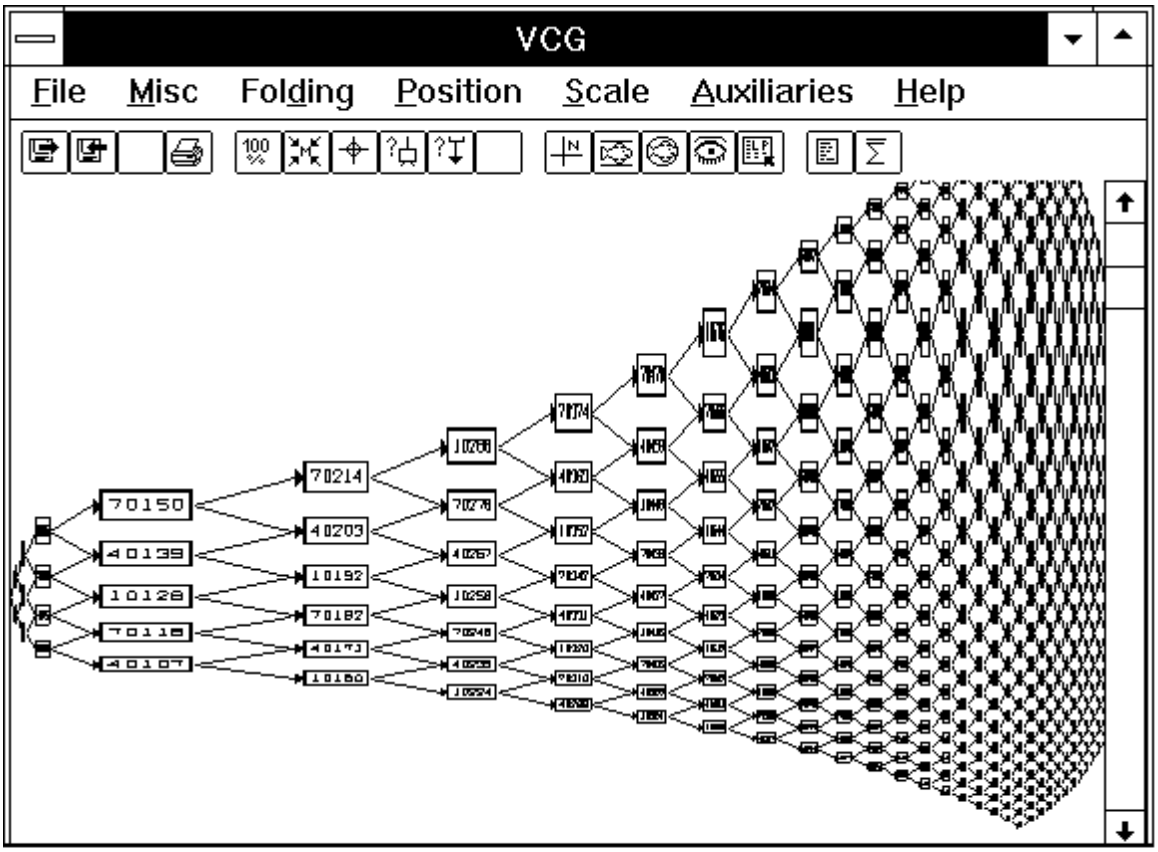

Abbildung 4: Kartesischer Fisch-Eye-View

Verschiedene Layouts für Fish-Eye-Views findet man in [Storey&Müller95]; Beschreibungen von unterschiedlichen Algorithmen zur deren Berechnung und weitere Beispiele in [Formella&Keller95], [Kaugars et al. 94] sowie [Saxer90].

Das Programm VCG[4] (für Windows) stellt dem Benutzer sowohl polare wie kartesische Fish-Eye-Views zur Verfügung und eignet sich somit besonders zur Demonstration dieses Konzepts. Die beiden Abbildungen **4** und **5** zeigen diese zwei Arten von Fish-Eye-Views.

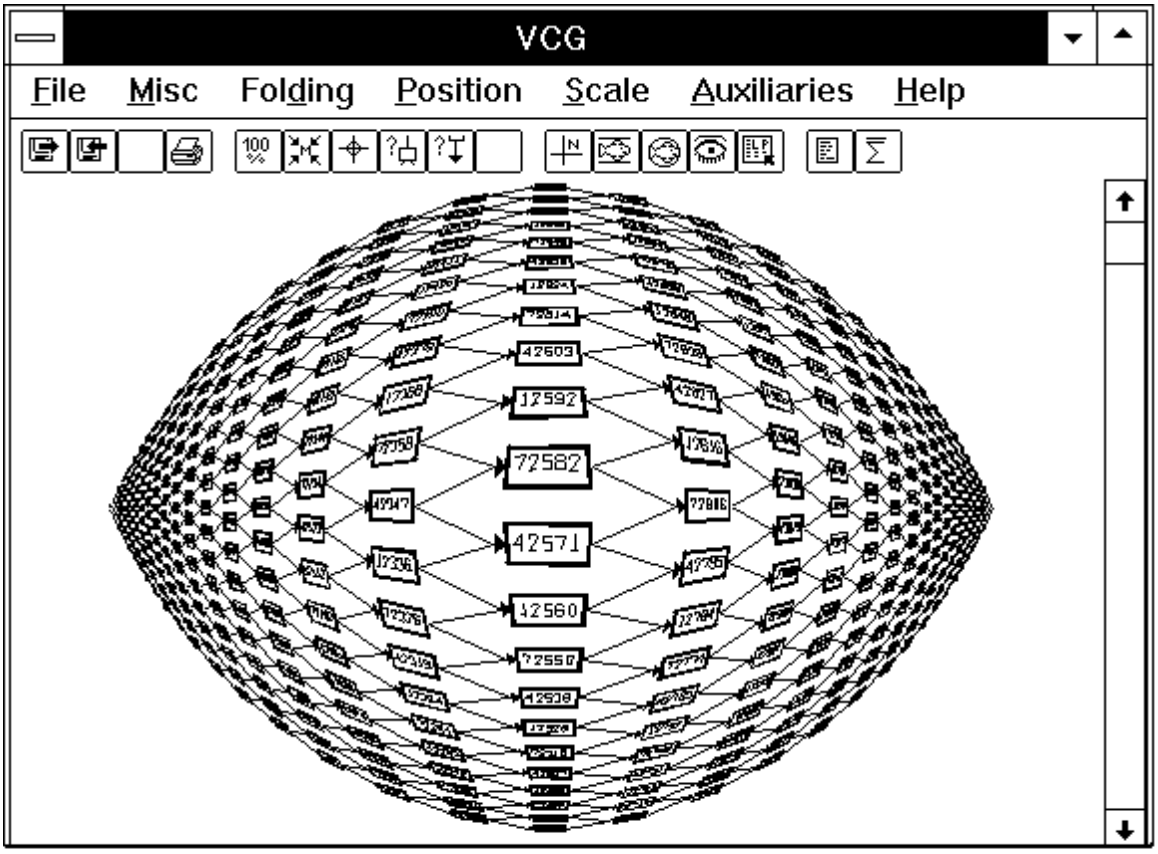

Abbildung 5: Polarer Fish-Eye-View

[4] http://www.cs.uni-sb.de/RW/users/sander/html/gsvcg1.html

Structural Navigators

Hinter diesem Begriff steht das Konzept eines Navigierens anhand der Struktur des Graphen. Einen derartigen Navigator stellt das Produkt daVinci[5] der Universität Bremen zur Verfügung [Fröhlich&Werner94].

Ausgehend von einem Knoten stellt der Navigator dem Benutzer verschiedene Möglichkeiten zur Verfügung. Er kann direkt eine Hierarchieebene nach oben/unten wechseln (Next/Prev Level), oder aber zum nächstgelegenen Vater-/Kindknoten (Nearest Parent/Child). Dies kann bei der Navigation in Graphen sehr nützlich sein, im Gegensatz zu Bäumen, bei denen durch die explizite Hierarchie die Übersicht gewährleistet ist.
Ebenso die Navigationsmöglichkeiten Left/Right Neighbor und Left/Right Sibling, die sich insofern unterscheiden, als das Left/Right Sibling nur zu Zwillingsknoten navigiert, man sich mit Left/Right Neighbor aber quer über den ganzen Graphen bewegen kann.

Spider Views

Spider Views sind Übersichten über einen Graphen bzw. Hypertext, bei denen sich ausgehend von einem Knoten interaktiv die Anzahl der angezeigten Hierarchie-ebenen einstellen läßt. Je mehr Ebenen angezeigt werden, desto detaillierter wird die Übersicht, der Graph wächst. Die Knoten sind um den ausgewählten Anfangsknoten kreisförmig (wie bei einem Spinnennetz) angeordnet.

Landmarks

Mit Landmarks können bestimmte Positionen im Informationsraum gekennzeichnet werden, die sich dem Benutzer einprägen und ihm ein Wiedererkennen dieser Stellen ermöglichen. Sie ermöglichen so dem Leser, sich schneller in der Übersicht zurechtzufinden. Ein Beispiel sind auf einem Stadtplan besonders hervorgehobene Plätze oder Gebäude (Eifelturm in Paris), anhand derer Tourist sich in einer Stadt orientieren kann.

---

[5] http://www.informatik.uni-bremen.de/~inform/forschung/daVinci/daVinci.html

Backtracking

Das Zurücksetzen ist eine der wichtigsten Navigationshilfen. Es bringt den Benutzer zu den vorherigen Knoten zurück. So lassen sich Hypertextsprünge rückgängig machen, wodurch der Benutzer eine Art Rettungsleine an die Hand bekommt. Sie gibt ihm die Sicherheit, immer wieder zu bereits besuchten Informationseinheiten bzw. Knoten zurückzufinden, gleichgültig wohin sich der Leser im Hypertextlabyrinth auch wendet (Ariadne-Faden).

Die BackTrack-Funktion muß zwei Mindestanforderungen genügen [Nielsen95 S.245]: sie sollte immer zugänglich sein und immer auf die gleiche Art aufgerufen werden. Außerdem sollte es immer möglich sein, so lange rückwärts zu gehen, bis man wieder am Einstiegsknoten ankommt.

Backtracking-Mechanismen funktionieren nicht immer auf die gleiche Weise, derartige Inkonsistenzen können beim Leser zu großen Verwirrungen führen. Schwierigkeiten entstehen dann, wenn der Benutzer über mehrere Schritte zurücksetzen will und er davor einige Knoten mehrfach aufgesucht hat. [Nielsen95] beschreibt allein fünf verschiedene Möglichkeiten des Backtrackings.

Pfade - Guided Tours

Als Navigationspfad wird die Sequenz aller bereits besuchten Knoten bezeichnet, anhand derer der Benutzer schnell und einfach zu bestimmten Knoten zurückkehren kann. Er muß im Gegensatz zum Backtracking nicht nacheinander alle Knoten rückwärts abklappern, sondern er kann direkt einen Knoten im Pfad anwählen.

