INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

Focus-and-Context Visualisierung der Informationsstruktur von Webressourcen

Martin Schmitz

Bachelorarbeit

Beginn der Arbeit:	01. Juni 2007
Abgabe der Arbeit:	03. September 2007
Gutachter:	Prof. Dr. Stefan Conrad
	Prof. Dr. Michael Schöttner

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 03. September 2007

_____

Martin Schmitz

Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Tool zur Visualisierung einer der wichtigsten Erfindungen der Neuzeit entwickelt und implementiert:
wwwVis, eine Anwendung zur Visualisierung des *World Wide Webs (WWW)*.

Das WWW besitzt eine unzählbare, potenziell unendliche Größe. Es wird jeden Tag von Milliarden von Menschen genutzt und erweitert. Dadurch bilden sich eigene Strukturen, die von den klassischen Informations- und Visualisierungssystemen nicht dargestellt werden können. Ziel ist es, ein Programm zu entwickeln, dass die Visualisierung der Webstruktur ermöglicht.

Diese Arbeit beschäftigt sich zunächst mit der Entstehung und den Grundlagen des WWW. Anschließend werden bereits bekannte Studien über die Struktur des WWW vorgestellt.

Im weiteren Verlauf spielen Eigenschaften, Anforderungen, Techniken und Methoden der Daten- bzw. Informationsvisualisierung eine große Rolle. Die wichtigste verwendete Art der Visualisierung ist die *Focus-and-Context-Visualisierung*. Sie liefert zum einen eine Hervorhebung interessanter Details des Fokuses und lässt zum anderen den Gesamtkontext nicht ausser acht. Darüber hinaus sind das *Information Visualization Data-State-Reference Model*, sowie das Mantra der Informationsvisualisierung wichtige Orientierungspunkte auf dem Weg zur Fertigstellung von *wwwVis*.

Bevor auf die eigentliche Entwicklung und Implementierung von *wwwVis* eingegangen wird, werden Hilfsmittel und Basisdaten der Anwendung erläutert.
Danach wird die graphische Benutzeroberfläche von *wwwVis* vorgestellt, die Funktionalitäten der Anwendung, zum Teil auch anhand von Beispielen, näher gebracht, sowie einige Implementierungen exemplarisch für die Gesamtentwicklung von *wwwVis* dargelegt.

Zu guter letzt finden sich noch ein paar Vorschläge für zukünftige Weiterentwicklungen in dieser Arbeit wieder.

Inhaltsverzeichnis

Inhaltsverzeichnis **1**

1 Einleitung **3**

1.1 Motivation und Ziele . . . 3

1.2 Struktur der Arbeit . . . 3

2 Theoretischer Hintergrund **5**

2.1 Das Internet und das World Wide Web (WWW) . . . 5

2.1.1 Abgrenzung von Internet und WWW . . . 5

2.1.2 Definition und Inhalt von Webressourcen . . . 7

2.1.3 Der Webgraph als Strukturbeschreibung des WWW . . . 8

2.1.4 Die Struktur des WWW . . . 9

2.1.5 The Small World Phänomen . . . 11

2.2 Daten- und Informationsvisualisierung . . . 12

2.2.1 Daten, Informationen und Wissen . . . 12

2.2.2 Data Mining . . . 13

2.2.3 Was versteht man unter Daten- und Informationsvisualisierung? . . . 13

2.2.4 Visualisierung: Eigenschaften und Anforderungen . . . 15

2.2.5 Möglichkeiten und Techniken der Visualisierung . . . 17

2.2.6 Focus-and-Context Visualisierung . . . 19

3 Hilfsmittel und Basisdaten **21**

3.1 Die Programmiersprache Java . . . 21

3.2 JDBC und DB2 . . . 21

3.3 Bachelorarbeit: *Extraction and Storage of Web Structures* . . . 22

3.3.1 Datenbankmodell . . . 22

3.3.2 Basisdaten . . . 22

3.4 Die Qt Entwicklungsumgebung . . . 24

3.4.1 Qt Designer . . . 24

3.4.2 Qt Jambi . . . 25

3.4.3 Visualisierungen in Qt Jambi . . . 25

4 Entwicklung und Implementierung von *wwwVis* **26**

4.1 Design der graphischen Benutzeroberfläche 26
4.2 Funktionalität von *wwwVis* 27
4.2.1 Verbindung zur Datenbank 28
4.2.2 HostNet 28
4.2.3 WebNet 28
4.2.4 SetNet 28
4.2.5 ClusterNet 29
4.2.6 Filter 29
4.2.7 Visualisierung 29
4.3 Aufbau und Struktur der Implementierung 31
4.4 Operationen im Daten- und Darstellungsraum 37
4.4.1 Verwendung der Basisdaten 37
4.4.2 Breitensuche 38
4.4.3 Visualisierung 38

5 Beispielvisualisierungen und -analysen mit *wwwVis* **39**
5.1 HostNet 39
5.2 WebNet 40
5.3 SetNet 41
5.4 ClusterNet 42

6 Abschluss **43**
6.1 Fazit 43
6.2 Ausblick auf zukünftige Arbeiten 43

Literatur **44**

Abbildungsverzeichnis **47**

Tabellenverzeichnis **47**

1 Einleitung

1.1 Motivation und Ziele

Das *World Wide Web (WWW)* ist heutzutage allgegenwärtig. Auf der ganzen Welt nutzen Menschen die Funktionen des WWW, um Daten oder Informationen aufzufangen, zu sammeln und auszutauschen. Täglich kommen nicht nur neue Nutzer des Webs hinzu, auch die Anzahl der Informationen im Web steigt von Sekunde zu Sekunde an. Immer mehr Privatpersonen, kleine Firmen, Hochschulen, große Organisationen oder riesige Unternehmen legen sich eigene Sammlungen von Informationen, so genannte Websites, an und verweisen dabei auf Websites von Freunden, Verwandten, Wissenschaftlern, Geschäftspartnern, Werbepartnern oder einfach auf irgendeine die ihnen gefällt oder die zum Thema der eigenen passt. Selbst auf Informationen der eigenen Website wird gelinkt. Eine Hauptseite verweist meist auf alle anderen Seiten und Dateien der Website. Durch interne und externe Links entstehen so hierarchische Strukturen im Web.
Nun wäre es natürlich interessant zu wissen, wie diese Strukturen genau aussehen, bzw. zu erfahren, wer mit wem in Beziehung steht.

Welche Websites verweisen auf die Universität Düsseldorf?
Und auf welche Websites linkt die Universität Düsseldorf?
Verweist die Universität Düsseldorf sogar auf die Universität Köln?
Welche Publikationen stellt ein Lehrstuhl zur Verfügung?
Welche Websites sind themenverbunden?

Mit Hilfe der üblichen Informationssysteme wie einem Webbrowser oder einer Suchmaschine ist dies nur schwierig und mit Mühe herauszufinden. Kenntnisse über die Strukturen des Webs können aber nicht nur einfache Fragen beantworten, sondern helfen bei der Entwicklung von Anwendungen im Information Retrieval oder bei der Verbesserung von Web-Crawlern, Suchmaschinen oder andere Informationsfiltern.
Darüber hinaus kann die Suche nach einer Datei oder einer Seite im WWW aufgrund der immer stärker werdenden Flut an Informationen sehr mühsam sein. Eine Auflistung aller Dateien einer Website wäre dabei durchaus hilfreich, insbesondere dann, wenn der Dateiname unbekannt ist.
Im Rahmen dieser Bachelorarbeit wurde daher ein Tool namens *wwwVis* entwickelt und implementiert, dass die Struktur des Webs visualisiert und mit dem es möglich ist, die Fragen zu beantworten, sowie die Suche nach Dateien in den Weiten des WWW zu erleichtern. Die Visualisierung geschieht in der Art *Focus-and-Context*, um die große Datenmenge des Webs zu meistern. Aber auch andere Techniken und Möglichkeiten der Visualisierung wurden genutzt, damit der Umgang mit dem Programm so einfach und der Nutzen von der Anwendung so hoch wie möglich ist. Die bei der Visualisierung verwendeten Daten stammen aus der Datenbank, die im Rahmen der Bachelorarbeit *Extraction and Storage of Web Structures* [Aba06] entwickeltet wurde.

1.2 Struktur der Arbeit

Diese Arbeit ist in vier Teilbereiche gegliedert. Zunächst wird der theoretische Hintergrund (2), der zum Verständnis des weiteren Verlaufs wichtig ist und als Grundlage zur Entwicklung des Tools beigesteuert hat, erläutert. Die Begriffe *Internet* und *World Wide*

Web werden differenziert, Webressourcen definiert, sowie die Struktur des Webs erklärt. Darüber hinaus ist das Themengebiet der Visualisierung ein großer Bestandteil dieser Arbeit. Eigenschaften, Anforderungen, Techniken und Möglichkeiten der Visualisierung spielen bei der Entwicklung einer visuellen Anwendung eine bedeutende Rolle.
Der Abschnitt Hilfsmittel und Basisdaten (3) stellt zum einen die bei der Implementierung von *wwwVis* verwendeten Hilfsmittel *Java*, *JDBC*, *DB2* und das Programmpaket *Qt*, sowie zum anderen die Basisdatenbank und deren Struktur vor.
Kapitel 4 beschäftigt sich mit der eigentlichen Entwicklung und Implementierung der Anwendung *wwwVis*. Neben der Präsentation der graphischen Benutzeroberfläche werden hier die Funktionalitäten, der Aufbau und die Struktur der Implementierung aufgezeigt, sowie anschließend stellvertretend drei spezielle Implementierungen erläutert.
Im letzten Abschnitt (5) dieser Arbeit werden schließlich vier Beispielvisualisierungen und -analysen auf der Basisdatenmenge durchgeführt, um den Einstieg in *wwwVis* zu erleichtern.

2 Theoretischer Hintergrund

2.1 Das Internet und das World Wide Web (WWW)

2.1.1 Abgrenzung von Internet und WWW

Die Begriffe *Internet* und *World Wide Web (kurz: WWW)* werden heutzutage häufig irrtümlicherweise synonym benutzt. Im Gegensatz zu diesem weit verbreiteten (Miss-) Verständnis sind Internet und WWW differenziert zu betrachten [MS04].
Das *Internet* bildet die Grundlage für das WWW und ist ein riesiger Zusammenschluss von millionenfach über die ganze Welt verteilten Computer-Netzwerken, die mittels verschiedener Kommunikationsverbindungen, auch bezeichnet als *communication links* (z.B. Glasfaserkabel, Kupferkabel oder Funk), Daten austauschen können [KR05]. Es entstand aus einem Versuchsnetzwerk des amerikanischen Verteidigungsministeriums, dem so genannten *ARPANET*, welches 1969 zunächst nur aus wenigen Rechnern bestand. Im Laufe der Jahre entwickelte sich daraus ein mächtiges Netz aus unzählbaren Rechnern, die potentiell alle miteinander interagieren können, so dass eine weltweite Kommunikation möglich wurde.
Die Kommunikation zwischen Computern erfolgt über *Protokolle* [KR05]. Sie kontrollieren den Versand und den Empfang von Daten zwischen Rechnern im Internet. Die wichtigsten und am häufigsten benutzten Protokolle zum Transport von Daten im Internet sind das *Transmission Control Protocol (TCP)* und das *Internet Protocol (IP)*. Das Internet verbindet neben den von vielen Menschen genutzten Arbeitsplatzrechnern, auch Telefone, personal digital assistants (PDA), Fernseher, Sicherheitssysteme und andere elektronische Geräte. Zusammengefasst werden sie als *Endsysteme* oder *Clients* bezeichnet. Endsysteme, die verschiedene Dienste und Anwendungen im Internet bereitstellen, werden in der Regel *Hosts* oder *Server* genannt. Beispiele für Dienste im Internet sind Emailanwendungen, Newsgroups, verteilte Datenbanken, Online-Spiele, der Transfer von Dateien über das *File Transfer Protocol (FTP)* oder aber eben das *World Wide Web* [CER07] [KR05] [MS04].
Das WWW ist ein Kommunikationsdienst, der die Technik des Internets mit Hilfe des *Hypertext Transfer Protocol (HTTP)* nutzt. Im Jahre 1989 entwarf es der Wissenschaftler Tim Berners-Lee gemeinsam mit Mitarbeitern der *Europäischen Organisation für Kernforschung (CERN)*, dem größten Teilchen-Physik-Labor der Welt mit Sitz in Genf [CER07]. Ursprünglich diente es lediglich zur Vereinfachung der Kommunikation und dem Datenaustausch zwischen Wissenschaftlern. Die Idee von Berners-Lee sah ein mit Verweisen vernetztes Informationsmanagementsystem vor, welches allgemeingültig, übertragbar und einfach sein sollte. Darüber hinaus sollte es die Möglichkeit bieten wichtige Informationen oder auch Verweise und Quellenangaben darzustellen und wiederauffindbar zu machen:

> „We should work toward a universal linked information system, in which generality and portability are more important than fancy graphics techniques and complex extra facilities. The aim would be to allow a place to be found for any information or reference which one felt was important, and a way of finding it afterwards [BL90].“

Auch damals sah er schon großes Potenzial in seiner Idee:

> „The result should be sufficiently attractive to use that it the information contained would grow past a critical threshold, so that the usefulness the scheme would in turn encourage its increased use. The passing of this threshold accelerated by allowing large existing databases to be linked together and with new ones [BL90].“

Um auf Informationen in anderen Quellen zu referenzieren, veranlasste Berners-Lee den Gebrauch von *Hypertext* [BL90]. Dieser ermöglicht neben der einfachen Darstellung von Informationen auch Querverweise auf andere Informationsträger, die auf irgendeinem Endsystem oder Host im Internet gespeichert sind. Infolgedessen ist es nun mittels der Verweise, den *Links*, sehr schnell möglich zwischen den einzelnen Hypertexten zu navigieren, ohne direkt zu wissen, wo diese Information bereitgestellt wurde. Die Hypertexte sollten nun mit *Client-Browser-Programmen* unter Verwendung des *HTTP* von den verschiedenen miteinander vernetzten Servern des Internets abgerufen werden können (Abbildung 1).
Bekanntlich wurde die Idee von Tim Berners-Lee dementsprechend umgesetzt, stetig er-

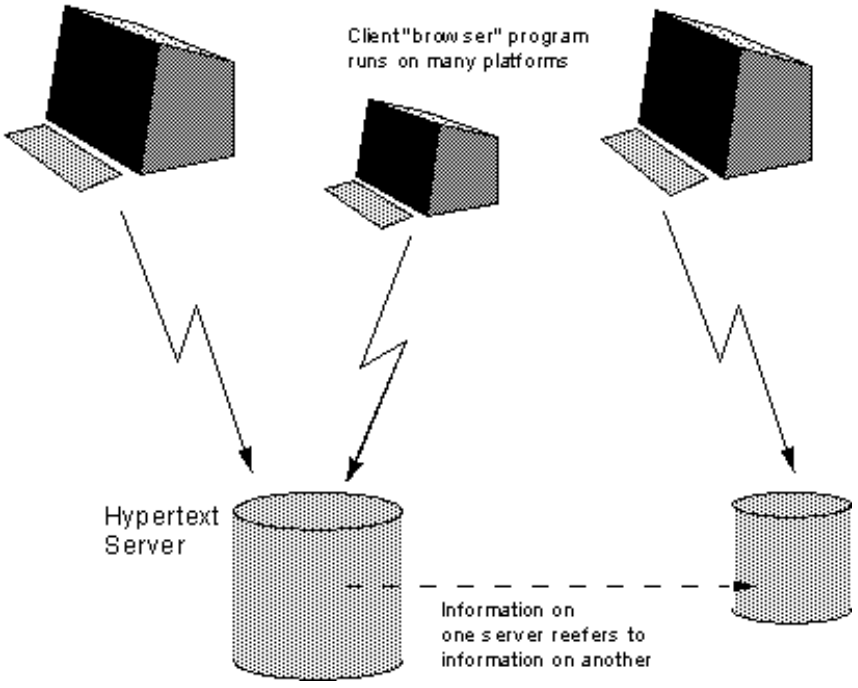

Abbildung 1: Client/Server-Modell für ein verteiltes Hypertextsystem (aus [BL90])

weitert und weiterentwickelt. Der Gebrauch des *World Wide Webs* hat sich in allen Bereichen bewährt. Es wird heute von einem Großteil der Weltbevölkerung in nahezu jedem Land der Erde genutzt. Täglich kommen tausende neue Nutzer hinzu. Neben den Hypertexten gibt es mittlerweile eine Vielzahl anderer Ressourcen, die über das WWW angefragt werden können. Die Größe des weltweiten Netzes wächst von Sekunde zu Sekunde, ist allerdings sehr schwer bestimmbar, da neben dem durch Suchmaschinen messbaren

Oberflächen-Web auch das so genannte *Deep Web* existiert. Das *Deep Web* beinhaltet Webressourcen, die sich in passwortgeschützten Datenbanken oder auf nichtzugreifbaren Servern befinden, und kann somit nicht von bekannten Suchmaschinen-Crawlern erfasst werden [Lew05].
Somit besteht das WWW aus einer unbekannten, potenziell unendlich erweiterbaren Menge von Daten bzw. *Webressourcen*.