Guided Tours, also geführte Unterweisungen oder Online-Präsentationen, sind zur Unterstützung ungeübter Benutzer in komplexen Hypertextbasen vorgesehen. Sie werden vom Autor vorgegeben, der das Leseverhalten der Benutzer seines Hypertextes antizipieren muß. Aus diesem Grunde muß es an jeder Stelle möglich sein, die Tour zu verlassen, selbständig browsen zu können und danach wieder mit der Tour fortzufahren. [Kuhlen91] beschreibt dieses Konzept ausführlichst.

Bookmarks

Manche Systeme erlauben dem Benutzer Lesezeichen (Bookmarks) zu vergeben, anhand derer er schnell und unkompliziert zu bestimmten Informationseinheiten zurückkehren kann. Der Unterschied zu Landmarks oder Navigationspfaden ist, daß Bookmarks nicht vom Autor vorgegeben werden, sondern genau das Interesse des Lesers widerspiegeln. Lesezeichen können den Namen des Knotens beinhalten oder es können vom Benutzer Kurzbezeichnungen eingegeben werden, die ihm das Wiederfinden erleichtern.

Lesezeichen können als lineare Listen verwaltet werden, was bei einer großen Anzahl allerdings schnell wieder unübersichtlich wird. Programme wie z.B. der Navigator der Firma Netscape erlauben auch geschachtelte, hierarchisch strukturierte Gruppen von Lesezeichen.

Zusatzinformationen über Struktur des Graphen

Verschiedene Systeme (z.B. daVinci) bieten dem Benutzer Informationsfenster, in denen verschiedene Eigenschaften des Hypertextes (Anzahl Hierarchieebenen, Anzahl Knoten, Anzahl Kanten usw.) angezeigt werden können. Besonders bei großen Hypertexten kann sich der Benutzer so schnell einen Überblick über dessen Charakteristika verschaffen.

2.3. Beurteilung und Entscheidung

Eine sinnvolle Kombination beider Verfahren, also sowohl eine Modifikation des zu visualisierenden Graphen, als auch der Einsatz geeigneter Navigationsinstrumente verspricht den größten Erfolg, da beide dem Benutzer wertvolle Hilfen zur effizienten Informationsrecherche bieten können.

Der Benutzer muß allerdings durchschauen können, ob und auf welche Art der Graph modifiziert wurde. Ansonsten ist er nicht in der Lage den Graph in seinem vollen Umfang sichtbar zu machen, oder überhaupt zu erkennen, daß er nicht den ursprünglichen, vollständigen Graphen, sondern ”nur” einen Teilbereich sieht. Ebenfalls sollte man ihn nicht mit einer Vielfalt von Navigationsinstrumenten überschütten, die er alle erst mit erwähnter kognitiven Mehrlast erlernen muß. Für verschiedenen Anwendungsfälle bieten sich unterschiedliche Verfahren zur Navigation an, es muß sauber differenziert werden, welches zu verwenden ist.

Für die Verteilung von Navigations- und Orientierungsfunktionalität auf verschiedene Werkzeuge müssen nach [Hammwöhner95] folgende Kriterien erfüllt sein:

- Es sollen nicht zu viele Werkzeuge gleichzeitig genutzt werden müssen, um den Koordinierungsaufwand nicht zu hoch zu treiben.
- Werden zu wenige Werkzeuge differenziert, ist deren Komplexität zu hoch, da einzelne Werkzeuge viele Funktionen anbieten und durch adäquate Präsentationsformen unterstützen müssen.

Im nächsten Kapitel wird der Designprozeß beschrieben, also wie die zu realisierende Lösung aussehen soll und welche Konzepte oder Verfahren zum Einsatz kommen und welche nicht.

3. Methodenauswahl und Grobentwurf

In Tabelle 1 werden alle beschriebenen Konzepte und Werkzeuge nochmals aufgelistet und auf ihre Anwendbarkeit im konkreten Problemfall hin, also die Visualisierung und Navigation von komplexen Hypertextgraphen, untersucht.

Konzept	Anwendung
Abstraktion	Das Konzept der Abstraktion, also die ausschließliche Darstellung relevanter Knoten, wird bereits in der Phase der Recherche angewendet, da der Suchroboter nur relevante Knoten und die zugehörige Strukturinformation an die Visualisierung weitergibt. Abstraktion braucht also nach der Recherche nicht mehr berücksichtigt zu werden. Um dem Benutzer allerdings die Unterscheidung zwischen Struktur- und Trefferknoten zu vereinfachen, können die Trefferknoten durch eine Färbung noch hervorgehoben werden.
Clustering	Ein Clustering, also das Zusammenfassen von Knoten nach frei definierbaren, strukturellen Gesichtspunkten ist in diesem konkreten Anwendungsfall nicht notwendig bzw. sinnvoll, da die Elemente bereits in einer Hierarchie angeordnet sind und eine weitere Gruppierung überflüssig wäre.
Partitionierung	Die Partitionierung ist bereits im bestehenden Prototyp implementiert. Teilbäume lassen sich separat anzeigen und auch beliebig in weitere Teilbäume zerlegen. Dieses Konzept wird ausführlich im Abschnitt 4.2. Metanavigation beschrieben.

Folding	Dieses Konzept könnte in das zentrale Ansichtsfenster integriert werden. Der Graph könnte vor der Anzeige derart zusammengefaltet werden, daß er den Umfang des Fensters nicht mehr wesentlich überschreitet, d.h. er sollte beispielsweise nicht mehr als doppelt so breit wie das Fenster sein. Über Graphen dieser Größe läßt es sich noch leicht per Scrollbar browsen.
Layouts	Unterschiedliche Layouttypen werden nicht verwendet, da zur Darstellung einer hierarchischen Datenstruktur sich ein hierarchisches Layout am besten eignet, da es die Eigenheiten dieser Struktur am ehesten in den Vordergrund zu stellen vermag.
Instrumente	
ScrollBars	Scrollbars werden vom Prototyp bereits zur Verfügung gestellt. Unter Verwendung des Foldingkonzepts könnte ihr Wert als Navigationsinstrument wieder in den Vordergrund treten. Der Baum wird so komprimiert, daß er den Umfang des Fensters, in dem er angezeigt wird, nicht mehr wesentlich überschreitet und so die Scrollbars sinnvoll angewendet werden können.
MapWindows	MapWindows eignen sich am besten als Navigations-instrumente für große Hypertextgraphen. Sie bieten dem Benutzer immer eine Übersicht über den gesamten Graphen in dem er sich bewegt, er kann dort direkt navigieren bzw. in den Baum hinein- und herauszoomen. MapWindows sind intuitiv bedienbar, weil der Benutzer die Struktur des Graphen sofort erkennt und die Navigation per Mausklick sofort einleuchtet.

FishEye-Konzept	Es würde sich ebenfalls gut zum explorativen Browsen auf großen Graphen eignen. Als zusätzlicher Vorteil würde der Benutzer sich nur auf ein Fenster konzentrieren müssen, nicht auf mehrere wie bei den MapWindows. Allerdings würde die Implementierung weitaus aufwendiger werden und die Performance der Anwendung deutlich sinken, da die Algorithmen für solche verzerrten Ansichten wesentlich komplexer sind [Formella&Keller95].
Structural Navigators	Sie werden bei der Realisierung nicht berücksichtigt werden, da die von ihnen gebotenen Möglichkeiten zur Navigation weitgehend von den MapWindows abgedeckt werden.
Spider Views	Diese Technik wird in die MapWindows zusätzlich integriert werden. So hat der Benutzer noch die Möglichkeit, sich deren Inhalte in unterschiedlichen Datailierungsstufen anzuzeigen, was besonders bei sehr großen Graphen deutlich zur Übersicht beitragen kann.
Landmarks	Landmarks zu vergeben, wäre nur sinnvoll, wenn man dafür automatisch markante und hochrelevante Knoten identifizieren könnte. Man könnte sich zumindest vorstellen, diejenigen Knoten zu markieren, innerhalb derer die bei der Recherche verwendeten Suchbegriffe übermäßig oft vorkommen. Diese Konzept wird in dieser Arbeit aber nicht weiter berücksichtigt werden.
BackTracking	Backtracking-Mechanismen werden vorerst nicht realisiert werden; viel mehr sollten sie durch die übrigen Instrumente überflüssig gemacht werden. Ebenso wie die Partitionierung wird von der Metanavigationskomponente, die bereits im Prototyp enthalten ist, auch ein Backtracking-Mechanismus zur Verfügung gestellt.

Guided Tours, Bookmarks oder Zusatzinformation	Diese Konzepte werden ebenfalls nicht in diese Arbeit einfließen, können in späteren Stadien des Projektes bei Bedarf noch zusätzlich Verwendung finden.

Tabelle 1: Methodenauswahl

Anhand der obigen Kriterien fiel die Entscheidung für die Realisierung des MapWindow-Konzeptes. Es verspricht einen deutlichen Beitrag zur Benutzer-unterstützung bei der Navigation und Orientierung und ist doch nur so aufwendig, daß eine Implementierung im Rahmen dieser Diplomarbeit vollständig durchführbar ist.

Zusätzlich wird noch die Spider-View-Technik mit integriert werden, da sie ebenfalls mit vertretbarem Aufwand realisierbar ist und in Kombination zum MapWindow geeignet scheint, dem Benutzer bei der Informationssuche in großen Hypertext-graphen zu unterstützen. Im MapWindow werden die Trefferknoten noch farblich gekennzeichnet werden, um sie von den Strukturknoten besser unterscheidbar zu machen.

Im Rahmen dieser Arbeit wird ein MapWindow für eine Gesamtübersicht mit allen beschriebenen Features zur Navigation implementiert werden. Weiterhin wäre auch ein MapWindow für eine lokale Sicht auf den aktuell vom Benutzer selektierten Knoten denkbar.

Abbildung 6 zeigt einen ersten Entwurf des Bildschirmaufbaus dieser Anwendung, der parallel laufende Browser wird nicht dargestellt.

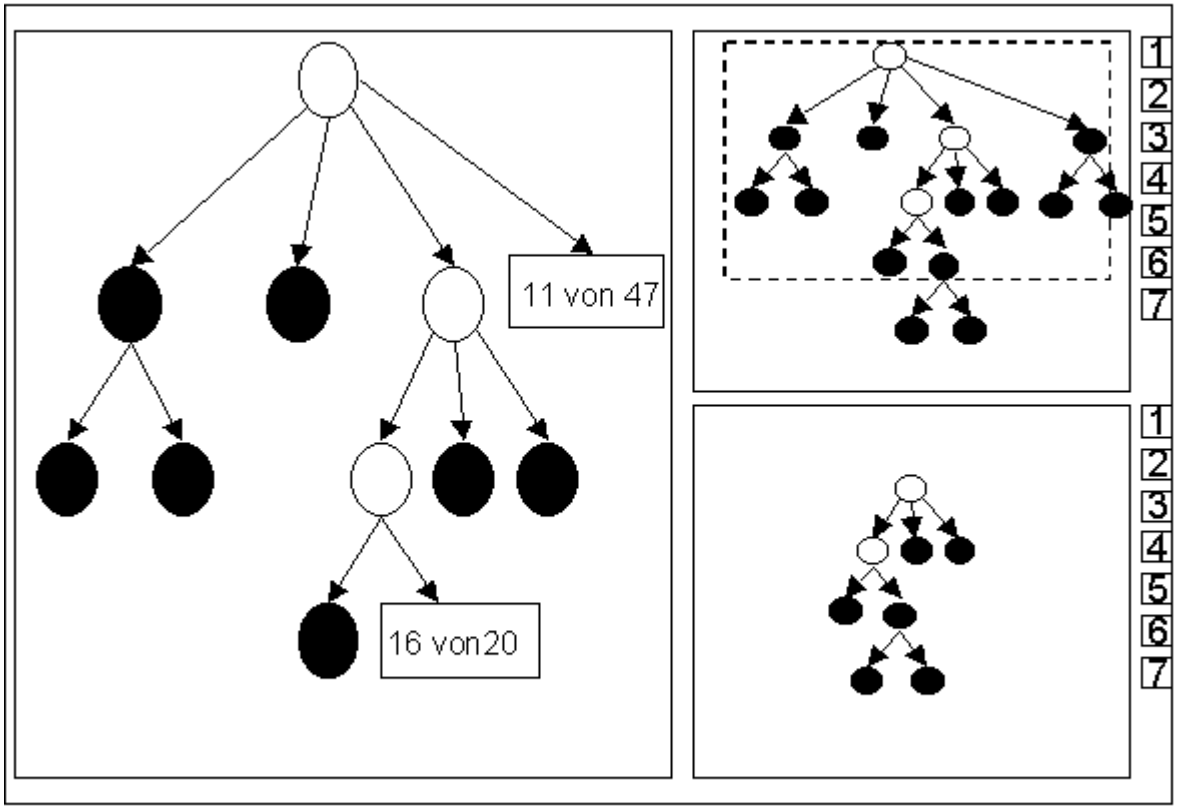

Abbildung 6: Entwurf der Bildschirmmaske

Links befindet sich das zentrale Fenster, in dem der Graph präsentiert wird, der sogenannte Viewer. Hier hat der Benutzer durch direkte Manipulation die Möglichkeit, sich die Inhalte der Knoten (WWW-Seiten) in dem parallel laufenden Browser anzeigen zu lassen. Die Rechtecke deuten das Foldingprinzip an, per Mausklick könnte der verborgene Teilbaum extrahiert werden. Der Benutzer erhält zusätzlich noch die Information, wieviele Knoten des Teilbaumes Treffer und wieviele nur Strukturknoten sind.

Rechts oben ein MapWindow, der sogenannte GlobalMapView, also eine Gesamtübersicht, innerhalb derer durch die gestrichelte Linie der Rahmen angedeutet wird, mit dem der Benutzer auf dem Gesamtgraphen navigieren kann. Daneben als Schaltflächen (Buttons) mit Ziffern 1-7 die SpiderView-Technik. In diesem Beispiel wird im MapWindow der Baum in fünf Hierarchieebenen angezeigt, diese Darstellung erfolgt nach Betätigung des Buttons 5.

Darunter der LocalMapView, eine Ansicht auf den aktuell selektierten Knoten und seinen Kontext, der ebenfalls mit den SpiderView-Buttons mehr oder weniger detailliert angezeigt wird.

4. Entwurf der Anwendung

Die folgenden Abschnitte beschreiben das bereits vorhandene System zur Visualisierung von Graphen. Die Spezifikation und der Entwurf des neu zu implementierenden Softwaremoduls, sowie seine Integration in das bestehende System wird dargelegt.

4.1. Der vorhandene Prototyp und seine Komponenten

Der bestehende Prototyp wurde im Rahmen des Projektkurses ”Suche in verteilten Informationsräumen” im Wintersemester 1996/97 an der Universität Konstanz entwickelt. Mit seiner Hilfe lassen sich bereits umfangreiche Graphen erstellen und verwalten. Der Prototyp stellt den Graphen graphisch dar und bietet dem Benutzer einige Navigations- bzw. Visualisierungsinstrumente.

Der Prototyp besteht aus 13 verschiedenen Klassen, die im folgenden einzeln, sowie in ihrem Zusammenwirken beschrieben werden.

- **ObjectLabel**: Diese abstrakte Klasse definiert das Grundobjekt für diese Applikation, welches nur die Definition einer abstrakten Methode zur Ausgabe des Labels enthält, die von allen abgeleiteten Klassen überschrieben werden muß. Als Label wird in diesem Zusammenhang die Beschriftung des Knotens bezeichnet, die bei der Visualisierung im Bild mit angezeigt wird.
- Davon abgeleitet **GraphObject**, welches eben diese abstrakten Methoden seiner Oberklasse überschreibt und noch zusätzliche Methoden zum Setzen, Auslesen und Löschen des Label bereitstellt.
- **Node** und **Edge** sind die zentralen Objekte der Anwendung, sie sind Unterklassen von GraphObject und enthalten weiterhin die charakteristischen Daten, um ein Informationsobjekt (WWW-Seite) zu repräsentieren. So müssen zu einem Knoten z.B. die URL oder die Menge der von ihm ausgehenden Verlinkungen (Links) zu anderen Knoten gespeichert werden.

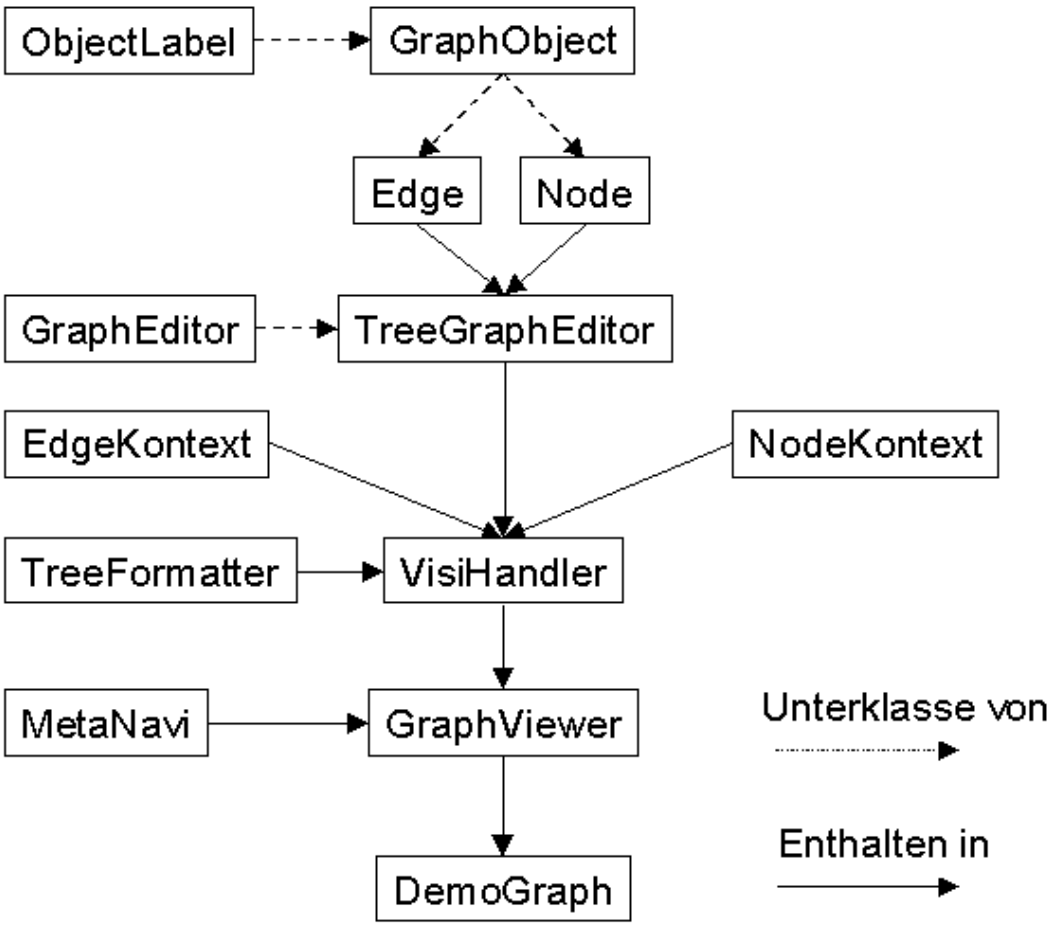

Abbildung 7: Komponenten des Visualisierungsprototyps

- **NodeKontext** und **EdgeKontext** sind Objekte, die für den jeweiligen Knoten (oder Kante) Kontextinformationen speichern, die ausschließlich für die Visualisierung oder Navigation relevant sind, wie die Koordinaten der Knoten, ihre Breite und Höhe zur Anzeige sowie weitere interne Daten.
- Der **GraphEditor** ist das Objekt, welches nun einen vollständigen Graphen und Funktionen zu seiner Manipulation, wie das Einfügen oder Löschen von Knoten in bestehenden Graphen, sowie weitere Hilfsfunktionen bereitstellt.

- Als eine Unterklasse davon ist der **TreeGraphEditor** definiert. Er behandelt den spezielleren Fall von Bäumen und stellt dafür die dem GraphEditor entsprechenden Methoden bereit.
- Der **VisiHandler** stellt nun einen Graphen oder Baum auf einer Anzeigefläche dar und erlaubt auch graphische Interaktionen, wie das Ausblenden oder separate Anzeigen von Teilbäumen. Scrollbars sind als einfache Navigationsmittel vorhanden.
- **MetaNavi** ist ein Objekt, welches das Navigieren in größeren Teilbäumen unterstützt. Der Benutzer, der per Teilbaumansicht (ViewSubTree) sich immer tiefer in die Teilbäume eines Graphen hineinnavigiert hat, bekommt eine Übersicht angezeigt, auf der er immer alle Navigationsschritte angezeigt bekommt und auch direkt wieder zurückspringen kann. Diese Verfahren wird im folgenden Kapitel an einem Beispiel erläutert werden.
- Der **GraphViewer** vereinigt nun ein VisiHandler- und ein MetaNaviobjekt, stellt sie gemeinsam am Bildschirm dar und behandelt die Benutzerinteraktionen.
- **DemoGraph** ist letztendlich das ablauffähige Applet, welches einen zu visualisierenden Graphen aus einer Strukturdatei einliest und darstellt.