2.1.2 Definition und Inhalt von Webressourcen

Neben Hypertexten gibt es inzwischen viele andere Ressourcen, die über das Web angefragt werden können. In erster Linie sind elektronische Dokumente, wie z.B. PDF-Dateien, Bilder, Audio-Dateien oder Video-Dateien zu nennen. Auch Hosts, die eine Sammlung von anderen Webressourcen im WWW bereitstellen, sind als Webressource zu bezeichnen. Vereinfacht gesagt können alle digitalen Informationen als Webressource bezeichnet werden, die über einen *Uniform Resource Locator (URL)* identifiziert werden. Berners-Lee, Fielding und Masinter definieren im RFC 3986 eine Ressource wie folgt:

> „... the term *resource* is used in a general sense for whatever might be identified by a URI. Familiar examples include an electronic document, an image, a source of information with a consistent purpose (e.g., today's weather report for Los Angeles), a service (e.g., an HTTP-to-SMS gateway), and a collection of other resources. ... [BLFM05]"

Ein *Uniform Resource Identifier (URI)* ist eine Folge von Zeichen, die eine Ressource, nicht zwingend eine Webressource, identifizieren [BLFM05]. Die Identifikation kann durch einen Namen oder einen Ort, einen *Locator*, geschehen (Tabelle 1). *Uniform Resource Locator (URL)* sind eine Teilmenge der URIs, die mit Hilfe von Adressen im Internet eine Webressource und damit dessen Speicherort identifizieren (Tabelle 2). Da die meisten URIs auch ein URL sind, werden die Begriffe URI und URL häufig synonym verwendet.

Sehr häufig werden neben dem Begriff der Webressource auch die Begriffe *Webseite (engl. Webpage)* und *Website* verwendet [KR05]. Eine *Webseite* besteht aus einem Hypertext-Dokument, das auf mehrere andere Webressourcen referenzieren kann. Dagegen ist eine *Website* eine Sammlung von Webressourcen oder Webseiten, die untereinander durch Verweise verbunden sind. Darüber hinaus werden diese meist von einem Host bereitgestellt, so dass alle Webressourcen einer Website mit demselben Präfix von URL beginnen.
Die verschiedenen Arten des Inhalts von Webressourcen werden durch *Multipurpose Internet Mail Extension-Types (MIME-Types)* charakterisiert [IAN07]. Es gibt 8 Hauptkategorien, die einen Medientypen beschreiben. Alle MIME-Typen werden als Untertypen in die 8 Hauptkategorien klassifiziert. Hypertexte im HTML-Format finden sich z.B. in der Hauptkategorie *text* wieder und werden mit *text/html* identifiziert (Tabelle 3). Neue MIME-Typen können jederzeit bei der *Internet Assigned Numbers Authority (IANA)* registriert werden. Dies erfordert jedoch ein Prüfungsverfahren, in dem die IANA verschiedene festgelegte Anforderungen an MIME-Types überprüft.

Uniform Resource Identifier (URI)
http://www.ietf.org/rfc/rfc3986.txt?number=3986 ftp://ftp.is.co.za/rfc/rfc3986.txt mailto:Max.Mustermann@mustermann.de urn:oasis:names:specification:docbook:dtd:xml:4.1.2 tel:+0049-0123-456789

Tabelle 1: Beispiele für Uniform Resource Identifier (URI)

Uniform Resource Locator (URL)
http://www.ietf.org/rfc/rfc3986.txt?number=3986 http://www.uni-duesseldorf.de/ http://www.uni-duesseldorf.de/logo.jpg ftp://ftp.is.co.za/rfc/rfc3986.txt

Tabelle 2: Beispiele für Uniform Resource Locator (URL)

MIME-Typ aus Hauptkategorie/Untertyp	Dateiendung des MIME-Typs	Beschreibung des MIME-Typs
application/pdf	*.pdf	Adobe PDF Dokumente
audio/mp4	*.mp4	MP4-Audiodateien
image/jpeg	*.jpeg *.jpg *.jpe	JPEG-Bilddateien
message/news		Newsgroup-Nachrichten
model/vrml	*.wrl	VRML-Modeldateien
multipart/encrypted		verschlüsselte mehrteilige Daten
text/html	*.htm *.html *.shtml	HTML-Hypertextdokumente
text/xml	*.xml	XML-Dokumente
video/mpeg	*.mpeg *.mpg *.mpe	MPEG-Videodateien

Tabelle 3: Beispiele für MIME-Typen

2.1.3 Der Webgraph als Strukturbeschreibung des WWW

Die einfachste und am häufigsten verwendete Datenstruktur zur Beschreibung der Webstruktur ist ein *Graph*. Auch Tim Berners-Lee sprach in seinem Entwurf zum Web schon von Knoten und Pfeilen als Links zur Repräsentation seines vernetzten Informationssystems, das später zum WWW werden sollte [BL90].
Eine durch Kanten verbundene Menge von Knoten wird als Graph bezeichnet [SS04]. Im Webgraph entsprechen alle Webressourcen den Knoten und alle Links zwischen den Webressourcen den Kanten [BKM+00]. Alle Kanten des Webgraphs haben eine Richtung, da z.B. ein Link von Ressource *a* auf eine andere *b* verweist. Infolgedessen ist der Webgraph

ein *gerichteter Graph*. Außerdem kann ein Link, also eine Kante, zwischen zwei Ressourcen *a* und *b* sowohl mehrfach vorkommen als auch in beide Richtungen verlaufen. Eine Kante von *a* auf sich selbst gerichtet, ist ebenfalls möglich. Der Webgraph ist daher ein *zyklischer Graph*. Der gerichteten, zyklischen Webgraphen G_{WWW} wird durch ein Tupel $G_{WWW} = (V, E)$ definiert, wobei V die endliche Menge der Webressourcen als Knoten und E die Menge der Links als Kanten darstellt [SS04]. Das Tupel $(a, b) \in E$ mit $a, b \in V$ identifiziert also eine Kante von a nach b. Abbildung 2 zeigt beispielhaft den Webgraphen einer geringen Menge von Webressourcen.

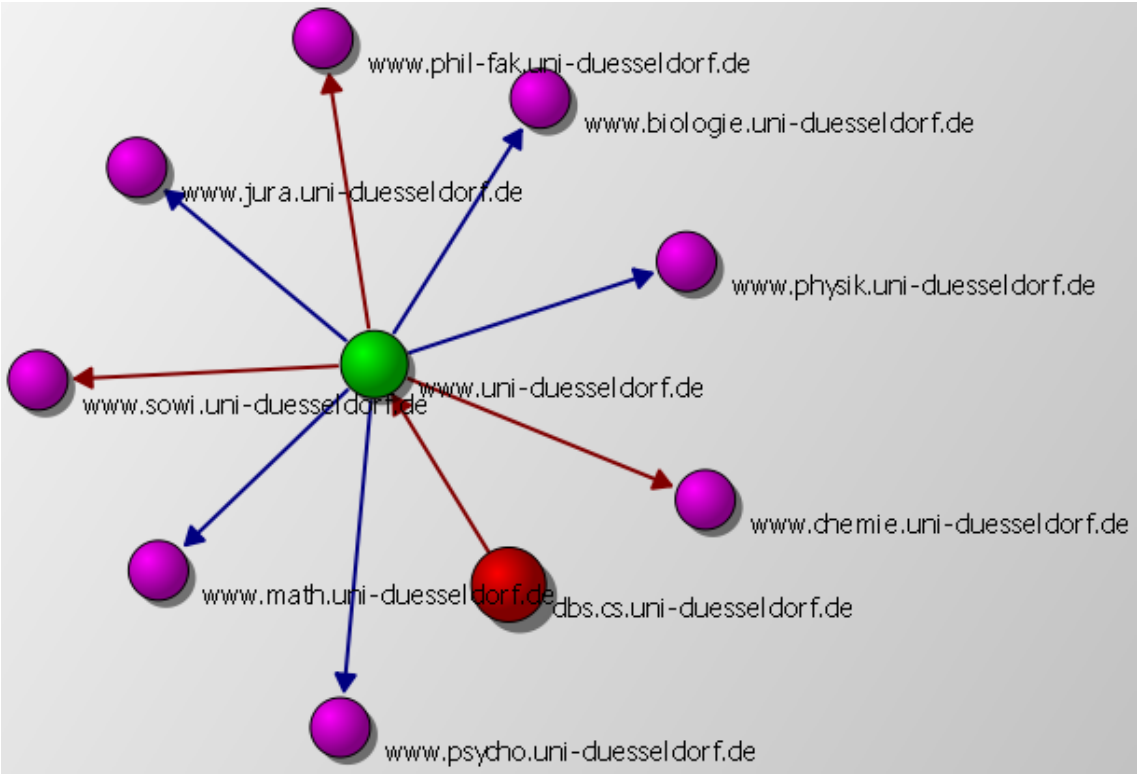

Abbildung 2: Ein Webgraph erstellt mit *wwwVis*

2.1.4 Die Struktur des WWW

Alle Ressourcen und Links des WWW lassen sich zwar mit Hilfe eines Webgraphen modellieren, die tatsächliche Struktur des Webs hingegen ist schwer zu ermitteln. Das Web hat eine riesige, komplexe und unkontrollierbare Größe [AJB99]. Es ist, im Gegensatz zu Telefon- oder Stromversorgungsnetz, ein virtuelles Netz aus Webressourcen und Links, welches stetig an verschiedenen Stellen wächst und von vielen Millionen Menschen unkoordiniert geändert und beeinflusst wird [KL01]. Daher glaubte man zunächst auch, dass sich überhaupt keine Strukturen im WWW erkennen lassen.
Selbst bekannte Suchmaschinen können nur einen Teil des Webs erfassen. Die Analyse der Webstruktur ist jedoch wichtig, um zum einen die vorhandenen Informationen, durch Links erzeugte Hierarchien und themenverwandte Gebiete im Web zu ermitteln

und zu verstehen, sowie zum anderen die Webstruktur für Anwendungen im Information Retrieval zu nutzen. Web-Crawler, Suchmaschinen oder andere Informationsfilter können durch Kenntnis der Strukturen des Webs verbessert werden. Darüber hinaus können Erkenntnisse über soziale Strukturen im Web ausgenutzt werden, um die Kommunikation zwischen Interessensgemeinschaften zu vereinfachen.
Studien belegen, dass das Web sich in gewissem Maße selbst organisiert und ordnet. Eine Untersuchung zur Graphenstruktur des Webs von Mitarbeitern der Firmen AltaVista, IBM und Compaq [BKM+00], mit 200 Millionen Websites und 1,5 Billionen Links als Testdaten, führte im Jahr 2000 zu dem Ergebnis, dass wenn das Web als Ganzes betrachtet wird, es einen Kern mit den bedeutensten Sites des Webs als Zentrum besitzt (*„strongly connected component (SCC)“*). Die Elemente des Kerns sind zusammenhängend, d.h. innerhalb dieses Kerns können alle Websites jedes andere Element des Kerns über einen Pfad von Links erreichen. Die übrigen Websites können in weitere Kategorien eingeteilt werden. Es gibt Sites die eine Verbindung mit dem Kern haben: die vorgelagerten Sites (*„IN“*) und die nachgelagerten Sites (*„OUT“*). Die *IN-Websites* linken auf Elemente des Kerns, während Websites aus dem Kern auf *OUT-Sites* linken. Einen Link von *IN-* nach *OUT-Sites* nennt man *tube*. Darüber hinaus gibt es Webressourcen, die keine Verbindung mit dem Zentrum des WWW haben, aber durchaus mit den *IN-* oder *OUT-Sites* verbunden sind. Diese bezeichnen die Mitarbeiter der Firmen als *tendrils*. Außerhalb jeglicher Struktur befinden sich die Ressourcen des Webs, die mit keiner der vorher genannten Einheiten verbunden sind (*„Disconnected components“*). Die Testergebnisse aus [BKM+00] zeigen, dass die vier Komponenten *SCC, IN, OUT* und *tendrils* in etwa die gleiche Menge an Webressourcen enthalten, in der Untersuchung ca. 45 Millionen. Die *disconnected components* hingegen sind ein kleinerer Teil des WWW.
Betrachtet man allerdings nur einen kleinen Ausschnitt des Webs, in dem Webressourcen nur wenige Links entfernt auseinander liegen, ist es wesentlich schwieriger Strukturen zu bestimmen [KL01]. Ein uneinheitliches und kompliziertes Gefüge erschwert hier eine effiziente Vermessung der Struktur des Webs. Handelt es sich jedoch um Webressourcen, die ein gemeinsames Thema oder gleiche Interessen verfolgen, ist zu erkennen, dass diese eine höhere Linkdichte haben. Sie fügen sich zu einem Cluster aus gemeinsamen Interessen zusammen, das wiederum eigene Strukturen aufweist. Folglich ist umgekehrt eine hohe Linkdichte auch ein Zeichen für eine themenverbundene Menge von Sites. Ein einfaches Beispiel für themenverbundene Ressourcen ist eine einzelne Website. Sie verfolgt gleiche Interessen und hat eine sehr hohe Linkdichte. Werden Bücher oder Publikationen im Web inspiziert, die durch Links hierarchisch geordnet sind, so ist ebenfalls eine hohe Linkdichte zwischen Kapiteln und Unterkapiteln zu erkennen. Aber auch mehrere themenverbundene Websiten können zu einem Cluster im WWW zusammenwachsen. Solch eine *„Community“ [KL01]* wäre z.B. eine Menge von Websites, die von einem Themenportal aus erreicht werden können. Kleinberg und Lawrence formulieren dies allgemein:

> „A characteristic pattern in such *communities* consists of a collection of *hub pages* linking in a correlated fashion to a collection of *authorities* on a common topic. A related pattern is one in which *authorities* on a topic link directly to other *authorities*, again creating a density of links [KL01].“

Hub pages sind Websites, die auf viele andere linken, *authorities* dagegen sind Websites, auf die viele andere Websites verweisen [Day99]. Eine themenverwandte, verlinkte

Sammlung aus *hubs* und *authorities* bildet also eine *Community*.
Ein weiterer Ansatz, um themenverwandte Websites und damit eine *Community* zu ermitteln, ist, dass nur diejenigen Sites zur *Community* gehören, die mehr Links auf Websites innerhalb als auf außerhalb des Themenverbundes liegende haben:

> „A *community* can also be defined as a collection of pages in which each member page has more links to pages within the community than to pages outside the community [KL01].“

Diese Definition ist besonders sinnvoll für *Communities*, in denen oft neue Sites eingegliedert oder ausgegliedert werden und deren Struktur sich häufig ändert.
Zusammenfassend bleibt festzuhalten, dass die Analyse der Struktur des Webs, besonders unter Betrachtung sozialer Aspekte, schwierig, aber dennoch möglich ist. In diesem Forschungsgebiet gibt es regelmäßig neue Erkenntnisse, da sich die Struktur ändert und weiterentwickelt.