Abbildung 7 zeigt eine Übersicht über alle beschriebenen Klassen und ihre Zusammenhänge.

4.2. Metanavigation

Das Konzept der Metanavigation ist bereits im Prototyp enthalten und soll an einem Beispiel genauer beschrieben werden.

Die Abbildungen 8 und 9 zeigen den Bildschirmaufbau des Visualisierungsprototyps. Im oberen Drittel links befinden sich drei Buttons, mit denen der Benutzer sich Teilbäume separat anzeigen lassen kann (View Subtree), Teilbäume ausblenden (Hide), sowie die ursprüngliche Ansicht wiederherstellen kann (Reset). In der Mitte stellt der Metanavigator den Label des Wurzelknotens des aktuellen Teilbaumes dar, in diesem Falle der Root-Knoten ”KHS-System”.

Im mittleren Drittel des Fensters wird der ursprüngliche Graph dargestellt, er besteht im diesem Beispiel aus 11 Knoten und paßt genau in die Anzeigefläche. Das untere Drittel des Fensters dient zur Anzeige von knotenspezifischen Informationen (Titel, URL und Schlüsselwörter), sowie Statusinformationen der Applikation.

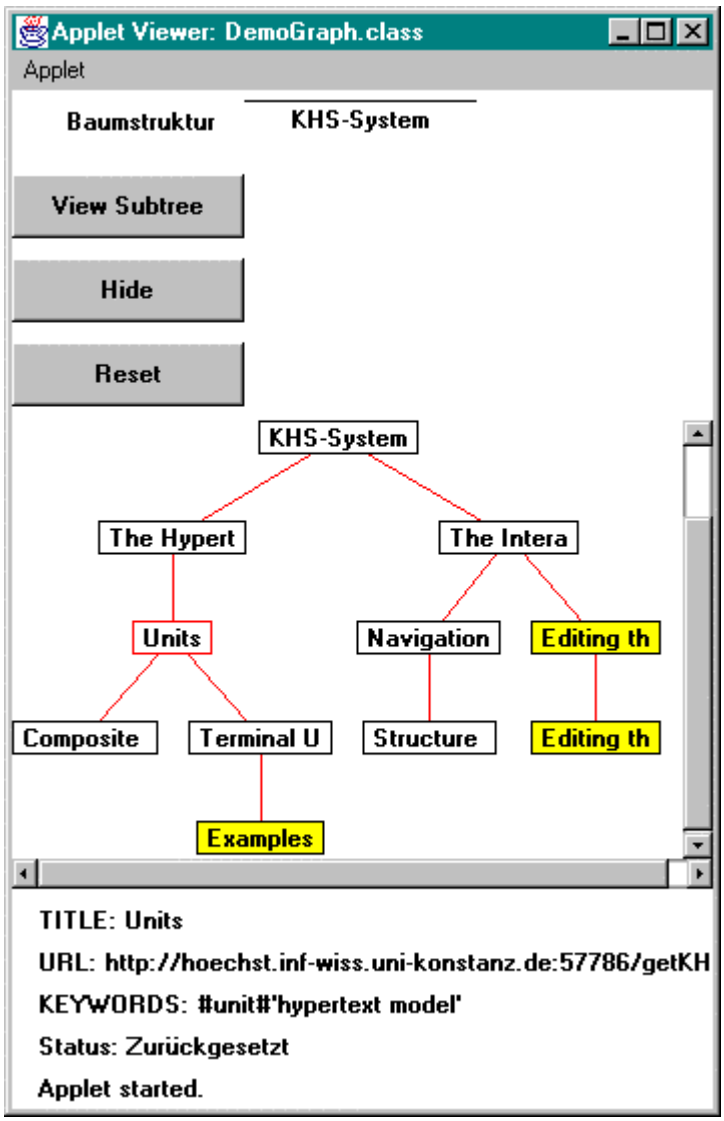

Abbildung 8: Normalansicht eines Graphen

In Abbildung 9 ist erkennbar, daß nur noch ein Teilbaum im Hauptfenster sichtbar ist, der Benutzer sich also den Teilbaum mit dem Wurzelknoten ”Untis” anzeigen läßt.

Im Metanavigator ist jetzt für jeden Teilbaum, in den der Benutzer hineinnavigiert hat, das entsprechende Label sichtbar. Per Mausklick auf eines der im Metanavigator angezeigten Label kann nun direkt wieder in den Teilbäumen ”aufwärts” gesprungen werden.

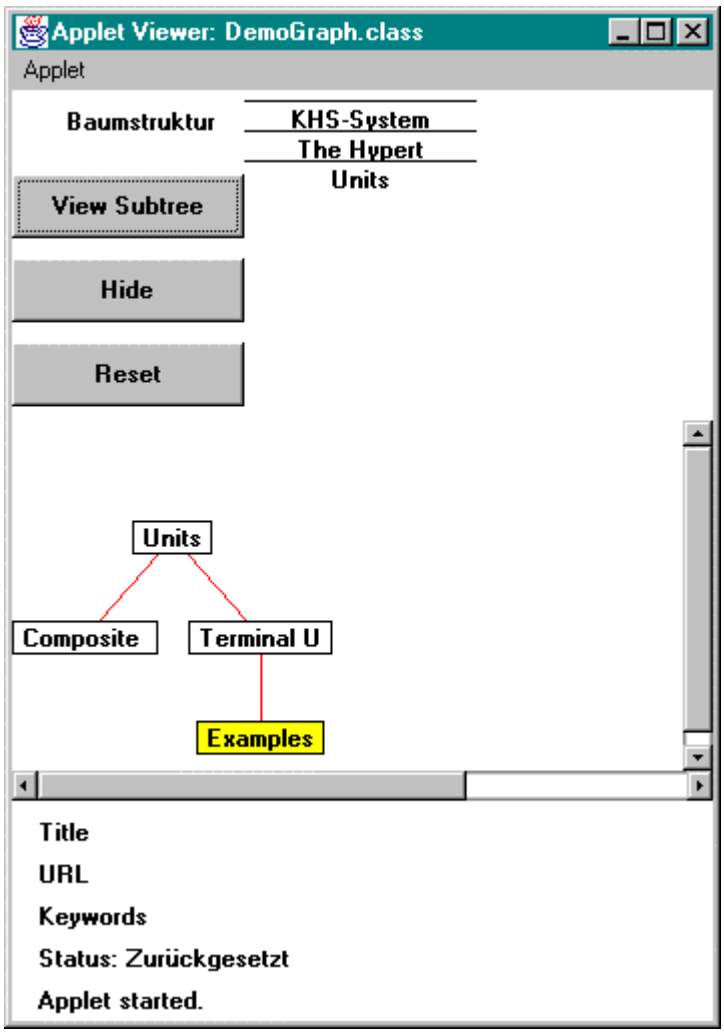

Abbildung 9: Einsatz des Metanavigators

Der Metanavigator ist gerade für größere Graphen bzw. Bäume geeignet, um sich die im Aufmerksamkeitsschwerpunkt liegenden Teile vergrößert darstellen zu lassen. Allerdings muß der Benutzer bei der Ansicht von Teilbäumen immer in etwa wissen, wo er sich im Graphen befindet; er kann leicht die Übersicht verlieren, da ihm diese nicht mehr am Bildschirm angezeigt wird.

4.3. Systementwurf

Ziel des Entwurfs soll eine Klasse sein, die ein Übersichtsfenster zur Verfügung stellt, in dem immer der gesamte Graph angezeigt wird, im Sinne des vorgestellten Konzept eines MapWindows.

4.3.1. MapWindow

In diesem MapWindow soll der Benutzer die Möglichkeit haben, sich interaktiv im Graph zu bewegen (Navigation) und einzelne Ausschnitte vergrößern zu können (Zooming). Es soll weiterhin ein Instrument bereitgestellt werden, mit dem sich der Graph mit einer definierten Anzahl von Ebenen darstellen läßt (SpiderView-Technik).

4.3.2. Navigation und Zooming

Die Navigation soll über einen sogenannten Navigationsrahmen erfolgen, der immer den aktuellen Ausschnitt des Graphen beinhaltet, welcher momentan im Hauptanzeigefenster dargestellt wird. Diesen Navigationsrahmen kann der Benutzer per Mausklick frei im MapWindow über den Graph bewegen, der neue Inhalt des Rahmens wird dann im Hauptfenster neu gezeichnet. Der Rahmen kann auch mit der Maus in neuer Größe aufgezogen werden, sein Inhalt wird entsprechend seiner Größe gezoomt dargestellt.

4.3.3. Koordinatentransformation

Um den Graphen auf einer Fläche definierter Größe in seinem vollständigen Umfang zeichnen zu können, müssen alle Knoten und Kanten in ihrer Größe und Position entsprechend dem Umfang des MapWindows transformiert werden. Es muß die relative Größe des Anzeigefensters zum gesamten Originalbaum bestimmt werden, da diese der Größe des anzuzeigenden Navigationsrahmen entspricht. Die Transformationsfunktion muß ebenfalls den aktuellen Zoomfaktor berücksichtigen, mit dem der Graph momentan dargestellt wird.

In Abbildung 10 sind links sind das Anzeigefenster (Viewer) und der Graph als Rechtecke dargestellt, die die jeweiligen Ausmaße dieser Objekte repräsentieren. Man sieht den Fall, daß der Graph viel größer als der Viewer ist, im Viewer ist also nur ein Teilausschnitt sichtbar.

Der gesamte Baum soll im MapWindow dargestellt werden, also muß er (besser die Koordinaten aller seiner Komponenten) in seinem vollständigen Umfang auf die aktuelle Größe des MapWindows transformiert werden. Der Ausschnitt des Graphen, der im Viewer sichtbar ist, wird als Navigationsrahmen transformiert. Er bestimmt sich als Schnittfläche (Intersection) der beiden Rechtecke, die den Viewer und den Graph repräsentieren.