2.1.5 The Small World Phänomen

Trotz der enormen Größe des gesamten WWW kann der Kern als kompakte, kleine Welt *(„small world“ [KL01])* angesehen werden. Der kürzeste Weg von einer Website zu einer anderen im Kern verläuft durchschnittlich über 16-20 Links [BKM+00]. Erste Untersuchungen ergaben einen Wert von 19 [AJB99], doch die bereits erwähnte Studie zur Graphenstruktur des Webs von Mitarbeitern der Firmen AltaVista, IBM und Compaq [BKM+00] zeigte, dass der Durchschnitt nur etwa 16 Links beträgt, wenn ein kürzester Weg wirklich existiert. In vielen Fällen existiert allerdings kein Pfad von einem Startpunkt zu einem Endpunkt. Könnte man Links in beide Richtungen verfolgen, würde die Distanz zwischen zwei zufälligen Webressourcen im Kern sogar nur durchschnittlich ca. 7 Links betragen. Auch die Studie von Lada A. Adamic [Ada99] bestätigt die These von einer *small world* im Kern des WWW.
Der Abstand von ca. 16 Links kann als *Durchmesser des Webs [AJB99]* interpretiert werden, wobei sich dieser, wegen der logarithmischen Abhängigkeit von der Größe des gesamten WWW, auch bei enormer Vergrößerung des Webs nur minimal ändern würde. Mit abkürzenden Links lässt sich der Durchmesser sogar noch verkleinern, so dass man wirklich von einer *small world* sprechen kann, in der recht schnell von einer Seite zur anderen navigiert werden kann.

2.2 Daten- und Informationsvisualisierung

2.2.1 Daten, Informationen und Wissen

Auch *Daten* und *Informationen* sind zwei unterschiedliche Begriffe. Oft ist es jedoch schwierig diese sauber zu trennen, da Informationen aus Daten bestehen.
Grundelemente von *Daten* sind *Zeichen*. Diese können z.B. lateinische, griechische oder kyrillische Buchstaben, Symbole oder Ziffern sein und haben alleingestellt noch keine Bedeutung. Werden diese Zeichen nach bestimmten Regeln oder Mustern zusammengestellt, so werden aus den Zeichen Daten [Nor99]. Daten haben einen physikalischen Bezug, d.h. Daten eines bestimmten Datentyps, beispielsweise Strings oder Integer, werden heute in Datenbanken oder auf anderen Datenspeichern, gesichert [Däs99]. Meßdaten aus wissenschaftlichen Experimenten oder Computersimulationen sind dazu oft an räumliche oder zeitliche Dimensionen gebunden.
Daten werden von verschiedenen Betrachtern unterschiedlich interpretiert. Ihnen kommt so erst eine Bedeutung zu. Aus Daten entstehen Informationen, wie North beschreibt:

> „Informationen sind also Daten, die in einem Bedeutungskontext stehen und [...] zur Vorbereitung von Entscheidungen und Handlungen dienen ([Nor99] S.40).“

Informationen sind daher abstrakte Daten. Dokumente, Begriffe oder andere textbasierte abstrakte Daten aus Datenbanken, Retrievalsystemen oder dem Web sind durch verschiedene Interpretationsmöglichkeiten von unterschiedlichen Betrachtern als Informationen anzusehen.
Verbindet ein Betrachter die Informationen mit seinen Erfahrungen sowie Erwartungen und stellt sie so in seinen persönlichen Kontext, dann wird aus Information Wissen. Wissen ist laut Probst wie folgt definiert:

> „Wissen bezeichnet die Gesamtheit der Kenntnisse und Fähigkeiten, die Individuen zur Lösung von Problemen einsetzen. Dies umfasst sowohl theoretische Erkenntnisse, als auch praktische Alltagsregeln und Handlungsanweisungen. Wissen stützt sich auf Daten und Informationen, ist im Gegensatz zu diesen jedoch an Personen gebunden ([PRR99] S.46).“

Verschiedene Betrachter können also aus denselben Informationen unterschiedliches Wissen für sich selbst gewinnen.
Die Menge an verteilten Informationen in lokalen Netzwerken oder dem WWW wächst von Zeit zu Zeit sehr stark an [Däs99]. Eine Folge dieser *Informationsflut* ist die Abnahme des Informationsgehalts, d.h. es ist fraglich, ob Informationen wiederauffindbar sind und somit die relevanten Informationen zur richtigen Zeit an den richten Ort gelangen können. Eine Möglichkeit dem entgegenzuwirken ist die Analyse, Filterung und Aufbereitung von Daten durch *Data-Mining Systeme*.

2.2.2 Data Mining

Um aus der immer größer werdenden *Informationsflut* noch relevante Informationen zu filtern, interessante Zusammenhänge zu entdecken und Wissen zu gewinnen, das später gezielt eingesetzt werden kann, können statistische Auswertungsverfahren auf Datenbeständen, beispielsweise in Datenbanken, angewandt werden. Dieser Prozess der automatischen Wissensfindung wird *Knowledge Discovery in Databases (KDD)* genannt:

> „Der Begriff *Knowledge Discovery in Databases (KDD)* kann als der nichttriviale Prozess der Identifikation gültiger, neuer, potenziell nützlicher und verständlicher Muster in Datenbeständen definiert werden [SH00]."

Der iterative und interaktive Prozess des *KDD* besteht aus sechs Teilschritten: der Festlegung des Problembereichs und der Ziele, der Datensammlung und -bereinigung, der Auswahl und Parametrisierung der Analysefunktionen und -methoden, dem *Data Mining*, der Bewertung und Interpretation der Ergebnisse sowie der Nutzung des gefundenen Wissens [SH00].
Data Mining ist demzufolge ein Teilschritt des *KDD* und wird von Saake und Heuer wie folgt definiert:

> „Unter *Data Mining* versteht man die Anwendung verschiedenster Modellierungs- und Entdeckungstechniken, um Wissen aus Daten zu gewinnen [SH00]."

Versucht man Wissen nicht nur aus herkömmlichen Daten zu generieren, sondern wendet spezielle Daten an, so wird dies als *Document/Text Mining* (bei Textdokumenten), *Image Mining* (bei Bilddaten), *Spatial Data Mining* (bei geographischen Daten) oder *Web Mining* (bei Daten aus dem WWW) bezeichnet.
Neben der Analyse von herkömmlichen oder speziellen Daten und der Anwendung von statistischen Verfahren sind Daten- und Informationsvisualisierung weitere unterstützende Methoden um Wissen aus Daten zu gewinnen.

2.2.3 Was versteht man unter Daten- und Informationsvisualisierung?

Visualisierung überführt Daten, Informationen oder auch gesammeltes Wissen in visuelle Darstellungsformen [Däs99]. Die Visualisierung gibt es im Grunde schon seit der Steinzeit in Form von Höhlenmalereien. Auch Seefahrer, Astronomen, Meteorologen, das Militär und viele andere nutzten seitdem die Vorteile der Visualisierung. Mit Einführung der Computersysteme erweiterten sich die Möglichkeiten der Visualisierung noch mal um ein Vielfaches [Sch04]. Mit keiner anderen Methode als der Visualisierung lassen sich so viele verschiedene Aspekte und Beziehungen kenntlich machen, verdeutlichen und vereinfachen. Mit Hilfe einer Visualisierung können sowohl Daten oder Informationen analysiert und interpretiert, als auch Muster oder Regeln erkannt und kontrolliert werden. Bilder und Graphiken können den Betrachter faszinieren und auf ihn einwirken [BYRN99]. Ebenso ist es psychologisch erwiesen, dass der Mensch Informationen besser aufnehmen und sich einfacher merken kann, die visualisiert wurden [Däs99].

Wird also von Daten- oder Informationsvisualisierung gesprochen, sind die Konzepte, Methoden und Anwendungen zur vereinfachten Darstellung und Aufbereitung von Daten bzw. Informationen aus einem Daten- bzw. Informationsspeicher, vorwiegend aus Datenbanken oder digitalen Dokumentsammlungen, gemeint. Die Informationsvisualisierung ermöglicht im Gegensatz zu traditionellen Informationssystemen die Filterung von Informationen, das Erkennen von Mustern, Regeln oder Abhängigkeiten sowie verschiedene Sichten auf eine bestimmte Menge von Informationen. Außerdem ist aufgrund von visualisierten Informationen ein Vergleich mit, bzw. die Darstellung im Kontext zu anderen Informationen möglich. Die Möglichkeit der Verarbeitung von inhomogenen oder verrauschten Daten ist ein weiterer Vorteil der Informationsvisualisierung. Ferner ist für die Nutzung auch nicht zwangsläufig Expertenwissen notwendig, da komplexe mathematische oder statistische Algorithmen im Hintergrund ablaufen und für den Nutzer der Visualisierung nicht sichtbar sind [Kei02].
Die Daten- und Informationsvisualisierung wird heute in vielen Bereichen eingesetzt. In der Betriebswirtschaft, Informatik, Biologie, Chemie, Physik, Mathematik, Medizin, Geographie und in vielen anderen Bereichen findet die Visualisierung ihre Anwendung.
Im Folgenden zwei Beispiele für Daten- bzw. Informationsvisualisierung:
Abbildung 3 zeigt 40.000 Erdbebenepizentren unterschiedlicher Intensität, gemessen in verschiedenen Jahren, veranschaulicht mit verschiedenen Farben als Beispiel für eine Datenvisualisierung. Ohne die Visualisierung auf einer Erdkugel wäre es nicht möglich gewesen, die Ränder der Kontinentalplatten zu identifizieren. In einer genaueren Untersuchung der 3D-Visualisierung ist beispielsweise auch zu erkennen, dass Erdbeben in manchen Gebieten bis zu einer Tiefe von 1000km registriert werden.

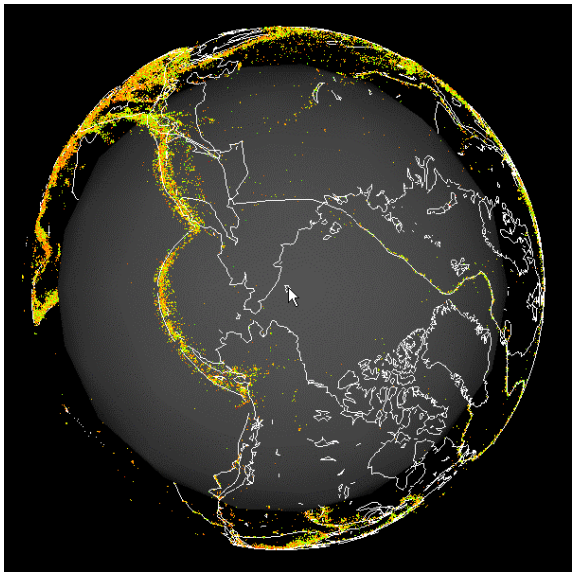

Abbildung 3: Beispiel für eine Datenvisualisierung: Globale Verteilung von Erdbebenepizentren (aus [Däs99])

Abbildung 4 zeigt einen Dateibaum einer Festplatte als Beispiel für eine Informationsvisualisierung. Diese Darstellung gewährleistet einen schnellen Zugriff auf Verzeichnisse und Dateien der Festplatte. Auch hier wären eine vorteilhafte, interaktive Untersuchung

der Verzeichnisstruktur und die Visualisierung der einzelnen Verzeichnisse im Kontext zu anderen ohne die Informationsvisualisierung nicht möglich.
Die Daten- oder Informationsvisualisierung liefert also einen *Informationellen Mehrwert*.

Abbildung 4: Beispiel für eine Informationsvisualisierung: Hyperbolic Tree - Visualisierung des Dateibaums einer Festplatte (aus [Däs99])

2.2.4 Visualisierung: Eigenschaften und Anforderungen

Ziel der Daten- und Informationsvisualisierung ist es, nach den Bedürfnissen des Nutzers und mit den Möglichkeiten der Visualisierung, eine optimale graphische Darstellung zu erreichen.
Um eine ideale Visualisierung zu gewährleisten müssen zunächst abstrakte Daten auf geometrische Darstellungsformen abgebildet werden [Sch04]. Das *Information Visualization Data-State-Reference Model* (Abbildung 5) von Chi [Chi00] beschreibt den Vorgang der Transformation von Informationen in geometrische Darstellungsformen. Als erstes werden *Daten* extrahiert (*Filtering*) und in *Analytische Abstraktionen* überführt. Oft sind dies Metadaten, die den Inhalt der eigentlichen Daten beschreiben. Das System hat die Kontrolle über den Datenraum. Der Datenraum wird durch das so genannte *Mapping* in den Darstellungsraum verlassen. Durch *Mapping* werden Metadaten-Informationen visualisiert und es entstehen *Visuelle Abstraktionen*. Der Endzustand, das *Bild* oder die Ansicht, den der Nutzer einer Anwendung letztendlich auf dem Bildschirm oder dem Papier sieht, wird durch *Rendering* der *Visuellen Abstraktionen* erreicht. Der Darstellungsraum wird vom Nutzer bestimmt und verändert. Die Informationsvisualisierung ist daher ein interaktiver Prozess zwischen System und Nutzer [SM04]. Auf dem Weg von Daten zu Bildern können verschiedenste Operatoren bzw. Methoden zur Überführung und Berechnung in den unterschiedlichen Stufen angewandt werden. So ist es durchaus möglich, dass unterschiedliche Bilder von derselben Datenmenge erzeugt werden.
Operatoren, die im Datenraum arbeiten, könnten Formatkonvertierungen, Bereinigungen, Filter oder das Berechnen charakteristischer Merkmale durch statistische Methoden sein. *Visuellen Abstraktionen* werden häufig mit Farben, unterschiedlichen Größen oder verschiedenen Positionen bearbeitet. Zoom-Funktionen, verschiedene Farbskalen, Kontrastveränderungen oder aber ein Scrollen des Bildes sind die gängigsten Operatoren auf

dem Endzustand, dem *Bild*.
Das *Information Visualization Data-State-Reference Model* kann auch auf das WWW angewandt werden. Die *Daten* des WWW sind Webressourcen, die mit Links verknüpft sind. Die *Analytische Abstraktion* der Webressourcen ist ein Graph. Ein Graph kann mit einer Tiefen- oder Breitensuche durchlaufen werden, um so eine Hierarchie der Webressourcen aufzustellen. Die Hierarchie entspricht der *Visuellen Abstraktion*, die schließlich in einer Baumdarstellung als *Bild* angezeigt werden kann. Operatoren der Baumdarstellung könnten z.B. das Rotieren des Bildes oder das Hineinzoomen in die Baumdarstellung sein.