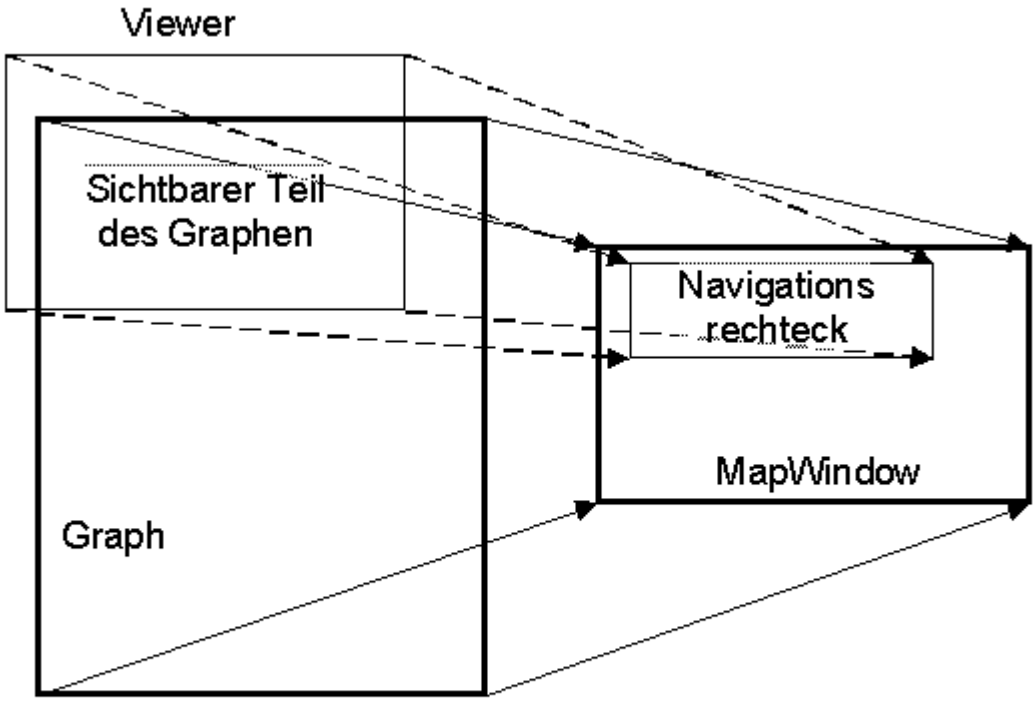

Abbildung 10: Transformation des Graphen

Um Punkte vom Original auf das MapWindow zu transformieren, verwendet man folgende Formeln:

$$X\text{-Koordinate}_{MapWindow} = X\text{-Koordinate}_{Original} * TransformationsFaktorX$$

$$Y\text{-Koordinate}_{MapWindow} = Y\text{-Koordinate}_{Original} * TransformationsFaktorY$$

$$TransformationsFaktorX = Breite_{MapWindow} / Breite_{Original}$$

$$TransformationsFaktorY = Höhe_{MapWindow} / Höhe_{Original}$$

Für die Rücktransformation verwendet man folgende Formeln:

$$X\text{-Koordinate}_{Original} = X\text{-Koordinate}_{MapWindow} * 1 / TransformationsFaktorX$$

$$Y\text{-Koordinate}_{Original} = Y\text{-Koordinate}_{MapWindow} * 1 / TransformationsFaktorY$$

Mit Hilfe dieser Formeln erhält man ein Abbild des gesamten Originalgraphen, welches exakt in das MapWindow paßt.

4.3.4. Berechnung des Navigationsrahmens

Innerhalb des Navigationsrahmens im MapWindow soll sich der Ausschnitt des Graphen befinden, der im Anzeigefenster zu sehen ist. Es soll also immer dem sichtbaren Teil des Graphen entsprechen. Im Extremfall paßt der Graph genau in das Anzeigefenster, der Navigationsrahmen ist dann genauso groß wie das MapWindow.

Die Größe des Rahmens entspricht also immer der gemeinsamen Fläche (Durchschnitt bzw. Intersection) der beiden Rechtecke, die einerseits dem gesamten Graphen und zum anderen der Größe des Anzeigefensters entsprechen. Der Rahmen wird dann ebenfalls noch transformiert und kann danach im MapWindow mit seinen neuen Koordinaten gezeichnet werden.

Wird der Rahmen im MapWindow bewegt, so müssen seine neuen Koordinaten rücktransformiert werden, damit so auch die neue Position des Graphen im Viewer berechnet werden kann.

4.3.5. Zooming mit dem Navigationsrahmen

Wird der Navigationsrahmen vom Benutzer in neuer Größe aufgezogen, so muß dessen relative Größe zum MapWindow berechnet werden, um daraus auch die neuen Faktoren zur Größenänderung (Zooming) im Anzeigefenster zu berechnen. Dabei muß auch berücksichtigt werden, wie groß der Anteil der Größe des im Anzeigefenster (Viewer) gezeichneten Graphen zu seiner Gesamtgröße ist. Dieses Anzeigeverhältnis (DisplayRatio) geht ebenfalls in die Berechnung der neuen Zoomfaktoren ein.

Die Berechnung der neuen Zoomfaktoren erfolgt nach folgenden Formeln:

Erst die Größenverhältnisse zum MapWindow:

$$ZoomFaktorX = Breite_{MapWindow} / Breite_{Navigationsrahmen} [\ \%\ ]$$

$$ZoomFaktorY = Höhe_{MapWindow} / Höhe_{Navigationsrahmen} [\ \%\ ]$$

Danach noch das Anzeigeverhältnis berücksichtigen:

$$ZoomFaktorX\ *= DisplayRatioX [\ \%\ ]$$

$$ZoomFaktorY\ *= DisplayRatioY [\ \%\ ]$$

4.3.6. Interaktion mit dem Anzeigefenster

Die beschrieben Aspekte der Koordinatentransformation, der Navigation und des Zooming machen deutlich, daß die neu zu implementierende Klasse engen Kontakt mit dem Anzeigefenster halten muß. Durch diese Schnittstelle müssen regelmäßig Informationen, wie Zoomfaktoren und Koordinaten aber auch Befehle durch Methodenaufrufe ausgetauscht werden. Es muß also frühzeitig darauf geachtet werden, eine geeignete Instanz zur Koordination dieses Informationsaustausches zu bestimmen.

Das Anzeigefenster selbst muß in seiner Funktionalität erweitert werden, da der zu verwendende Prototyp weder auf diese Form der Interaktion, noch auf weitergehende Anzeigeverfahren, wie beispielsweise das Zooming ausgelegt ist.

Es müssen hier für das MapWindow wesentliche Funktionen zur Verfügung gestellt werden, die z.B. eine Berechnung des Umfangs des Graphen, oder der relativen Größe des Anzeigefensters zum Graphen erlauben.

4.4. Systemintegration

Da ein bereits bestehendes System um zusätzliche Komponenten bzw. Funktionalitäten erweitert wird, ist folglich kein unabhängiger Systementwurf möglich, sondern man muß sich Gedanken über die Integration der entworfenen Komponenten machen.

Der Prototyp muß einerseits um eine neue Klasse für das MapWindow erweitert werden, zum anderen muß die Klasse, die das Anzeigefenster für den Graphen realisiert, um zusätzliche Funktionalität ergänzt werden.

Abbildung 11 zeigt den Aufbau des Prototyps mit den Komponenten, um die er erweitert bzw. abgeändert wird. Die dick umrandeten Kästen markieren die Stellen, an denen das Konzept verändert wird.

Es wurde eine neue Klasse **GraphInfo** implementiert, die die Funktionalität für das MapWindow bereitstellt. Die Klasse **VisiHandler**, die für das Anzeigefenster verantwortlich ist, wurde um die beschriebenen Methoden zur Kommunikation bzw. Datenaustausch mit dem MapWindow erweitert.
Schließlich wurde die Klasse **GraphViewer**, die die beiden genannten Klassen, sowie den Metanavigator enthält, derart ergänzt, daß sie die Schnittstelle zwischen GraphInfo und VisiHandler darstellt.

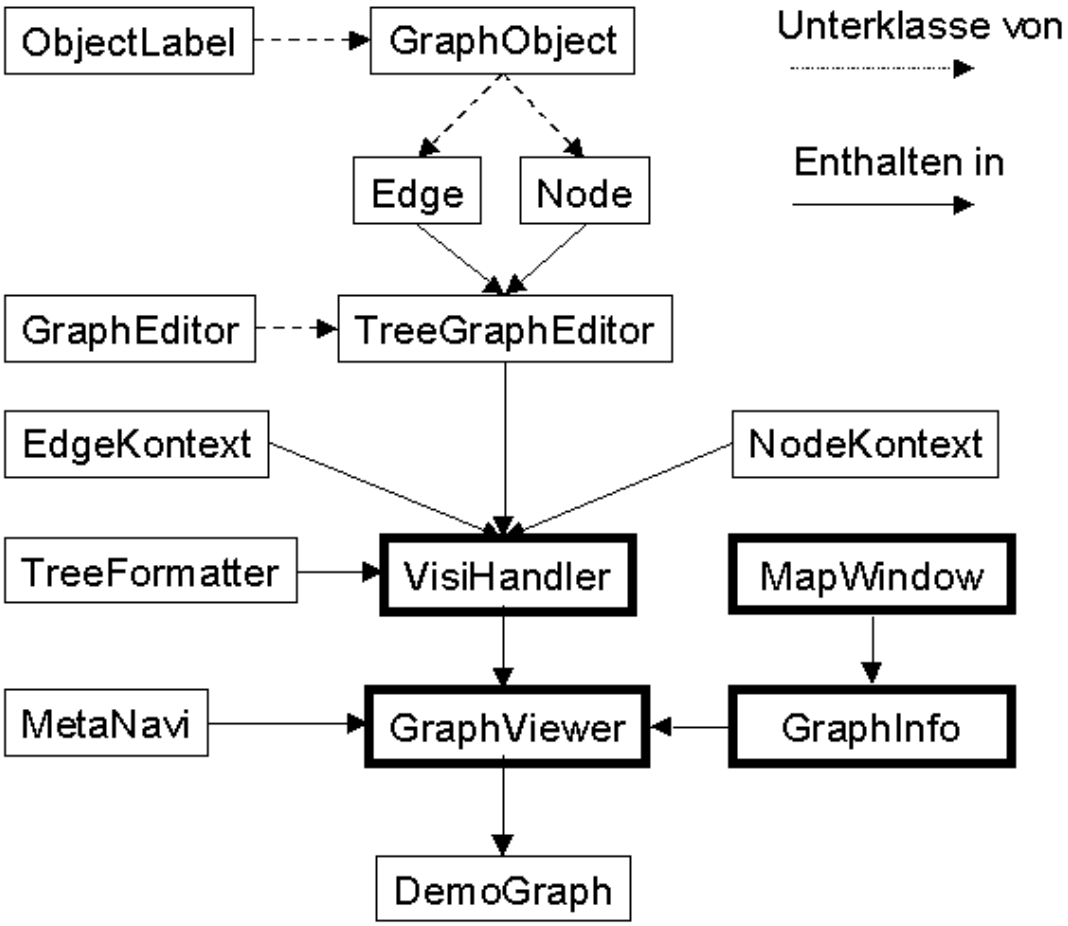

Abbildung 11: Integration in den Prototyp

5. Implementierung

Dieses Kapitel beschreibt die Implementierung des vorgestellten Systementwurfs, die neuen Klassen **MapWindow** und **GraphInfo**, sowie die Erweiterungen in **VisiHandler** und **GraphViewer**. Es werden nicht alle Daten bzw. Methoden vorgestellt, sondern nur diejenigen, die für das grundlegende Verständnis notwendig sind.

Die Implementierung erfolgte vollständig in Java 1.1, als Entwicklungsumgebung wurde Visual Café 1.5 von Symantec verwendet. Getestet wurde das Applet sowohl mit dem einfachen Appletviewer, wie in konfortablen Internet-Browsern, wie z.B. Microsoft Explorer 4.0 oder Netscapes Navigator.

5.1. MapWindow

Die Klasse MapWindow enthält alle notwendigen Daten und Methoden, um das Konzept eines MapWindows zu realisieren. Sie werden nachfolgend im einzelnen beschrieben.

5.1.1. Daten

Das MapWindow muß Zugriff auf den zu visualisierenden Graphen haben, dazu wird ihm im Konstruktor eine Referenz auf die Instanz der Klasse VisiHandler übergeben. Der Graph, sowie die zugehörigen NodeKontexte werden als Datenstrukturen deklariert, da sie zur Darstellung im MapWindow als Kopien des Originals vorliegen müssen. Weiterhin wird noch ein Image deklariert, um darauf den Graphen im Hintergrund zur Vermeidung von Flackereffekten zeichnen zu können. Schließlich existieren noch Objekte, die die verschiedenen Bestandteile des MapWindows repräsentieren, wie z.B. das Navigations- bzw. Zoomingrechteck.

5.1.2. Methoden

Der Konstruktor **MapWindow(GraphViewer gv)** erzeugt eine Kopie des Graphen und seines Kontextes, sowie die Objekte für den Navigationsrahmen und den Zoom-Rahmen. Er erhält als Parameter eine Referenz auf den GraphViewer und damit auch auf den VisiHandler, der ja im GraphViewer enthalten ist.

Die Methode **reshapeMap()** bestimmt die Größe des MapWindows und stellt fest, ob der ursprüngliche Graph in den Viewer paßt, also vollständig angezeigt wird. Sie wird immer vor der Methode paint() aufgerufen, da sie die Parameter für die Koordinatentransformation setzt.

Die Methode **paint()** zeichnet zuerst einen Rahmen um das MapWindow. Danach werden alle Knoten ausgelesen und ihre Koordinaten transformiert. Nach dem Zeichnen aller Knoten und Kanten wird noch über die Methode paintNavRect() der Navigationsrahmen bestimmt und gezeichnet. Alle diese Objekt werden in einem Image dargestellt, welches zum Schluß im MapWindow sichtbar gemacht wird. Trefferknoten werden noch in einer anderen Farbe gezeichnet, um sie von Strukturknoten besser unterscheiden zu können.

PaintNavRect() errechnet den Umfang des Viewer unter Berücksichtigung des aktuellen Zoomfaktors, transformiert diese Daten und zeichnet den neuen Navigationsrahmen.

Zur Koordinatentransformation vom Viewer ins MapWindow existieren mehrere Funktionen, die alle unterschiedliche graphische Objekte transformieren. Das einfachste Objekt ist ein Punkt, er wird durch **Point viewer2map(Point p1)** transformiert. Der Originalpunkt ist der Inputparameter p1, der entsprechende Bildpunkt wird zurückgegeben.

Für Rechtecke ist **Rectangle viewer2map(Rectangle r1)** zuständig. Von dem übergebenen Rechteck r1 werden die zwei Punkte bestimmt, die seine linke obere und rechte untere Ecke bilden. Diese Punkte werden transformiert, ein neues Rechteck wird aus ihnen errechnet und zurückgegeben.
In gleicher Manier werden auch die Objekt Dimension und NodeKontext transformiert.

Die Rücktransformation funktioniert genauso, es existiert die Methode **Point map2viewer(Point p1)**, die Punkte aus dem Bildraum (MapWindow) zurück in den Ursprungsraum (Viewer) transformiert.

Die Methode **void setZoomFactors()** wird nach dem Aufziehen des Zoomrahmens aufgerufen. Sie berechnet die neuen Zoomfaktoren wie im Abschnitt ”4.3.5. Zooming mit dem Navigationsrahmen” beschrieben und übergibt diese an den VisiHandler, der nun seinerseits den Ursprungsgraphen neu zeichnet.

5.2. GraphInfo

Die Klasse GraphInfo dient lediglich als verwaltende Instanz zur Anzeige des MapWindows und eventueller zusätzlicher Navigationsinstrumente, wie SpiderViews etc. Dies hat den Vorteil, daß bei zukünftigen Erweiterungen der gesamten Applikation die ursprünglichen Klassen nicht verändert werden müssen. Es würde eine Erweiterung der Klasse GraphInfo für jede neue Komponente genügen.

5.2.1. Daten

Die Klasse GraphInfo hält Referenzen zu allen Komponenten, die an der Visualisierung beteiligt sind, in diesem Falle also zum VisiHandler und zum MapWindow. Weiterhin wird noch eine Scrollbar deklariert, die das SpiderView-Konzept demonstriert. Mit ihrer Hilfe kann über einen Zahlenbereich gescrollt werden, der der Anzahl der Ebenen des Graphen entspricht.

Es sind noch Hilfsvariablen vorhanden, die zusätzliche Verwaltungsinformationen zum Graphen enthalten können, so z.B. die Anzahl der Knoten / Ebenen oder die Ebene mit den meisten Knoten.

5.2.2. Methoden

Der Konstruktor initialisiert die Hilfsvariablen und erzeugt das MapWindow und die SpiderView.

Die Methode **getInfo()** traversiert den Baum und bestimmt die Anzahl der Knoten und Ebenen, sowie die Breite (Ebene mit maximaler Anzahl Knoten) des Baumes.

HideAtLevel() blendet ab einem bestimmten Level alle tiefer liegenden Knoten im Baum aus.

5.3. VisiHandler

Die Klasse VisiHandler muß um die zusätzlich notwendig gewordene Funktionalität erweitert werden, um ein reibungsloses Zusammenspiel mit dem MapWindow zu gewährleisten.

5.3.1. Berechnung der Dimensionen der Objekte

Der VisiHandler muß aufwendiger auf eine Größenänderung des Viewers reagieren. Bis jetzt wurde diese Änderung nur in der Methode, die für das Zeichnen der Objekte verantwortlich ist (**paint()**), beachtet.

Für die Visualisierung des Graphen im MapWindow ist der reale Umfang des (gezoomten) Graphen, sowie der davon im Viewer angezeigte Teil wichtig, um unter anderem den Navigationsrahmen richtig setzen zu können. Wird vom Benutzer im MapWindow der Positionsrahmen neu gesetzt (**setOffset()**), muß der VisiHandler diesen neuen Offset zur Darstellung des Graphen berücksichtigen. In der Methode **reshapeViews()** werden die Dimensionen der relevanten Objekt berechnet, sie wird bei jeder Änderung eines für sie relevanten Parameters aufgerufen.

5.3.2. Zooming

Die Methode **paint()** muß zusätzlich zum Offset auch noch die Zoomfaktoren in X- und Y-Richtung berücksichtigen, die ihr vom MapWindow übergeben werden (**setZoomFactors()**). Das Verhältnis vom im Viewer angezeigten Teil des Graphen zum Gesamtumfang des Graphen (dispRatioXY), ist ebenfalls vom aktuell gewählten Zoomfaktor abhängig.

5.3.3. Interaktion zwischen Viewer und MapWindow

Um dem Benutzer immer ein konsistentes Verhältnis zwischen Inhalt des Viewers und Inhalt des Navigationsrahmens im MapWindow bieten zu können, ist es notwendig, daß der VisiHandler alle Verschiebungen des Graphen per Scrollbars im Viewer auch dem MapWindow übergibt, so daß dort der Navigationsrahmen angepaßt werden kann. Dazu wurde der Eventhandler des VisiHandlers angepaßt, er reagiert auf eine Betätigung der Scrollbars, markiert diese Events aber nicht mehr als verarbeitet, sondern gibt sie wieder in die Eventqueue zurück, von der aus sie auch andere Komponenten der Anwendung erreichen können. Der GraphViewer verwaltet diese Events, indem er die entsprechenden Funktionen des MapWindows aufruft. Weiterhin werden auch neue Events erzeugt, falls z.B. Knoten bewegt oder Teilbäume ausgeblendet werden.

In die andere Richtung, also vom MapWindow aus, funktioniert diese Kommunikation über einen direkten Methodenaufruf von z.B. **setOffset()** im VisiHandler. Man könnte hier auch neue Events zur Kommunikation erzeugen, aber da das MapWindow bereits eine direkte Referenz auf den VisiHandler besitzt, sind direkte Methodenaufrufe eleganter zu implementieren.

5.4. GraphViewer

Der GraphViewer wurde um eine Deklaration eines Objekts der Klasse GraphInfo ergänzt, welches in einem zusätzlichen Panel dargestellt wird. Sein Eventhandler wurde erweitert, um die neuen Events vom VisiHandler abzufangen und an das MapWindow übergeben zu können. Weitere Änderungen waren nicht notwendig.

5.5. Das fertiggestellte Applet

Eine aktuelle Version des fertigen Applets liegt im WorldWideWeb[6] und kann von dort frei heruntergeladen werden. Sie ist allerdings auf die Darstellung eines einzigen Beispielgraphen beschränkt. Nach der Integration des Applets in das Konstanzer Hypertextsystem (KHS), kann es im realen Betrieb evaluiert werden. Das KHS besteht aus ca. 700 verschiedenen Beschreibungen von Online-Datenbanken, die polyhierachisch organisiert sind und einen hohen Verknüpfungsgrad zwischen den Datenbanken aufweisen, die zu ähnlichen Inhalten Informationen enthalten [Rittberger94].

Die Hypertexte, die zum Testen Verwendung fanden, wurden als Suchergebnisse im KHS generiert, als Dateien extrahiert und vom Applet wieder eingelesen. Es wurde mit vielen Graphen unterschiedlichster Größe getestet und läßt auch auf schwächeren Maschinen akzeptable Antwortzeiten erwarten.

In den Abbildungen 12 bis 15 wird die Funktionsweise des MapWindows graphisch deutlich gemacht und der damit verbundenen Nutzen für den Anwender demonstriert.