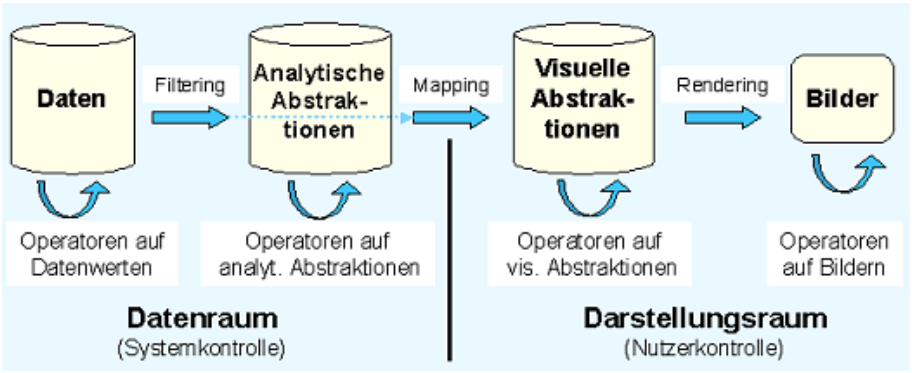

Abbildung 5: *Information Visualization Data-State-Reference Model* nach [Chi00] (aus [Sch04])

Darüber hinaus sollten die Möglichkeiten der Visualisierung optimal ausgenutzt werden. Shneiderman formuliert das *Mantra der Informationsvisualisierung* [Shn96] als Regeln für eine optimale Visualisierung von Daten und Informationen:

> „Overview first, zoom and filter, then details-on-demand [Shn96]“

Das Mantra spezifiziert den idealen Ablauf einer Informationsvisualisierung. Von einer Informationsmenge können verschiedene Ergebnisbilder dargestellt werden, die beispielsweise über ein Menü oder andere Interaktionsmöglichkeiten ausgewählt werden können.
Eine optimale Informationsvisualisierung liefert zunächst einen Überblick über die allgemeinen Eigenschaften einer Informationsmenge. Dieser *Overview* dient zur ersten Orientierung mit den unbekannten Informationen. *Zoom in* und *zoom out* Funktionen bieten die Möglichkeit einen bestimmten Bereich der Informationsmenge zu vergrößern und zu fokusieren. *Filteroperationen* verringern die Informationsmenge, so dass nur die wirklich relevanten Informationen, die zur Lösung eines Problems in Frage kommen, betrachtet werden. Zu guter Letzt sieht Shneiderman in seinem Mantra vor, dass auf Wunsch ein Visualisierungsergebnis mit beliebigem Detailgrad erzeugt werden kann (*Details on Demand*). Wählt der Benutzer ein Element der Informationsmenge aus, so sollten Details des Elements angezeigt werden, wenn diese vom Nutzer gewünscht sind.
Neben dem *Information Visualization Data-State-Reference Model* und dem *Mantra der Informationsvisualisierung* gibt es noch drei weitere Kriterien, die für eine erfolgreiche Informationsvisualisierung von Bedeutung sind und befolgt werden sollten [Sch04]: die *Effektivität*, die *Expressivität* und die *Angemessenheit* der Visualisierung. Eine Visualisierung

kann nur effektiv sein, wenn sie intuitiv wahrgenommen werden kann. Ferner sollten zur Einhaltung der Expressivität keine falschen oder irreführenden Informationen, die durch falsche Farbwahl oder falsche Größen entstehen können, visualisiert und vermittelt werden, sondern nur die in einer Datenmenge enthaltenen Informationen. Ebenso ist es für die Angemessenheit der Visualisierung wichtig, dass sie für den Nutzer in akzeptabler Zeit abläuft. Die Anforderungen an die Leistungsfähigkeit eines Computersystems sollten zudem nicht zu hoch sein, sowie die Darstellung der Informationen nur in notwendiger Genauigkeit erfolgen. Nur so kann ein effektiver Nutzen aus der Visualisierung erziehlt werden.
Bei der Entwicklung einer *graphischen Benutzeroberfläche* (kurz: *GUI* für *graphical user interface*), die eine Daten- oder Informationsvisualisierung beinhaltet, sind die verschiedenen Fähigkeiten, Kenntnisse und Preferenzen der Nutzer zu berücksichtigen [BYRN99]. Gegebenenfalls können auch Altersunterschiede oder kulturelle Besonderheiten eine Rolle für die Entwicklung einer graphischen Benutzeroberfläche spielen. Aufwendige oder komplexe Visualisierungen können für einige Nutzer innovativ, für andere hingegen hinderlich oder gar lästig sein. So sollte ein GUI wünschenswerterweise mehrere Varianten, z.B. bei der Menüführung, anbieten.
Die Anwendbarkeit, oft als *Usability* bezeichnet, der GUI und der damit verbundenen Visualisierungen spielt ebenfalls eine große Rolle. Die Frage nach der geeigneten Wahl der Visualisierungstechniken ist der zentrale Aspekt der *Usability*. Zur Überprüfung der Anwendbarkeit von graphischen Benutzeroberflächen und Visualisierungen werden oft empirische Untersuchungen zu Rate gezogen.

2.2.5 Möglichkeiten und Techniken der Visualisierung

Es gibt viele verschiedene Methoden und Techniken aus den Bereichen Daten- und Informationsvisualisierung, die bei einer Visualisierung zum Einsatz kommen können. Zudem werden regelmäßig neue Methoden entwickelt, die den steigenden Anforderungen und neuen Computerentwicklungen gerecht werden. Bekannt sind Standard-Techniken wie Balken-, Linien- oder Kreisdiagramme. Darüber hinaus sind insbesondere Methoden der platzsparenden Anzeigetechnik, Techniken zur Visualisierung der Strukturierung von Informationsbeständen, sowie zur Visualisierung der Elemente eines Informationsbestandes und intuitive Interaktionstechniken besonders hilfreich [SM04].
Platzsparende Anzeigetechniken: Ein Problem vieler Visualisierungen ist eine riesige Datenmenge als Grundlage, die nicht vollständig in einem Bild dargestellt werden kann. Eine Lösung dieses Problems liefern die beiden Techniken *Übersicht und Detail* und *Focus-and-Context* zur platzsparenden Anzeige. Beide Techniken werden heute in vielen Visualisierungen eingesetzt und sind oft unverzichtbar. Die *Übersicht und Detail*-Technik besteht aus einem Bild mit Übersichtsinformationen und einem Bild mit Detailinformationen, so dass die Menge an Informationen übersichtlicher dargestellt werden kann.
Die *Focus-and-Context*-Visualisierungstechnik hebt die im Fokus liegenden Teile der Visualisierung hervor und blendet deren Details ein, ohne aber den Gesamtkontext zu vernachlässigen. Eine andere Variante der *Focus-and-Context*-Visualisierung sieht eine Verzerrung der Ansicht vor, bei der entweder der Fokus vergrößert oder der Gesamtkontext verkleinert dargestellt wird. Weitere Erläuterungen im nächsten Kapitel *Focus-and-Context Visualisierung*.

Techniken zur Visualisierung der Strukturierung von Informationsbeständen: Um die Strukturierung einer Informationsmenge zu verdeutlichen, werden häufig Visualisierungstechniken genutzt, die einerseits das Navigieren auf den Strukturen ermöglichen und andererseits die Visualisierung nur so komplex wie nötig halten. Oft werden nur Hierarchien der Informationsmenge dargestellt. Man differenziert hier zwischen expliziten oder implizierten, zwischen achsenparallelen oder radialen und zwischen 2D oder 3D Techniken. Die hierarchischen Strukturen der Informationsmenge werden entweder explizit, z.B. durch Kanten, oder implizit durch spezielle Anordnungen aufgezeigt. Achsenparallele Visualisierungen stellen die Hierarchie von oben nach unten oder rechts nach links Ebene für Ebene dar, während radiale graphische Darstellungen ihre Elemente um den Wurzelknoten herum ausrichten. Außerdem gibt es Techniken, die die 3-Dimensionalität ausnutzen und welche, die sich auf 2D Darstellungen beschränken.
Techniken zur Visualisierung der Elemente eines Informationsbestandes: Eine weitere Klasse von Techniken beschäftigt sich mit der Visualisierung der Elemente einer Informationsmenge. Diese werden, als in Beziehung zueinander stehende Objekte, mit bestimmten Eigenschaften veranschaulicht. Häufig ähneln sich mehrere Objekte. Zur Visualisierung verwandter Elemente sind ebenfalls verschiedene Techniken, abhängig von der Informationsmenge, vorgesehen. Gleiche Farben, gleiche Größen oder aber ein Gruppieren von ähnlichen Objekten, z.B. durch Anziehung oder Abstoßung selbiger zur Anordnung im Gesamtkontext, können Teil einer solchen Visualisierung sein.
Intuitive Interaktionstechniken: Bei den intuitiven Interaktionstechniken wird zwischen Techniken zur Navigation, zur Anfrage und Selektion, zur Steuerung des Mappings und zur Manipulation im Darstellungsraum unterschieden. Sie erleichtern für den Benutzer das Zusammenspiel mit der Anwendung bzw. der Informationsmenge. Zur Navigation in riesigen Informationsmengen dienen oft Techniken, mit denen der Nutzer Elemente des Datenraumes bewegen und anordnen kann. Entsprechend dem *Mantra der Informationsvisualisierung* können dann Details ein- und ausgeblendet werden. Methoden zur Anfrage und Selektion von bestimmten Elementen der Informationsmenge können sowohl auf den Daten- als auch auf den Darstellungsraum angewandt werden. In einigen graphischen Benutzeroberflächen werden auch Techniken zur Steuerung des Mappings, also verschiedene Abbildungen der abstrakten Daten auf Visuelle Abstraktionen (vgl. *Information Visualization Data-State-Reference Model*), eingesetzt. Zur graphischen Repräsentation sind dann verschiedene Objekte, z.B. Kreise oder Rechtecke, anwählbar. Der Benutzer kann nun, abhängig von seinen Vorlieben oder dem Zweck der Visualisierung, selbst entscheiden, welche graphische Abbildung er einsetzt. Spezialfälle sind Abbildungen auf Icons oder Pixel, die Techniken der *Icon-basierten-* oder *Pixel-Visualisierung* einsetzen [Kei02].
Neben den platzsparenden Anzeigetechniken gibt es noch weitere Methoden zur Beeinflussung des Darstellungsraums: Das *interaktive Rearrangement* ermöglicht die Anordnung der Bilder auf unterschiedlichen Wertebereichen oder in unterschiedlichen Umgebungen. So können Muster oder Regeln auf der Informationsmenge entdeckt werden. Die *Semantische Lupe* zeigt Details on Demand im Gesamtkontext.
Ansonsten gibt es noch viele weitere Möglichkeiten und Techniken der Visualisierung, die jeweils stark von der Datenmenge, den Nutzern oder auch von den verfügbaren Computersystemen abhängen. Jede Visualisierungstechnik hat Vor- und Nachteile. Es wird in erster Linie versucht die Vorteile der Techniken zu nutzen und zu kombinieren, sowie die Nachteile der Techniken auszugleichen. Eine Kombination aus mehreren Techniken

der Visualisierung nennt man *Interaktives Linking and Brushing* [Kei02]. Dabei werden die verschiedenen Visualisierungen verknüpft und zur Unterscheidung eingefärbt. In einer verknüpften Visualisierung kann häufig mehr erkannt werden als in den jeweiligen Einzelvisualisierungen, da so durch Vergleich der Visualisierungen Abhängigkeiten erkannt oder Folgerungen gezogen werden können. Balken-, Streudiagramme, Pixel- und Iconvisualisierungen werden häufig per *interaktivem Linking and Brushing* kombiniert.

2.2.6 Focus-and-Context Visualisierung

Eine spezielle Klasse von Visualisierungstechniken, die zur platzsparenden und übersichtlicheren Anzeige dienen, sind die *Focus-and-Context-Visualisierungen*, im Folgenden abgekürzt mit *FC-Visualisierungen*. Besonders häufig werden diese bei Visualisierungen angewandt, die eine große Menge von Informationen als Grundlage haben. Kartendarstellungen aus der Geographie oder Graphendarstellungen, wie z.B. der Webgraph, sind besonders geeignet für diese Art von Veranschaulichung [SK03].
Rauschenbach, Weinkauf und Schumann definieren die *FC-Visualisierung* wie folgt:

> „*Focus-and-context techniques* have been proposed to solve the problem of displaying very large layouts on desktop workstations equipped with powerful processors. These techniques combine a *focus display*, which shows a part of the layout at a high degree of detail, and a *context display*, which presents the whole picture in lower detail to provide the overview. The position of the focus is determined by the current *point of interest* of the user [RWS00].“

Die *FC-Visualisierung* hebt also die im Fokus liegenden Teile der Visualisierung hervor und blendet gegebenenfalls deren Details ein, ohne aber den Gesamtkontext zu vernachlässigen. Eine andere Variante sieht eine Verzerrung der Ansicht vor, bei der entweder der Fokus vergrößert oder der Gesamtkontext verkleinert dargestellt wird. *Zoom in-* und *Zoom out-Funktionen* können ebenfalls als *FC-Techniken* angesehen werden [SM04], da dadurch ein animierter Wechsel zwischen Fokus und Kontext in der Visualisierung stattfindet. *Panning-Methoden* verfügen über effektivere Übergänge zwischen Kontext und Fokus oder umgekehrt, stehen aber erst am Anfang der Entwicklung.
Grundsätzlich wird zwischen drei Gruppen von *FC-Techniken* unterschieden: den *räumlichen*, den *dimensionalen* und den *hinweisenden* Methoden [KMH02].
Räumliche Methoden vergrößern den Fokus und verkleinern den Kontext, oft durch Verzerrungen. Beispiele sind *Fish-Eye-Views* und *Perspective Walls* (Abbildung 6), *Hyperbolic Trees*, *Document Lenses* und andere verzerrende Methoden. Auch in der Photographie werden die *räumlichen* Methoden häufig angewandt, um ein Objekt im Fokus deutlich und beispielsweise den Hintergrund nur unscharf darzustellen (Abbildung 6).
Dimensionale Methoden vergrößern den Detailgrad der im Fokus liegenden Objekte. Sie zeigen mehr Dimensionen, also mehr Informationen über die Objekte, an. Bekannte Beispiele sind *Magic Lenses* oder *Tool Glasses*.
Hinweisende Möglichkeiten der *FC-Visualisierung* heben die im Fokus liegenden Elemente durch Farb- oder Kontrastveränderungen hervor. Des Weiteren können insbesondere bei geographischen Darstellung auch Veränderungen der Schärfe oder der Transparenz von Ebenen ein Mittel zur Betonung des *point of interest* sein. Auch eine Technik, die ein Verschieben der Elemente vorsieht, kann als hinweisende *FC-Methode* eingesetzt werden.

Ausser den genannten Techniken gibt es eine unzählbare Menge von weiteren Techniken der *FC-Visualisierung*.

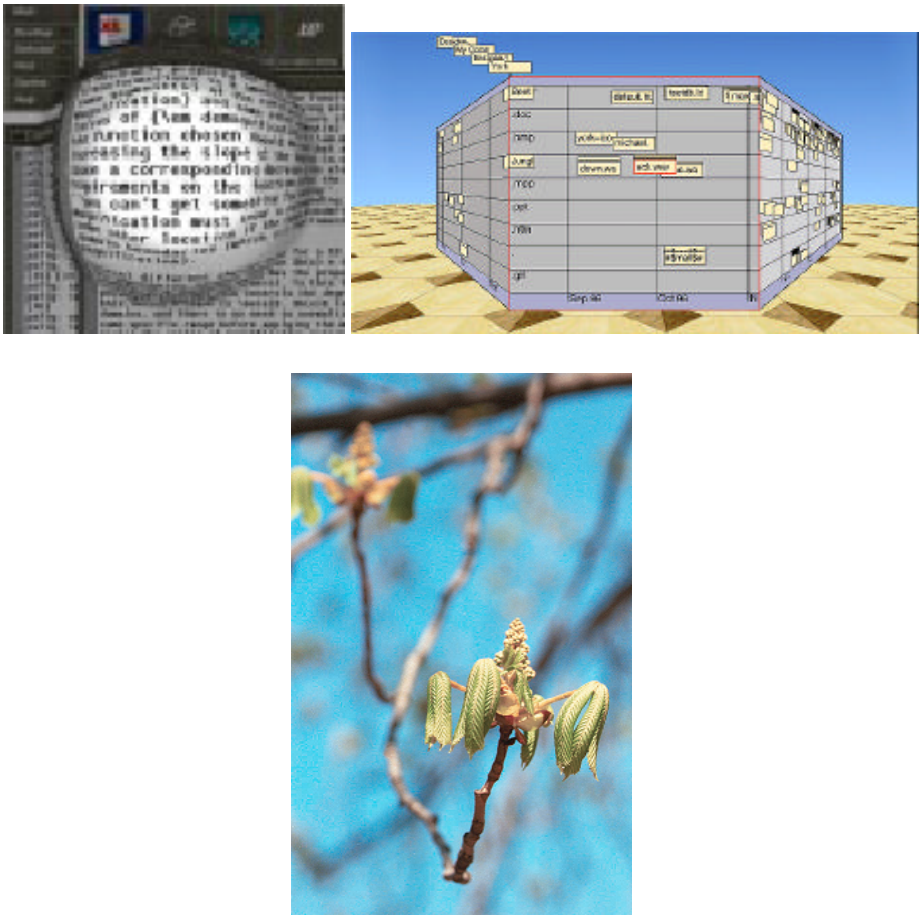

Abbildung 6: Beispiele für *Focus-and-Context Visualisierungen* mit Hilfe einer *Perspective Wall* (aus [SK03]), mit Hilfe des *Fish-Eye-View*-Effekts (aus [SM04]) und aus der Photographie (aus [KMH02])

3 Hilfsmittel und Basisdaten

Dieser Abschnitt soll einen Überblick über die verwendeten Hilfsmittel und Basisdaten, die zur Entwicklung von *wwwVis* benötigt wurden, liefern. Die Hilfsmittel Qt, Java und DB2 werden kurz beschrieben, sowie die Struktur, der in der Bachelorarbeit *Extraction and Storage of Web Structures* ermittelten Basisdaten, erläutert.

3.1 Die Programmiersprache Java

Java ist eine sauber definierte, plattformunabhängige Programmiersprache, die Aspekte der klassischen imperativen Programmierung und Konzepte der objektorientierten Programmierung verbindet [Krü03]. Im Jahr 1995 wurde *Java* von der Firma *SUN* auf den Markt gebracht und entwickelte sich seitdem zu einer der bekanntesten Programmiersprachen der Welt. Die inzwischen neueste Version ist das *Java Development Kit (JDK) 1.6.0*. *Java* besitzt eine umfangreiche Klassenbibliothek, die nützliche vordefinierte Klassen, wie z.B. Datenstrukturen oder Klassen zum Dateieinlesen und -auslesen, liefert und dem Programmierer so die Arbeit erleichtert. Die Klassenbibliothek beinhaltet dazu auch graphische Funktionen, die unter dem Begriff *Java Foundation Classes (JFC)* zusammengefasst werden. Die drei wichtigsten Komponenten der *JFC* sind das *Abstract Windowing Toolkit (AWT)* mit den elementaren graphischen und fensterbasierten Funktionen, das *Swing Toolset* mit zusätzlichen Dialogelementen und das *Java 2D API*, welches komplexere Graphik- und Bildbearbeitungsfunktionen zur Verfügung stellt. Außerdem gibt es mit der *Java Database Connectivity (JDBC)* eine brauchbare Bibliothek für den Zugriff auf relationale Datenbanken. Java ist demnach sehr gut zur Entwicklung und Implementierung kleinerer, aber auch umfangreicheren Projekten, geeignet und wurde deshalb für die Implementierung von *wwwVis* ausgewählt.

3.2 JDBC und DB2

Die *Java Database Connectivity (JDBC)* ist eine Programmierschnittstelle, die den Zugriff mit Java auf *Structured Query Language (SQL)*-Datenbanken ermöglicht [SH00]. *JDBC* ist vergleichbar mit *ODBC (Open Database Connectivity)* für die Programmiersprachen *C* und *C++*, ist aber deutlich übersichtlicher und einfacher zu benutzen. Die SQL-Anweisungen können direkt genutzt werden.
Alle *JDBC* Klassen sind in der Java-Klassenbibliothek über das Package `java.sql` zu erreichen. Eine Datenbankanfrage wird über die vier wichtigsten Klassen des Packages ausgeführt:
Mit Hilfe der Klasse `java.sql.DriverManager` wird ein Treiber eines Datenbanksystems geladen, so dass Verbindungen zur Datenbank hergestellt werden können. Über die Klasse `java.sql.Connection` kann eine Verbindung zur Datenbank aufgebaut werden. Anschließend sind SQL-Anweisungen mit der Klasse `java.sql.Statement` über eine zuvor aufgebaute Verbindung an die Datenbank zu senden. Das Ergebnis der Anfrage speichert *JDBC* in einer Relation, die durch die Klasse `java.sql.ResultSet` abrufbar ist. Neben der gesamten Relation sind, wie bei *SQL* üblich, auch nur einzelne Spalten oder einzelne Tupel abfragbar.

DB2 von *IBM* ist ein kommerzielles relationales Datenbanksystem, auf das mit der *JDBC*-Schnittstelle zugegriffen werden kann [SH00]. Es unterstützt relationale und XML-Daten und ist auf Servern mit verschiedenen Betriebssystemen wie Unix, Linux oder Windows lauffähig [IBM07]. Ferner nutzt *DB2* ebenfalls die Datenbankanfragesprache *SQL* und gewährleistet einen kontrollierten Mehrbenutzerbetrieb, eine Zugriffskontrolle und andere Datenbanksicherheitsmechanismen. *IBM* stellt zudem eine Vielzahl von Werkzeugen für die Definition, Anfrage und Darstellung von *DB2*-Datenbanken bereit.

3.3 Bachelorarbeit: *Extraction and Storage of Web Structures*

Die grundlegenden Daten für diese Arbeit wurden der im Rahmen der Bachelorarbeit *Extraction and Storage of Web Structures* von Maksims Abalenkovs [Aba06] entstandenen Datenbank entnommen. Die Datenbank enthält Daten, die das WWW repräsentieren. Angefangen mit dem Start-Host *http://dbs.cs.uni-duesseldorf.de* wurde das Web mittels eines Crawlers bis zu einem bestimmten Grad durchsucht und dann in der Datenbank gespeichert. Das Verfahren kann jederzeit wiederholt und die Datenbank erweitert werden. Es folgen eine kurze Erläuterung der Datenbankstruktur und Beispiele für die Basisdaten.

3.3.1 Datenbankmodell

Abbildung 7 zeigt das *Entity-Relationship Modell* der Basisdatenbank. Ausprägungen vom Entity *Host* besitzen eine eindeutige *HID* zur Identifikation und einen Namen *HNAME*. Jedem Host gehört eine Menge von Ressourcen an, die Ausprägungen vom Entity Resource sind. Alle Ressourcen haben eine eindeutige *RID*, einen *URL*, einen MIME-Typ, gespeichert in *RType*, und die *HID* des Hosts, dem die Ressource angehört. Darüber hinaus verfügt der Entity-Typ Resource zwar noch über mehr Attribute in der Datenbank, allerdings sind diese für den Entwurf und die Implementierung von *wwwVis* nicht relevant. Ressource *a* linkt im WWW auf eine Ressource *b*. Dies wird mit der Relation *links* modelliert. Sie enthält die *RID* von *a* als *FRID* und die *RID* von *b* als *RID*. Das *Entity-Relationship Modell* wurde durch die drei Tabellen *Hosts*, *Resources* und *Links* und den dazugehörigen Attributen in einer *DB2*-Datenbank umgesetzt. Mittels *JDBC*-Schnittstelle kann daher auf die Basisdatenbank und deren Tabellen, Spalten und Tupel zugegriffen werden, um den Datenraum zu visualisieren.

3.3.2 Basisdaten

Die Basisdatenbank, die im Rahmen der Bachelorarbeit *Extraction and Storage of Web Structures* mit ihren drei Tabellen aufgebaut wurde, beinhaltet derzeit 1573 Tupel in *Hosts*, 150217 Tupel in *Resources* und 510519 Tupel in *Links*. D.h. also, dass 150217 Ressourcen auf 1573 verschiedenen Hosts liegen und 510519 Links zwischen den Ressourcen vorhanden sind. Dies ist zwar nur ein sehr kleiner Ausschnitt aus dem WWW, die Datenbank lässt sich aber jederzeit erweitern, um somit die Basisdaten für *wwwVis* zu vergrößern. Beispieleinträge der Datenbank zeigen die Tabellen 4, 5 und 6.

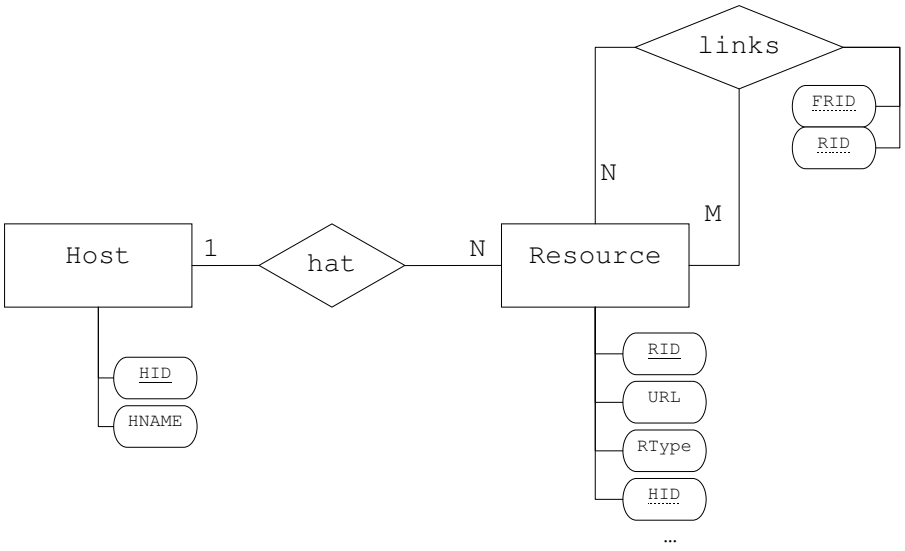

Abbildung 7: *Entity-Relationship Modell* der Basisdatenbank (nach [Aba06])

HID	HNAME
1	http://dbs.cs.uni-duesseldorf.de
2	http://cs.uni-duesseldorf.de
3	http://www.uni-duesseldorf.de

Tabelle 4: Beispieleinträge der Tabelle *Hosts* aus der Basisdatenbank

RID	URL	RTYPE	HID
1	http://dbs.cs.uni-duesseldorf.de/lehre/docs	text/html	1
2	http://dbs.cs.uni-duesseldorf.de/lehre/docs/java...		
	.../javabuch/html/cover.html	text/html	1
3	http://www.cs.uni-duesseldorf.de/index.html	text/html	2
4	http://www.cs.uni-duesseldorf.de/kontakt/hhu.gif	image/gif	2

Tabelle 5: Beispieleinträge der Tabelle *Resources* aus der Basisdatenbank

FRID	RID
1	2
1	3
3	4

Tabelle 6: Beispieleinträge der Tabelle *Links* aus der Basisdatenbank

3.4 Die Qt Entwicklungsumgebung

Qt ist eine C++ Entwicklungsumgebung der norwegischen Firma Trolltech zur Programmierung leistungsstarker, plattformübergreifender Software, insbesondere zur Entwicklung von *Graphical User Interfaces* [QTW07]. *Qt* wird mittlerweile weltweit von vielen Programmierern, auch bekannter Firmen wie *Adobe* oder *IBM*, genutzt und ist derzeit in der Version 4.3.1 erhältlich. Für kommerzielle Projekte ist es käuflich zu erwerben, ansonsten aber auch unter der *General Public License (GPL)* verfügbar. Ein großer Vorteil der GUI-Implementierung mit *Qt* ist, dass *Qt* eine umfangreiche Klassenbibliothek mit graphischen Komponenten zur Interaktion, den so genannten *Widgets*, besitzt. Enthalten sind einfache Standard-*Widgets* wie z.B. Buttons, Textfelder oder Bilder, aber auch *Table-Widgets* zur Darstellung von Tabellen oder *Tree-Widgets*, die Baumstrukturen in das *GUI* einfügen. Ein vom Nutzer als *Container* selbst definiertes *Widget* kann mehrere Standard-*Widgets* enthalten, so dass komplexe eigene Layouts möglich sind. Eine Kommunikation zwischen den *Widgets* erfolgt über die *signals and slots*-Funktion von *Qt*. Ein Widget, welches gerade aktiv ist, sendet z.B. bei bestimmten Aktionen ein Signal. Alle vorher implementierten Slots, die mit dem Signal verbunden sind, werden daraufhin ausgeführt. Ein Beispiel für eine *signals and slots*-Funktion ist ein OK-Button, der gedrückt wurde. Beim Drücken des Buttons wird ein Signal gesendet. Infolgedessen werden alle verbundenen Slots, beispielsweise eine Methode zum Schliessen des Fensters, ausgeführt. *Qt* unterstützt alle gängigen *GUI*-Funktionen wie Menüs, Kontextmenüs, *Drag and Drop* und Toolbars.
Trolltech bietet zudem Programmierern einige Tools an. Der *QtDesigner* erleichtert das Erstellen von komplexen Widget-Containern, der *QtAssistant* bietet eine ausführliche Hilfe und der *QtLinguist* vereinfacht die Übersetzung von Anwendungen, um ein *GUI* mehrsprachig implementieren zu können. Alle Standard-Widgets, Tools und weitere *Qt*-Funktionen findet man in der *Qt Reference Dokumentation* [QTD07] wieder.

3.4.1 Qt Designer

Der *QtDesigner* ist ein Tool, welches das Erstellen von komplexen Layouts, die die verschiedensten *Widgets* als Elemente enthalten, erleichtert [QTW07] [QJW07]. Mit Hilfe des integrierten Editors ist eine problemlose Positionierung und Skalierung der *Widgets* per *Drag and Drop* gewährleistet. Eine automatische Ausrichtung der Elemente im Layout ist durch den *QtDesigner* ebenso möglich wie das Definieren von *signals and slots*. Darüber hinaus kann zu Beginn, wenn gewünscht, ein vordefiniertes Layout ausgewählt werden. So ist z.B. ein *MainWindow* mit Menüleiste, Toolbar und Statusleiste direkt im *QtDesigner* inbegriffen.

Die erstellten Layouts bzw. *User Interfaces* werden im *.ui Format* gesichert und können mit Hilfe des *Java User Interface Compiler (JUIC)* in Java-Programmcode, der anschließend editierbar ist, konvertiert werden.

3.4.2 Qt Jambi

Qt Jambi ist die Java Version der C++ Entwicklungsumgebung [QJW07]. *Jambi* macht die Implementierung eines *GUI* mit Java möglich und bietet alle Vorzüge der normalen Qt-Version. Im Unterschied zu den graphischen Bibliotheken der *Java Foundation Classes* besitzt *Qt Jambi* die Vorteile einer intuitiven und mächtigen Schnittstelle zur Anwendungsprogrammierung (*application programming interface*, kurz: *API*), sowie den *signals and slots*-Mechanismus. Der *Qt Jambi*-Programmiercode ist zudem kürzer, präziser und oft leichter zu verstehen. Auch einen einfachen, verständlichen Editor, wie den *QtDesigner* mit einem vordefinierten *MainWindow*, sucht man bei *Swing* und den anderen *JFC* vergeblich.
Qt Jambi ist, ähnlich wie die C++ Version und wie die Programmiersprache Java, plattformunabhängig. Mit *Qt Jambi* programmierte *GUI* behalten auf Windows, Linux oder Mac OS Betriebssystemen das gewohnte Aussehen des verwendeten Systems.
Informationen, Dokumentationen, Tools und alles Weitere zu *Qt Jambi* findet man in der *Qt Jambi Reference Documentation* [QJD07], sowie die Klassen des umfangreichen *API* in der *Qt Jambi API Javadoc Documentation* wieder [QJA07].
Die Java-Version von Qt wurde für die Implementierung von *wwwVis* verwendet. So können die Vorteile von Qt und Java für die Anwendung optimal ausgenutzt werden.