[6] http://www.inf-wiss.uni-konstanz.de/~ettl

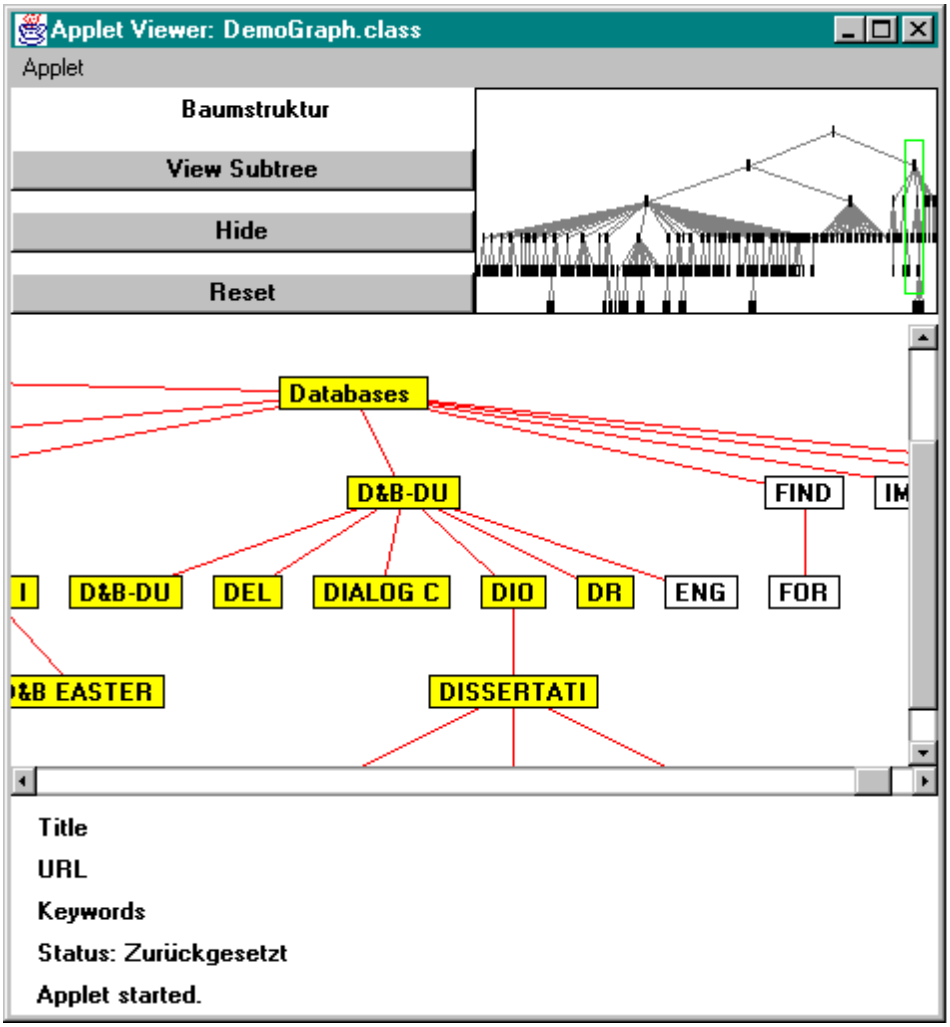

Abbildung 12: MapWindow mit großem Graphen

Der in Abbildung 12 visualisierte Graph besteht aus knapp 200 Knoten. Rechts oben im MapWindow wird dem Benutzer einen Gesamtansicht des Graphen nebst Navigationsrahmen (rechts außen im MapWindow erkennbar) angezeigt.

Man kann weiterhin erkennen, daß bei dieser Anzahl an Knoten die Darstellung im MapWindow undeutlich zu werden beginnt, da die Knoten zu klein werden und fast nicht mehr unterschieden werden können. Zwar läßt sich das Fenster, in dem das Applet dargestellt wird bzw. der Platz auf der Webseite vergrößern, aber es wird deutlich, daß eine Obergrenze für die Anzahl gleichzeitig im MapWindow darstellbarer Knoten existiert.

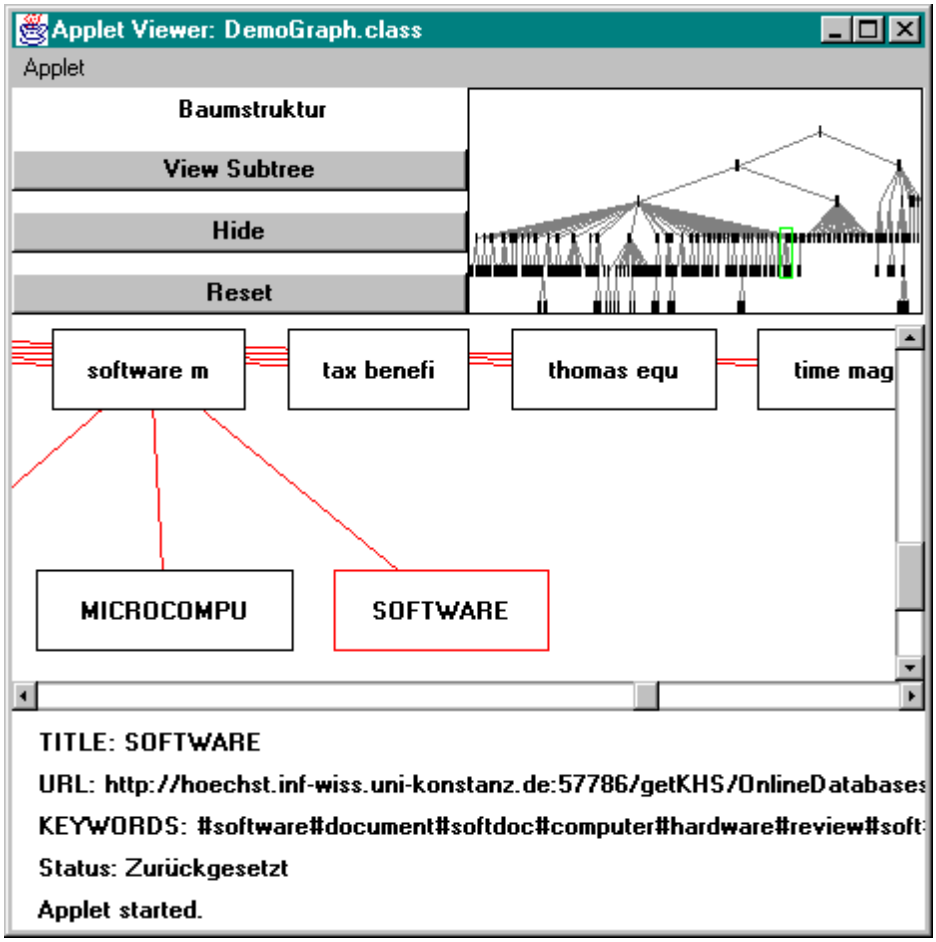

Abbildung 13: Großer Graph mit Zooming

In Abbildung 13 wird der gleiche Graph visualisiert, allerdings wurde gezoomt. Man erkennt, daß der Umfang der Knoten im Viewer deutlich größer ist, als in der vorhergehenden Abbildung. Diesen Größenunterschied könnte man durchaus zur Darstellung weiterer Informationen des Knotens nutzen.
Der Inhalt des neu aufgezogenen Navigationsrahmens (im MapWindow unten rechts) entspricht exakt dem Inhalt des Viewers.
Per Mausklick kann nun mit dem neuen Rahmen wieder über den gesamten Graphen navigiert werden.

Es wird auch noch eine Schwäche beim Zoomen im VisiHandler des Prototyps deutlich, nämlich daß die Knotenbeschriftungen nicht an die Größe der Knoten angepaßt wird, sondern immer die gleiche Schriftgröße verwendet wurde. Bei eng nebeneinander liegenden Knoten überlappen sich die Schriftzüge und werden unleserlich.
Hier müßte der Visihandler noch weiter verbessert werden, um entweder automatisch die Schriftgröße anzupassen, oder bei stark vergrößerten Ansichten die Beschriftung vollständig wegzulassen.

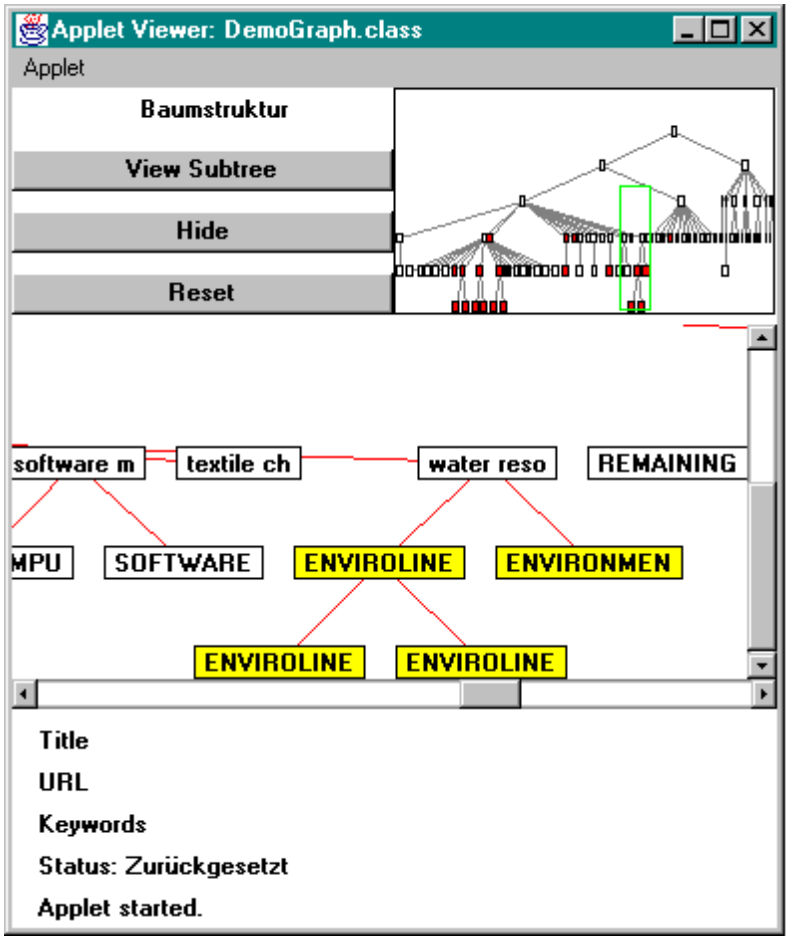

Abbildung 14: Mittelgroßer Graph mit Treffermarkierung

In Abbildung 14 ist ein mittelgroßer (ca. 60-70 Knoten) Graph visualisiert. Bei Graphen dieser Größe können die Knoten im MapWindow schon so groß gezeichnet werden, daß die Treffer durch ihre Färbung von den Strukturknoten unterschieden werden können. Im Viewer ist diese Unterscheidung immer möglich, außer wenn extrem gezoomt wurde.