3.4.3 Visualisierungen in Qt Jambi

Qt Jambi unterstützt sowohl die Verwendung von 2D- als auch von 3D-Graphiken [QJW07]. Mittels hochwertigem *Rendering* garantiert Qt Jambi eine qualitative Abbildung von Visuellen Abstraktionen zu Bildern. Die Visualisierung geschieht hauptsächlich über die Klassen `QPainter` und `QPaintDevice`. Die Klasse `QPainter` führt Graphikoperationen, wie z.B. das Zeichnen von geometrischen Figuren in verschiedenen Farben und Größen, aus. Die Klasse `QPaintDevice` dagegen ist eine Abstraktion des zweidimensionalen Raumes, auf der die Graphikoperationen ausgeführt werden können. Insgesamt gesehen ist auch die Visualisierung mit Qt leichter und intuitiver als mit anderen Werkzeugen.

4 Entwicklung und Implementierung von *wwwVis*

wwwVis ist eine Anwendung mit integrierter graphischer Benutzeroberfläche, die die im Rahmen der Bachelorarbeit *Extraction and Storage of Web Structures* gewonnenen Daten graphisch darstellt und somit eine Visualisierung der Webstruktur in Ausschnitten liefert. Die Anwendung wurde unter Berücksichtigung der Anforderungen an eine optimale Daten- bzw. Informationsvisualisierung und mit Unterstützung der bereits erwähnten visuellen Möglichkeiten implementiert. Insbesondere wurden das *Information Visualization Data-State-Reference Model* und das *Mantra der Informationsvisualisierung* bei der Entwicklung von *wwwVis* beachtet. Dieses Kapitel stellt im weiteren Verlauf das Design des *GUI*, den Aufbau und die Funktionalität von *wwwVis*, sowie einige wichtige Operationen auf den Daten- bzw. Darstellungsraum vor. Abschließend werden exemplarisch einige Beispiele für die Visualisierung und Analyse der Webstruktur erläutert.

4.1 Design der graphischen Benutzeroberfläche

Die graphische Benutzeroberfläche (Abbildung 8) von *wwwVis* besteht aus vier verschiedenen Widgets, die mit Hilfe des *QtDesigners* gemeinsam in ein von *Qt* bereitgestelltes *QMainWindow* eingefügt wurden. Neben der im *QMainWindow* enthaltenen Menüleiste (1), umfasst das *GUI* ein Widget zur Anzeige von Verzeichnisbäumen (engl. *TreeWidget*) (2), ein Graphik-Widget (3), ein Nachrichtenfenster (4) und eine Kommandozeile (5). Die Größe der graphischen Benutzeroberfläche kann jederzeit während der Laufzeit des Programms verändert werden. Dabei behalten die Widgets stets ihre Ausrichtung und Positionierung bei. Jedes Widget hat zudem sein eigenes Kontextmenü, dass per Rechtsklick abgerufen werden kann.

(1) Menüleiste:
Über die Menüleiste können alle Funktionen und Operationen von *wwwVis* ausgeführt werden.
(2) TreeWidget:
Das *TreeWidget* dient zur Auflistung von Daten aus der Basisdatenbank. In erster Linie werden hier Webressourcen bzw. je nach vorher angewandter Funktion der *URL* von Hosts oder Dateien, verzeichnet. Nach einem Doppelklick auf ein Element des *TreeWidgets* öffnet sich der Webbrowser mit entsprechendem *URL* und zeigt das angeklickte Element. Ferner enthält das Kontextmenü des *TreeWidgets* eine Vielzahl der Funktionen, die auch über die Menüleiste erreichbar sind.
(3) Graphik-Widget:
Im Graphik-Widget des *GUI* werden alle Visualisierungen veranschaulicht. Eine Vollbildansicht des Visualisierungsfensters kann über das Kontextmenü des Widgets angezeigt werden. Ebenso ist es per Rechtsklick möglich die Zoomfunktion zu benutzen oder Namen ein- und auszublenden.
(4) Nachrichtenfenster:
Fehler- und Erfolgsmeldungen von Funktionen der Anwendung werden im Nachrichtenfenster ausgegeben. Fehlermeldungen werden dabei deutlich sichtbar in roter Farbe hervorgehoben. Es erfasst außerdem alle Aktionen, die bisher bereits ausgeführt wurden. So ist es für einen Benutzer möglich den bisherigen Ablauf seiner Arbeit mit *wwwVis*

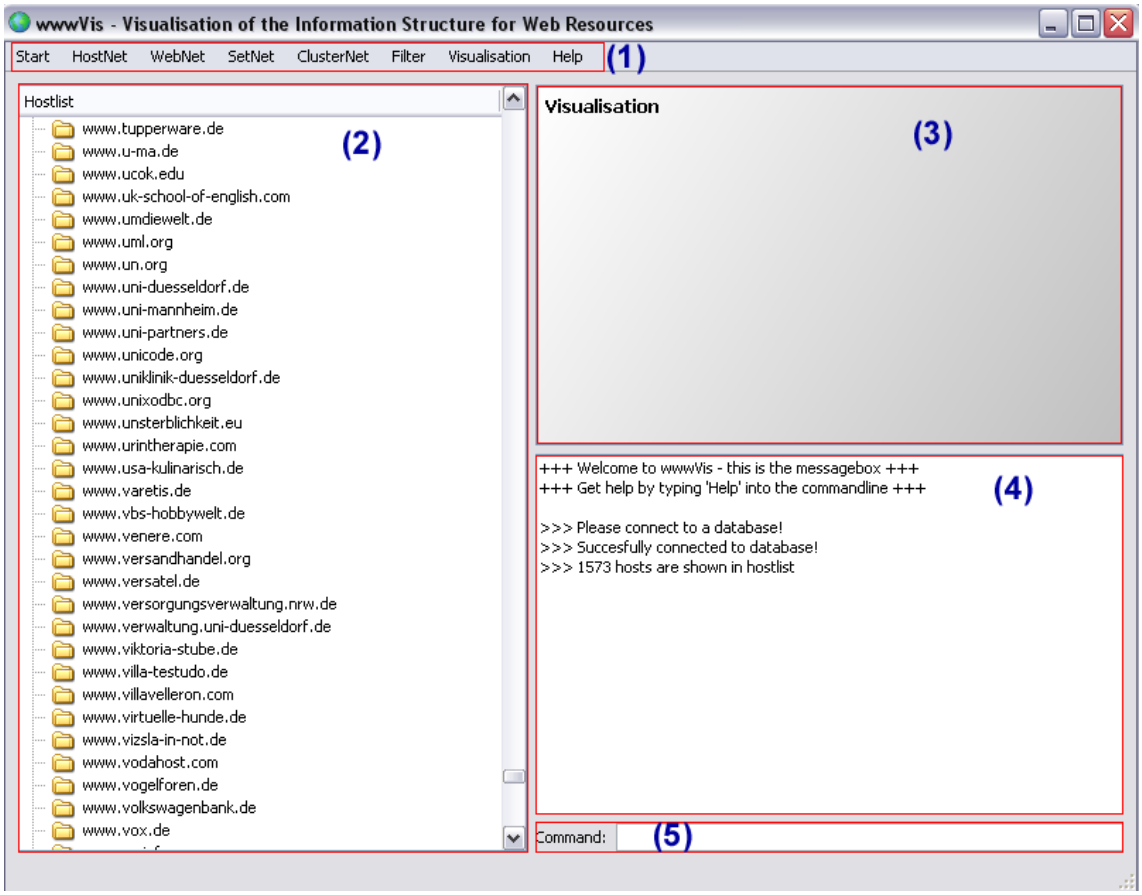

Abbildung 8: Die graphische Benutzeroberfläche von *wwwVis*

nachzuschlagen.
(5) Kommandozeile:
Über die Kommandozeile können ebenfalls alle Funktionen von *wwwVis* aufgerufen werden. Durch Eingeben von kurzen, präzisen Kommandos ist ein schnelleres Arbeiten mit der Anwendung möglich. Einige Benutzer bevorzugen diese Art von Steuerung eines *GUI*, da es besonders bei ausreichender Kenntnis der verfügbaren Kommandos den Umgang mit der graphischen Benutzeroberfläche erleichtert. Im Untermenü *Help –> Commands* ist eine Liste aller in der Kommandozeile nutzbaren Kommandos vorhanden.

4.2 Funktionalität von *wwwVis*

Die zentrale Aufgabe von *wwwVis* ist die Visualisierung des WWW. Zur Analyse und zum Verständnis der Webstruktur sind darüber hinaus weitere Funktionen integriert. Zur ersten Orientierung und Übersicht über die Webstruktur dient in der Regel die Funktion *WebNet*.
Ein Anwender von *wwwVis* muss nicht zwangsläufig Kenntnisse über die verwendeten Algorithmen und Implementierungen des Programms haben. Es ist ausreichend, wenn der Nutzer Grundkenntnisse über das WWW besitzt. Erklärungen zur Implementierung

von *wwwVis* finden sich in den nächsten Kapiteln *Aufbau und Struktur der Implementierung*, sowie *Operationen im Daten- bzw. Darstellungsraum* wieder. Folgende grundlegende Funktionalitäten sind Bestandteil von *wwwVis* und können über die Menüleiste ausgewählt werden:

4.2.1 Verbindung zur Datenbank

Nach dem Start der Anwendung ist es zunächst nötig, eine Verbindung zur Basisdatenbank herzustellen. *Start –> Connect to Database* stellt eine Verbindung zu einer Datenbank her. Es öffnet sich ein Dialogfenster, in dem die Verbindungsdetails eingetragen werden können. Die Details werden auf Wunsch in der Datei `db.conf` auf der Festplatte gespeichert. *Start –> Disconnect from Database* trennt wiederum die Verbindung zur Datenbank.

4.2.2 HostNet

Das *HostNet* ist eine Funktion zur Darstellung der Ressourcen eines Hosts und zur Visualisierung der Links zwischen diesen Ressourcen. So können beispielsweise alle Dateien, die auf dem Host *http://www.uni-duesseldorf.de* abgelegt sind, im *TreeWidget* aufgelistet werden (*HostNet –> Choose Host to show all files of Host*). Als Übersicht über alle verfügbaren Host der Basisdatenbank kann zuvor eine Hostliste im *TreeWidget* angezeigt werden (*HostNet –> Show Hostlist*). Die Visualisierung der Links, zwischen Ressourcen des Hosts untereinander, geschieht über *HostNet –> Show HostNet* und liefert eine Graphendarstellung mit Knoten und Kanten im *Graphik-Widget*.

4.2.3 WebNet

Die Hauptanwendung von *wwwVis* ist das *WebNet*. Damit lassen sich die Links zwischen Hosts des WWW anzeigen. Angefangen mit einem Startpunkt, der über *WebNet –> Choose Host as Startpoint* zu bestimmen ist, werden alle Links und erreichbaren Host bis zu einer bestimmten Linktiefe ermittelt (*WebNet –> Show WebNet*). Der daraus resultierende Webgraph wird anschließend im Visualisierungsfenster gezeigt.

4.2.4 SetNet

Das *SetNet* stellt ähnlich wie das *WebNet* die Links zwischen Hosts aus der Basisdatenbank bis zu einer bestimmten Linktiefe dar, jedoch reduziert auf Links zwischen Hosts einer vorher ausgesuchten Menge an Hosts. *SetNet –> Choose Set of Hosts* definiert diese Menge, das Set, der berücksichtigten Hosts, indem sich ein neues Dialogfenster öffnet. Aus der Liste aller Hosts können nun die Gewünschten selektiert werden. Der zuerst gewählte Host übernimmt die Rolle des Startpunktes. Die Darstellung des resultierenden Graphen im Visualisierungsfenster erfolgt daraufhin über *SetNet –> Show SetNet*.

4.2.5 ClusterNet

Die Funktion *ClusterNet* erlaubt die Selektion von mehreren Startpunkten, wiederum über ein Dialogfenster (*ClusterNet –> Choose Set of Hosts as Startpoints*). Folgend werden die Links zwischen Hosts aus der Basisdatenbank, beginnend mit den verschiedenen Startpunkten, bis zu einer bestimmten Linktiefe ermittelt. Um jeden Startpunkt herum entstehen so Cluster. Der Webgraph mit den verschiedenen Clustern der unterschiedlichen Startpunkte wird über (*ClusterNet –> Show ClusterNet*) visualisiert. Die maximale Anzahl von Startpunkten beträgt allerdings nur 5, da sonst die Darstellung unübersichtlich wird und diese nicht mehr in angemessener Zeit geschehen kann.

4.2.6 Filter

Eine Restriktion der Menge an Webressourcen, die in den Visualisierungsfunktionen *HostNet*, *WebNet*, *SetNet* und *ClusterNet* berücksichtigt werden, ist durch Anwendung von Filteroperationen möglich. In *wwwVis* gibt es zwei Arten von Filteroptionen: die Dateitypenfilter (*Filter –> Filetypes*) und die Domainfilter (*Filter –> Domains*). Die Dateitypenfilter beschränken die Anzeige von Dateien eines Hosts und sind daher nur auf das *HostNet* anwendbar. So können z.B. nur PDF-Dokumente oder nur HTML-Dateien aufgelistet werden, die auf einem Host abgelegt sind. Domainfilter verringern die Menge an betrachteten Hosts bei Realisierung von *WebNet*, *SetNet* oder *ClusterNet*. Deshalb werden beispielsweise bei Einstellung des Domainfilters auf **.de* nur Hosts verwendet und angezeigt, dessen *URL* auf *.de* endet. Hier würden also nur Webressourcen auf Hosts aus Deutschland berücksichtigt. Filtereinstellungen sind optional. Sollte sich jedoch die Größe der Basisdatenbank enorm erhöhen, so sind Filteroptionen von Nöten, um eine Visualisierung der riesigen Datenmenge noch zu ermöglichen. In diesem Fall erscheint eine entsprechende Meldung im Nachrichtenfenster der Anwendung. Ein Zurücksetzen oder Ausschalten der Filter ist jederzeit durch *Filter –> Filetypes –> Show all files* bzw. *Filter –> Domains –> Show all* möglich.

4.2.7 Visualisierung

Die Visualisierung ist Teil aller Funktionen von *wwwVis*. *HostNet*, *WebNet*, *SetNet* und *ClusterNet* stellen jeweils einen Ausschnitt des Webs dar und liefern dem Nutzer mehr Informationen (*Informationeller Mehrwert*) als z.B. die Listendarstellung im *TreeWidget*, als die Daten aus der Basisdatenbank oder Darstellungen außerhalb dieser Anwendung in Informationssystemen ohne Visualisierung.
Webressourcen, Links und deren Beziehungen zueinander werden mit der Visualisierung in *wwwVis* sichtbar. Webressourcen werden in *wwwVis* als Knoten und Links als Kanten dargestellt, so dass im Graphik-Widget ein gerichteter Graph entsteht. Eine Kante von einem Knoten auf sich selbst wird in der Anwendung nicht dargestellt, da dies mit Hilfe des *HostNets* im Detail realisiert wird. *HostNet* veranschaulicht die Dateien bzw. den Inhalt eines Hosts und dessen Links untereinander.
Vor einer Visualisierung ist es notwendig die Linktiefe (engl. *depth of links*) einzustellen. Die Linktiefe gibt an, wie weit Links ausgehend von einem Startpunkt verfolgt werden

sollen, also wie viele Links ein Host, der visualisiert wird, maximal von dem Startpunkt entfernt sein darf. Bei einer Linktiefe von 2, angefangen mit Startpunkt *http://www.uni-duesseldorf.de*, werden z.B. alle Webressourcen visualisiert, die über einen Pfad von maximal 2 Links von *http://www.uni-duesseldorf.de* aus erreichbar sind. Die Linktiefe wird von *wwwVis* automatisch abgefragt, ist jedoch auch nachträglich über *Visualisation –> Set Depth of Links* veränderbar. Abhängig von der jeweiligen Linktiefe haben Knoten in *wwwVis* verschiedene Farben und unterschiedliche Größen. Der Startknoten wird in rot, Knoten der Linktiefe 1 in grün, der Linktiefe 2 in magenta, der Linktiefe 3 in gelb und der Linktiefe 4 in hellblau dargestellt. Die maximal einstellbare Linktiefe beträgt 4, da ansonsten die Laufzeit des Programms eine akzeptable Zeit überschreitet. Kanten besitzen auch unterschiedliche Farben gemäß der Anzahl der Links zwischen zwei Webressourcen. Existiert nur ein Link, so ist die Kante schwarz. Wenn mehr als ein Link, aber weniger als 10 Links zwischen zwei Ressourcen vorhanden sind, so wird die Kante in blau eingefärbt. Bestehen sogar mehr als 10 Links, so ist die Kante rot. Rote Kanten symbolisieren also eine relativ hohe Linkdichte zwischen zwei Webressourcen.
Die Visualisierungen in *wwwVis* orientieren sich an den bereits erwähnten Anforderungen an eine optimale Visualisierung. Die Implementierung der Visualisierung orientiert sich insbesondere am *Information Visualization Data-State-Reference Model*, die Darstellung hingegen speziell am *Mantra der Informationsvisualisierung*. Die zentrale Idee des Mantras von Shneiderman: „Overview first, zoom and filter, then details-on-demand [Shn96]“ wurde in *wwwVis* umgesetzt. Zunächst ist es möglich sich mit einer Visualisierung einen Überblick (*Overview*) zu verschaffen, um dann, wenn gewünscht, die Darstellung mit Knoten und Kanten heran- *(Visualisation –> Zoom In)* oder herauszuzoomen *(Visualisation –> Zoom out)*, sowie verschiedene Filter anzuwenden. Ferner ist die Einblendung von Details möglich, indem ein Host für die Funktion *HostNet* ausgesucht wird, um somit den Inhalt dieses Hosts aufzulisten. Die Veranschaulichung von Webressourcen im Webbrowser kann ebenfalls als *details-on-demand*-Methode betrachtet werden, da sich so Details zu jeder Ressource recherchieren lassen.
Das Problem der riesigen Datenmenge als Grundlage für die Visualisierung, wird mit Techniken der *Focus-and-Context Visualisierung* und mit den Filteroptionen umgangen. Die Basisdaten, und damit das WWW, können niemals vollständig im Graphik-Widget veranschaulicht werden. In *wwwVis* wurden sowohl *hinweisende* als auch *dimensionale FC-Techniken* eingesetzt, um einen Knoten im Fokus hervorzuheben, ohne dabei den Gesamtkontext zu missachten. In der Visualisierung kann ein Knoten durch Anklicken fokussiert werden. Der Knoten wird dann vergrößert und farblich anders dargestellt. Dies ist eine *hinweisende FC-Technik*. Der Graph im Graphik-Widget richtet sich bei einer Visualisierung zunächst selbst aus, um beispielsweise Knoten gleicher Linktiefe ähnlich anzuordnen. Ein Verschieben von Knoten und Umpositionieren des gesamten Graphs ist darüber hinaus eine weitere *hinweisende FC-Methode* in *wwwVis* zur Fokussierung verschiedenster Knoten. Das bereits angesprochene Zooming ist eine *dimensionale FC-Methode*. *Räumliche Focus-and-Context*-Techniken, wie z.B. der *Fish-Eye-View-Effekt*, hingegen wurden bei der Visualisierung nicht verwendet. Eine unscharfe Darstellung des Gesamtkontextes, hier die Links zu anderen Webressourcen, würde bei der Visualisierung des WWW wenig Sinn machen, da gerade der Gesamtkontext interessant ist. Eine Ausblendung von Teilen des Gesamtkontextes kann ohnehin durch *Zooming* oder die Filteroptionen erzielt werden.
Neben den *Focus-and-Context*-Methoden sind noch einige andere Techniken, die vorher

bereits erwähnt wurden, bei der Entwicklung von *wwwVis* implementiert worden. So wird beispielsweise die Visualisierung der Strukturierung des Informationsbestandes, hier die Hierarchien zwischen den Webressourcen, sowohl explizit durch die Kanten, als auch implizit durch Anordnung realisiert. Die Anordnung verläuft radial um den Startpunkt herum. Die Visualisierung ist zudem zwar auf einer zweidimensionalen Fläche abgebildet, durch Wölbung der Knoten und mittels Schatten wird jedoch eine dreidimensionale Abbildung simuliert. Intuitive Interaktionstechniken sind mit der möglichen Bewegung des Graphen und den Filteroptionen in der Anwendung integriert. Unterschiedliche *Mapping*-Methoden wurden nicht verwendet, da die Abbildung der Daten des WWW auf Kanten und Knoten zum einen sinnvoll und zum anderen üblich in der Informatik ist.
Zusammenfassend bleibt festzuhalten, dass *wwwVis* versucht, eine optimale Visualisierung des WWW zu erreichen. Da die Datenmenge aber potenziell unendlich ist und mitunter komplizierte Strukturen auftreten können, ist eine ideale Visualisierung leider nicht immer möglich. Das Ausnutzen aller Funktionen von *wwwVis* kann aber die Visualisierung optimieren. Mit Hilfe der Veranschaulichung der einzelnen Funktionen des Programms ist es zudem möglich, sowohl *Hubs* und *Authorities*, als auch themenverwandte Websites zu erkennen oder einfach nur die Existenz eines Pfades von einer Webressource zur anderen zu überprüfen. Die Untersuchung des *Small-World-Phänomens* ist mit Hilfe von *wwwVis* nur begrenzt möglich, da keine Linktiefen größer als 4 eingesetzt werden können.

4.3 Aufbau und Struktur der Implementierung

Die Implementierung der Anwendung wurde mit der Programmiersprache Java, sowie der *JDBC* und *Qt Jambi* durchgeführt. Das Programm ist, wie Abbildung 9 zeigt, in 8 Pakete aufgeteilt. Jedes Paket enthält verschiedenen Klassen, die gemeinsam eine bestimmte Aufgabe erfüllen:

wwwVis enthält die Klassen `MainWindow`, `Ui_MainWindow` und `MainWindowAction`. Die Klasse `MainWindow` stellt das Menüfenster mit Menüleiste und Statusleiste zur Verfügung. `MainWindowAction` enthält zudem alle Aktionen, die über die Menüleiste und über die Kontextmenüs ausgeführt werden können. Die vier verschiedenen Widgets von `wwwVis` sind in der Klasse `Ui_MainWindow`, die, wie alle mit *Ui* beginnenden Klassen, mit dem *QtDesigner* erstellt wurde, implementiert.

wwwVis.ui ist ein Paket, dass alle Dialogfenster zusammenfasst. `Ui_DepthOfLinks` ist beispielsweise das Dialogfenster zur Abfrage der Linktiefe oder `Ui_ProgressBar` der Balken, der erscheint, wenn die Anwendung gerade eine Funktion ausführt. `Ui_MyTreeWidget` ist ein eigens definiertes *TreeWidget*, welches das Standard-*TreeWidget* von Qt mit einem Kontextmenü erweitert.

wwwVis.databaseconnection umfasst mit `Ui_DatabaseConnection` und `DatabaseConnection` die Klassen, welche eine Verbindung zur Basisdatenbank herstellen. `Ui_DatabaseConnection` beinhaltet das Dialogfenster zur Abfrage der Datenbankdetailinformationen, `DatabaseConnection` baut mit

`createDatabaseConnection()` eine Verbindung zur Datenbank auf und stellt zudem noch mit `quitDatabaseConnection()` eine Methode zur Trennung der Datenbankverbindung zur Verfügung.

wwwVis.graph (Abbildung 11) fasst mit den Klassen `Hostlist`, `Net`, `HostNode` und `Link` die Klassen zusammen, die, entsprechend dem *Information Visualization Data-State-Reference Model*, für das *Filtering* von Daten zu *Analytischen Abstraktionen* und das *Mapping* von *Analytischen Abstraktionen* zu *Visuellen Abstraktionen* zuständig sind. D.h. die Klassen überführen alle Daten aus der Datenbank in einen aus Kanten und Knoten bestehenden Gesamtgraphen und danach den Gesamtgraphen bis zu einer bestimmten Linktiefe mittels Breitensuche in eine hierarchische Struktur, die anschließend visualisiert werden kann.
Die Klasse `Hostlist` ist zunächst, unter Berücksichtigung der Filteroptionen, für die Ausgabe aller Webressourcen der Basisdatenbank zuständig. Über die Methoden `createHostlist()` bzw. `createHostfilelistFromHost()` werden die Hosts bzw. Dateien eines Host im TreeWidget ausgegeben. Die Klasse `Net` dient der Extrahierung von Daten aus der Datenbank. Die Webressourcen und Links aus der Datenbank werden in einer `HashMap` gespeichert. Die `HashMap` bildet somit den Gesamtgraphen aus Kanten und Knoten. Für die Speicherung des Graphen wurde eine `HashMap` verwendet, da sie eine optimale Datenstruktur zur Speicherung einer großen Menge von ungeordneten Elementen ist. Eine `n x n -Adjazenzmatrix` mit n Knoten, abgebildet auf die Zeilen und Spalten, wäre in diesem Fall keine gute Lösung. Die Matrix würde sehr viele Lücken aufweisen, wenn ein Eintrag der Matrix für eine Kante steht, da viele Kanten zwischen zwei Knoten im Webgraphen nicht existieren. Ein Element der `HashMap` besteht aus dem *URL* der Webressource und einer Instanz der Klasse `HostNode`. Die Klasse `HostNode` repräsentiert eine Webressource und dessen ausgehende Links. Eine Ausprägung der Klasse hat einen Namen, meist der *URL*, und einen `Vector` aus Instanzen der Klasse `Link`, der alle ausgehenden Links enthält. Ein `Link` besitzt die *URL* der Zielwebressource und die Anzahl der Links, die dorthin führen.
Darüber hinaus enthält die Klasse `Net` die Methode `breadthFirstSearch()`, die mittels Breitensuche das *Mapping* von Graph zu Hierarchie durchführt. Der Graph wird mit einer Breitensuche bis zu einer bestimmten Linktiefe durchlaufen. Alle erreichbaren Knoten werden mit ihrer `HostNode`-Instanz in einer `Queue` gespeichert. Die `Queue` ist eine geordnete Liste, beinhaltet die hierarchische Struktur und kann nun visualisiert werden. Die Implementierung der Breitensuche wird später im Kapitel 4.4.2 erläutert.

wwwVis.visualisation (Abbildung 10) beinhaltet die Klassen zur Visualisierung der Graphenstruktur. Die Klasse `Graph` enthält den Startpunkt und alle Knoten des Graphen in einem `Vector` als Klassenvariablen. Die Methoden `drawGraph()`, `drawClusters()` und `drawBackground()` ermöglichen das Zeichnen des Graphen bzw. der Ansicht. `scaleView()` steuert die Zoomfunktion im Graphik-Widget. Des Weiteren werden in den Methoden `keyPressEvent()` und `wheelEvent()` Tastendrücke und Mausradbewegungen zur Skalierung der Ansicht behandelt. Eine Instanz der Klasse `Node` repräsentiert einen Knoten in der Visualisierung. Der zugehörige Graph, der Name, die Linktiefe und eine Liste der ausgehenden Kanten wird in Variablen der Klasse gesichert. `addEdge()` fügt der Kantenliste eine Kante hinzu, `paint()` zeichnet

den Knoten und `shape()` fügt den Schatten des Knotens in die Visualisierung ein. Visualisierte Kanten sind Ausprägungen der Klasse `Edge` und besitzen neben dem Start- und Endknoten der Kante auch die Anzahl der existierenden Links als Klassenvariable. `paint()` sorgt auch hier dafür, dass der Knoten im Graphik-Widget gezeichnet wird. Angaben für Größen, Farben und weitere Layouteigenschaften, sowie die genauen Punkte, an denen die Knoten und Kanten visualisiert werden, befinden sich ebenfalls in den Klassen `Node` und `Edge`.

wwwVis.messagebox besteht nur aus der Klasse `Messagebox`. Sie steuert die Ausgabe der Informationen im Nachrichtenfenster. `showMessageInMessagebox()` sendet eine normale Meldung in schwarzer Farbe an das Nachrichtenfenster. `showErrorInMessagebox()` hingegen schickt eine Fehlermeldung. Die Fehlermeldung erscheint im Nachrichtenfenster gut sichtbar für den Benutzer in rot.

wwwVis.command besitzt ebenfalls nur eine Klasse. `Command` vergleicht die Eingaben in der Kommandozeile mit den implementierten Kommandos und führt daraufhin die entsprechenden Aktionen aus. Sollte die Eingabe in der Kommandozeile nicht bekannt sein, wird eine Fehlermeldung im Nachrichtenfenster ausgegeben.

wwwVis.filter ist für Regulierung Filtereinstellungen der Anwendung zuständig. `getFiletypesFilter()` und `getDomainsFilter()` liefern die aktuellen Filteroptionen zurück, während `setAllFiletypesUnchecked()`, `setShowAllFiletypesUnchecked()`, `setAllDomainsUnchecked()` und `setShowAllDomainsUnchecked()` die Optionen im Menü *Filter* aufeinander abstimmen. Wenn beispielsweise *Show All* in der Menüleiste unter *Filter* ausgewählt wurde, dann müssen alle anderen Optionen dort deselektiert werden.