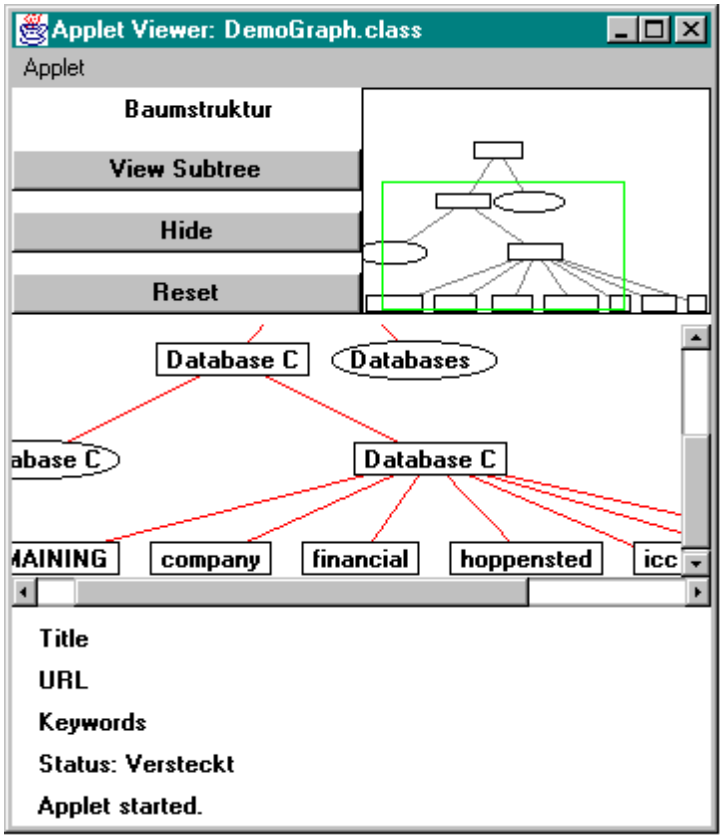

Abbildung 15: Metanavigation und MapWindow

Abbildung 15 zeigt wieder den mittelgroßen Graphen, allerdings wurden hier über den Metanavigator (im Fenster rechts oben) ganze Teilbäume ausgeblendet. Im MapWindow werden diese Änderungen an der Struktur des zu visualisierenden Graphen ebenfalls vorgenommen und angezeigt. Von den ursprüngliche etwa 60-70 Knoten, aus denen der Graph besteht, sind hier nur noch 12 sichtbar. Die beiden oval gezeichneten Knoten repräsentieren dabei die ausgeblendeten Teilbäume.

Es wäre auch denkbar gewesen, im MapWindow immer die Gesamtansicht des vollständigen Graphen anzuzeigen, aber es wurde Wert auf eine konsistente Darstellung der Graphen gelegt, um den Benutzer nicht zu verwirren. So sollte der Graph, der den Inhalt des MapWindows bildet, immer mit demjenigen übereinstimmen, der im Viewer angezeigt wird. Es besteht aber auch die Möglichkeit, den Benutzer selbst entscheiden zu lassen, welche dieser alternativen Darstellungsweisen er bevorzugt.

6. Beurteilung und Ausblick

Das aus dieser Arbeit hervorgegangene Navigationsinstrument bietet dem Anwender eine wesentliche Unterstützung beim Überblicken von großen Hypertexten. Er kann sich mehr auf die eigentlichen Inhalte dieser Strukturen konzentrieren und muß weniger Aufwand für Orientierung und Navigation treiben.

Die Entscheidung für das Konzept eines MapWindows hat sich in zweierlei Hinsicht als richtig erwiesen. Zum einen ist der Aufwand für Entwurf, Implementierung und Test einer solchen Navigationskomponente im Hinblick auf den Nutzen, den sie bietet, vergleichsweise gering. Die Verwendung der komplexeren Fish-Eye-Views wäre sicher zeitaufwendiger geworden als die etwa vier Wochen, die die Realisierung des MapWindows in Anspruch genommen hat.

Zum anderen ist dieses Konzept einfach erlern- und bedienbar, was denjenigen Benutzern zu Gute kommt, die Informationssysteme eher unregelmäßig nutzen und nicht durch täglichen Umgang damit zu Experten geworden sind. Besonders diese Anwender sind auf Navigations- und Orientierungshilfen angewiesen, die professionellen Rechercheure bedürfen ihrer Hilfe hingegen in wesentlich geringerem Ausmaße.

Insofern hat sich auch die aufwendige Erarbeitung des aktuellen Standes der Technik über derartige Konzepte und Systeme gelohnt, zumal der Wert intelligenter Navigationssysteme in unserer multimedialen Zukunft sicher noch weiter anwachsen wird. Beim explosionsartigen Anstieg von aufkommenden Daten-, Informations- und Wissensmengen, der nicht allein im Internet, sondern in nahezu allen Bereichen des privaten und öffentlichen Lebens zu beobachten ist, werden Metainformationssysteme bald zu unentbehrlichen Wegbegleitern für alle Informationssuchenden werden.

Alleine durch exploratives Browsing werden in naher Zukunft kaum noch sinnvolle Recherchen mehr möglich sein, da die Anzahl der relevanten Informationseinheiten, die als Ergebnismenge anfallen, in keiner Relation mehr zum Zeitaufwand stehen, der zu ihrer Beschaffung notwendig ist.

Es bleibt zu hoffen, daß der Mensch seine natürlichen Grenzen bei der Bewältigung von Komplexität und der Verarbeitung von Informationen erkennt und beachtet, da sie auch nicht durch den Einsatz noch so ausgefeilter Hilfsmittel aufgeweicht werden können.

Abbildungsverzeichnis

Literatur

[Agosti96] Agosti M., A. Smeaton: Information Retrieval and Hypertext Kluwer Academic Publishers 96

[Aßfalg92] Aßfalg R.: Eine Fish-Eye-Navigationshilfe für das Konstanzer Hypertextsystem (KHS), Bericht 6/92, Blaue Reihe, Informationswissenschaft, Universität Konstanz, 1992

[Balzert96] Balzert H.: Lehrbuch der Softwaretechnik : Softwareentwicklung, Spektrum, Akad. Verlag, 1996

[Beard&Walker90] Beard D., Walker J. Q.: Navigational techniques to improve the display of large two-dimensional spaces.
aus: Behaviour & Information Technology, 1990, Vol. 9, No. 6, S.451-466

[Bekavac&Rittberger97] Bekavac B., Rittberger M.: Kontextsensitive Visualisierung von Suchergebnissen
aus: Fuhr N.: Hypertext – Information Retrieval – Multimedia 97, Proceedings HIM´97 (S. 307-321), Universitätsverlag Konstanz 1997

[Bekavac96] Bekavac B., Suchverfahren und Suchdienste des WWW
aus: NfD – Nachrichten für Dokumentation, 1996, Bd. 47, Nr. 4, S. 175-189

[Elzer97] Elzer P., Krohn U.: Visualisierung zur Unterstützung der Suche in komplexen Datenbeständen
aus: Fuhr N.: Hypertext – Information Retrieval – Multimedia 97, Proceedings HIM´97 (S. 27-38), Universitätsverlag Konstanz 1997

[Ettl&Jäger97] Algorithmen zur Visualisierung von WWW-Hypertexten, Projektkurs VIR 96/97, Informationswissenschaft Konstanz 1997

[Flanagan96] Flanagan D.: Java in a Nutshell - A Desktop Quick Reference for Java Programmers, O'Reilly & Associates Inc., 1996

[Formella&Keller95] Formella A., Keller J.: Generalized Fisheye Views of Graphs
aus: Brandenburg: Graph Drawing Symposium on Graph Drawing (Passau Germany) Proceedings Springer 1995

[Fröhlich&Werner94] Fröhlich M., Werner M.: Demonstration of the Interactive Graph Visualization System da Vinci
aus: Tamassia: Graph Drawing Proceedings, Springer 1994

[Hammwöhner97] Hammwöhner R.: Das Konstanzer Hypertextsystem (KHS) im wissenschaftlichen und technischen Kontext, Universitätsbibliothek Konstanz, Schriften zur Informations-wissenschaft, Band 32, 1997

[Kaugars et al. 94] Kaugras K., Reinfels J., Brazma A.: A Simple Algorithm for Drawing Large Graphs on Small Screens
aus: Tamassia: Graph Drawing Proceedings, Springer 1994

[Kimelman et al. 94] Kimelman D.: Reduction of Visual Complexity in Dynamic Graphs
aus: Tamassia: Graph Drawing Proceedings, Springer 1994

[Kuhlen et al. 96] Kuhlen R., Rittberger M., Bekavac B.: ”Complex Operations in a Distributed, Network-Like Information Space. Resource Selection, Searching, Structuring and Visualisation in an Electronic Market Place” Proceedings of the SIGIR´96 Workshop on Networked Information Retrieval, 1996, http://SunSITE.Informatik.RWTH-Aachen.de/Publications/CEUR-WS/Vol-7/bekavac.pc

[Kuhlen91] Kuhlen R.: Hypertext: ein nichtlineares Medium zwischen Buch und Wissensbank, Springer 1991

[Madden et al. 95] Madden B., Madden P., Powers S. Himsolt M.: Portable Graph Layout and Editing
aus: Brandenburg: Graph Drawing Symposium on Graph Drawing (Passau Germany) Proceedings Springer 1995

[Nielsen95] Nielsen J.: Mutimedia and Hypertext – The Internet and Beyond Academic Press Inc. 1995

[Orosco97] Orosco R.: AutoFocus: User Assistance in Information Visualization
aus: Fuhr N.: Hypertext – Information Retrieval – Multimedia 97, Proceedings HIM´97 (S. 34-52), Universitätsverlag Konstanz, 1997

[Rittberger94] Rittberger M., Selektion von Online-Datenbanken in einem offenen Hypertextsystem
aus: Rauch, Strohmeier, Hiller, Schlögl: Mehrwert von Informationen – Professionalisierung der Informationsarbeit Proceedings ISI’94, Universitätsverlag Konstanz, 1994

[Sablowsky&Frick96] Sablowsky R., Frick A.:Automatic Graph Clustering (System Demonstration)
aus: North: Graph Drawing Symposium on Graph Drawing (California USA), Springer 1996

[Saxer90] Saxer K.-H., Gloor P.: Navigation im Hyperraum – Fisheye Views in HyperCard
aus: Gloor P., N.A. Streitz (Hrsg.) Hypertext and Hypermedia Informatik Fachberichte 249, Springer 1990

[Scheuch97] Scheuch K., Entwicklung von Methoden für die automatische Erkennung von Firmenstrukturen in WWW-Hypertexten, Universität Konstanz, Diplomarbeit 1997

[Storey&Müller95] Storey M., Müller H.: Graph Layout Adjustment Strategies
aus: Brandenburg: Graph Drawing Symposium on Graph Drawing (Passau Germany) Proceedings Springer 1995

[Utting&Yankelovich 89] Utting K., Yankelowitch N.: Context and Orientation in Hypermedia Networks
aus: ACM Transaction on Information Systems, Vol. 7, No.1 (pp. 58-84) Jan. 1989

[Wang&Miyamoto95] Wang X., Miyamoto I.: Generating Customized Layouts
aus: Brandenburg: Graph Drawing Symposium on Graph Drawing (Passau Germany) Proceedings Springer 1995