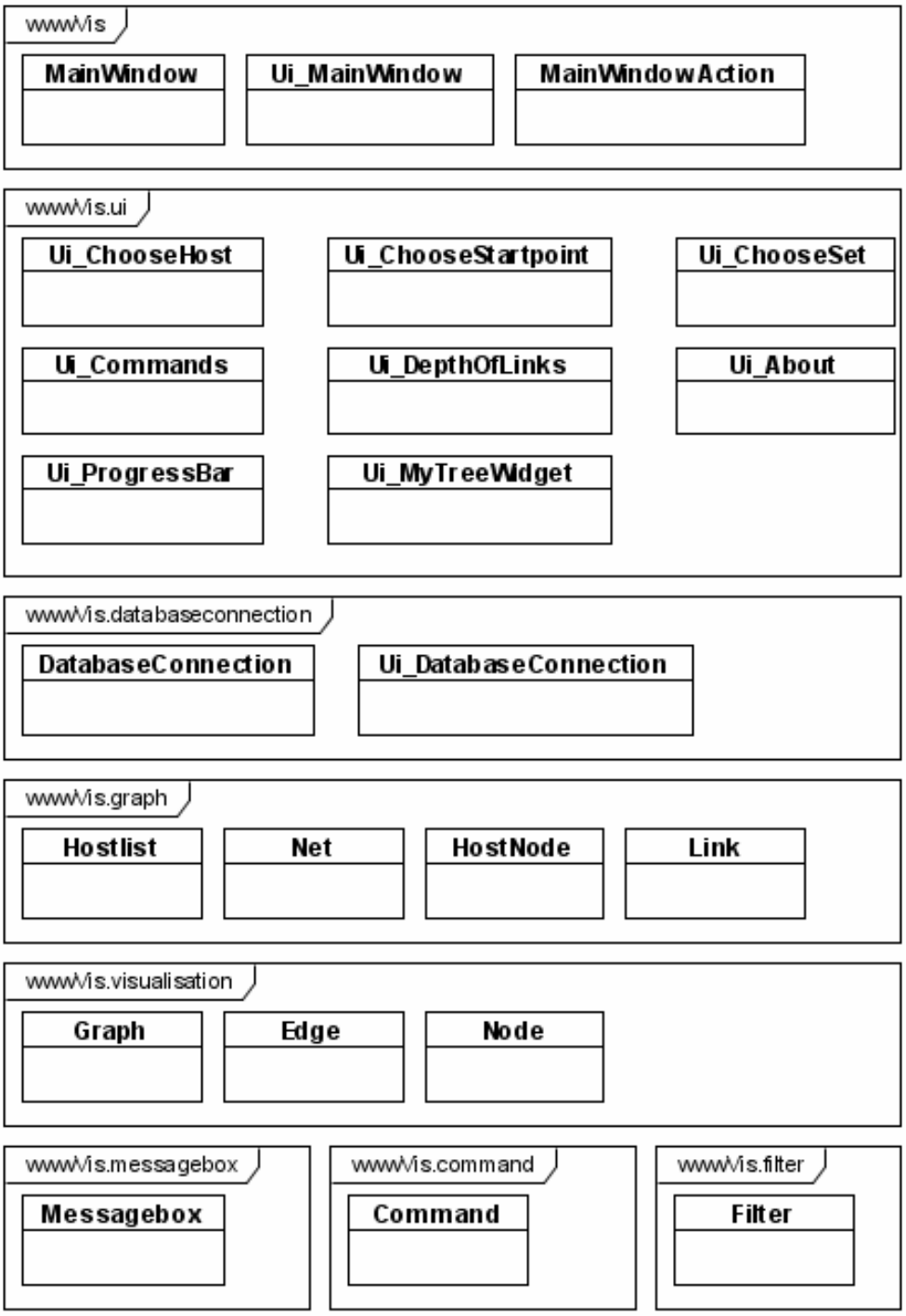

Abbildung 9: Paketdiagramm von *wwwVis*

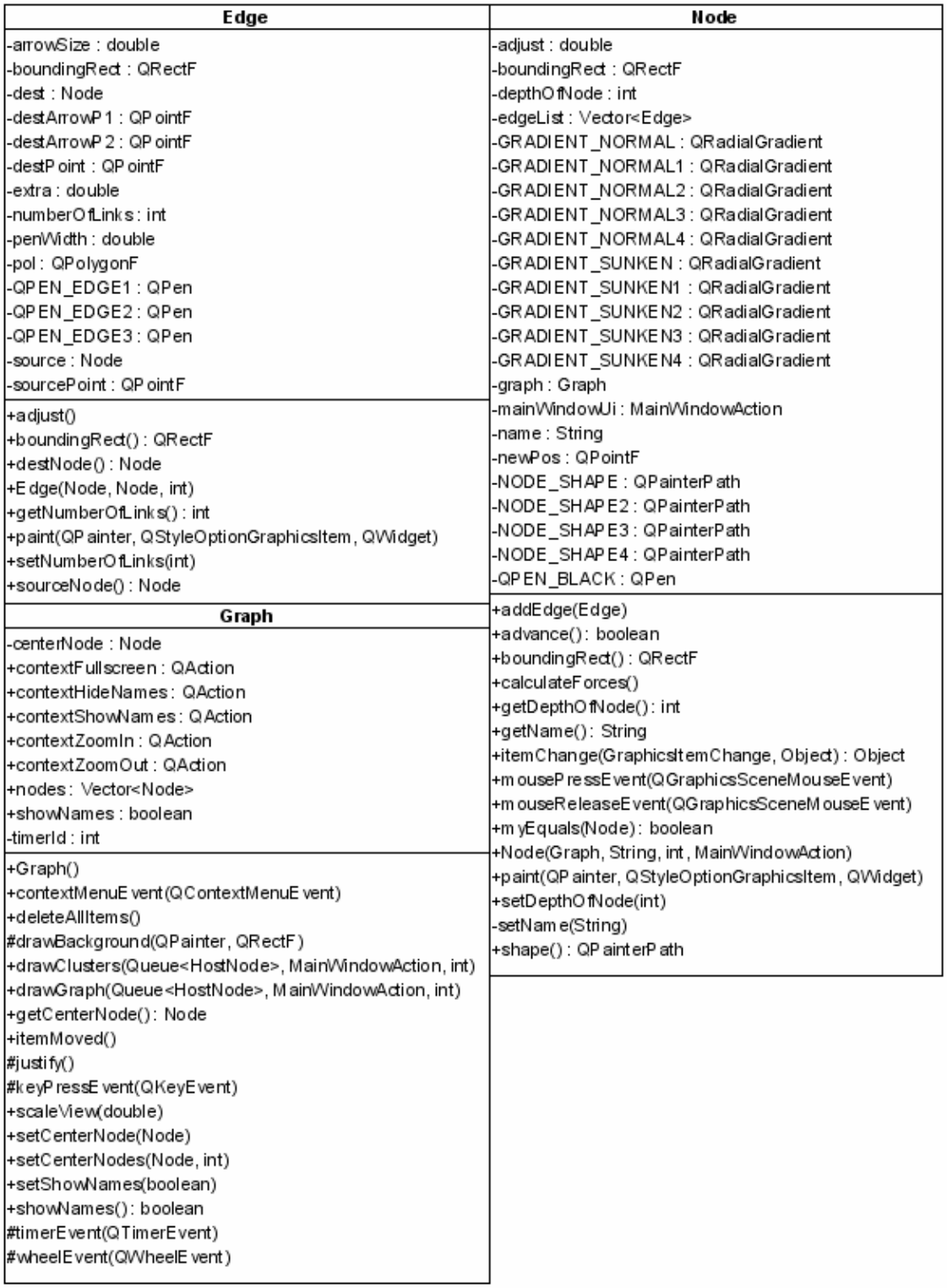

Abbildung 10: Klassendiagramm des Pakets `wwwVis.visualisation`

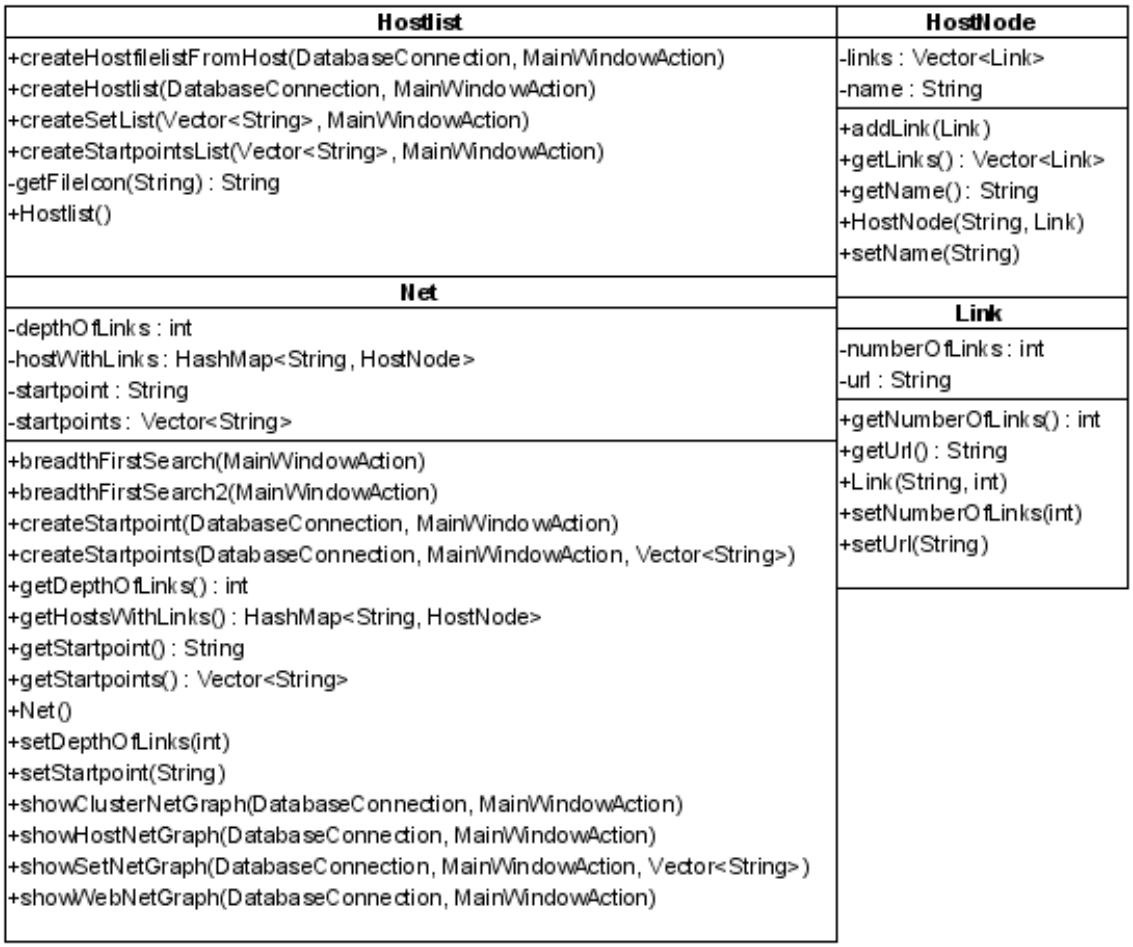

Abbildung 11: Klassendiagramm des Pakets `wwwVis.graph`

4.4 Operationen im Daten- und Darstellungsraum

Nachfolgend werden drei in *wwwVis* implementierte Operationen und Algorithmen näher vorgestellt. Zum einen die Verwendung der Basisdaten und die Breitensuche als Operationen im Datenraum, sowie zum anderen die Visualisierung als Operation im Darstellungsraum.

4.4.1 Verwendung der Basisdaten

Mit Hilfe der *JDBC* kann auf die Daten der Basisdatenbank zugegriffen werden. Anfragen in *wwwVis* sind in *SQL* formuliert. Nach Ausführung der Anfragen ist die Ergebnismenge in einem `ResultSet` gespeichert und kann anschließend in der Anwendung verwendet werden. Für die Abfrage aller Hosts der Datenbank, die später im *TreeWidget* aufgelistet werden sollen, verwendet *wwwVis* eine einfache *SQL*-Anfrage, wobei *schema* dem Datenbankschema und *filteroptions* den aktuell eingestellten Filteroptionen entsprechen:

```
SELECT HNAME FROM schema.HOSTS
                  WHERE filteroptions ORDER BY HNAME
```

Aufgrund der Struktur der verwendeten Basisdatenbank sind dagegen Anfragen, in denen ermittelt wird welche Webressource auf welche anderen linkt, wesentlich komplexer. Da ein mehrfacher *Join* von Tabellen nötig ist, werden die Selektionen, die durch die Filteroptionen garantiert werden, so früh wie möglich in der *SQL*-Anfrage durchgeführt. Beispielhaft hier die in *wwwVis* verwendete *SQL*-Anfrage zur Berechnung der Hosts, auf die jeder einzelne Host der Basisdatenbank linkt. Ein Tupel der Ergebnismenge besitzt drei Attribute: den Hostnamen *(h1.NAME)*, den Namen des Hosts auf den *h1.NAME* linkt *(h2.NAME)* und die Anzahl der existierenden Links *(ALL)*.

```
SELECT h1.HNAME, h2.HNAME AS LNAME, ALL FROM
  (SELECT HID, LHID, SUM(NUMBER) AS ALL FROM
    (SELECT FRID,HID AS LHID, COUNT(FRID) AS NUMBER
    FROM schema.LINKS l, schema.RESOURCES r WHERE
    l.RID=r.RID GROUP BY FRID,HID)
  AS L1, schema.RESOURCES a
  WHERE L1.FRID=a.RID GROUP BY HID,LHID) AS H3,
  (SELECT * FROM schema.HOSTS WHERE filteroptions) AS h1,
  (SELECT * FROM schema.HOSTS WHERE filteroptions) AS h2
WHERE h1.HID=H3.HID AND h2.HID=H3.LHID
```

4.4.2 Breitensuche

Die Breitensuche ist in der Klasse `Net` implementiert. Sie durchläuft den in einer *HashMap* gespeicherten Graphen mit seinen Knoten und Kanten nach der Linktiefe geordnet, beginnend mit dem vorher definierten Startknoten und bis zu der vorher definierten Linktiefe. Nach dem Startknoten werden alle Knoten der Linktiefe 1 und deren Links betrachtet. Danach die Knoten der Linktiefe 2 usw. Alle besuchten Knoten werden in der `drawQueue` gespeichert und markiert. Dadurch wird ein erneutes Durchlaufen der Knoten verhindert. Die drawQueue enthält letztendlich alle Knoten mit ihren Kantenlisten, die später im Graphik-Wiget visualisiert werden können. Im ungünstigsten Fall müssten alle Knoten und alle Kanten durchlaufen werden. Dann wäre die Laufzeit der Breitensuche, für alle Knoten V und alle Kanten E, in *wwwVis* $O(|V|+|E|)$. Tatsächlich ist sie aber wesentlich besser, da im Webgraphen nicht alle Knoten miteinander verbunden sind.

4.4.3 Visualisierung

Die Visualisierungen in *wwwVis* wurden insbesondere mit Hilfe der *Qt Jambi* Klassen `QPainter`, `QPointF`, `QRectF`, `QPolygonF`, `QPen`, `QRadialGradient` und `QGraphicsScene` realisiert. `QPainter` ist die Hauptklasse für Graphikoperationen in *Qt Jambi*. Mit ihr ist das Zeichnen einfacher geometrischer Figuren in einem Widget möglich. Die Visualisierung der Knoten und Kanten wurde in *wwwVis* mit Hilfe der `QPainter` Klasse implementiert.
Knoten werden in den visuellen Darstellungen im Graphik-Widget als Ellipsen, gefüllt mit einem radialen Verlauf (`QRadialGradient`), jeweils in verschiedenen Farben dargestellt. Die Veranschaulichung der Kanten geschieht wie üblich als Linie mittels der Klasse `QPen`, angefangen von einem Punkt (`QPointF`) in der Darstellung zu einem anderen `QPointF`. Alle Kanten im Webgraphen haben eine Richtung, die durch eine Pfeilspitze symbolisiert wird. Die Pfeilspitze ist ein Polygon und damit eine Instanz der Klasse `QPolygonF`. Das Graphik-Widget wurde als Rechteck der Klasse `QRectF` definiert und mit einer gräulichen Hintergrundfarbe gefüllt. Um weitere graphische Elemente in das Graphik-Widget einfügen und das Zooming realisieren zu können, muss das Widget mit einer Fläche als Szene (`QGraphicsScene`) überdeckt werden. Erst dann können die Kanten und Knoten ins Graphik-Widget eingefügt werden.

5 Beispielvisualisierungen und -analysen mit *wwwVis*

5.1 HostNet

Um beispielsweise die Frage zu beantworten, welche Publikationen der *Lehrstuhl für Datenbanken und Informationssysteme* der Universität Düsseldorf auf seiner Website zur Verfügung stellt, kann man die Funktion `HostNet` von *wwwVis* nutzen. Zur Recherche sind folgende Arbeitsschritte mit *wwwVis* notwendig:

1. Verbinden mit der Basisdatenbank über *Start –> Connect to Database*, nach erfolgreicher Verbindung zur Datenbank erscheint eine Meldung im Nachrichtenfenster
2. Anzeigen der Hostliste über *HostNet –> Show Hostlist* und Auswahl des Hosts *dbs.cs.uni-duesseldorf.de* per Rechtsklick aus dem *TreeWidget* oder alternativ direkte Auswahl des Hosts über *HostNet –> Choose Host to show all files of Host*, nun werden alle Dateien eines Hosts aufgelistet
3. Filteroptionen können gegebenenfalls über *Filter –> Filtypes* angewandt werden, um die Anzahl der angezeigten Dateien zu verringern, z.B. wenn nur HTML/XHTML-Dateien angezeigt werden sollen
4. Die Visualisierung der Struktur des Host kann schließlich über *HostNet –> Show HostNet* im Graphik-Widget veranschaulicht werden (Abbildung 12)

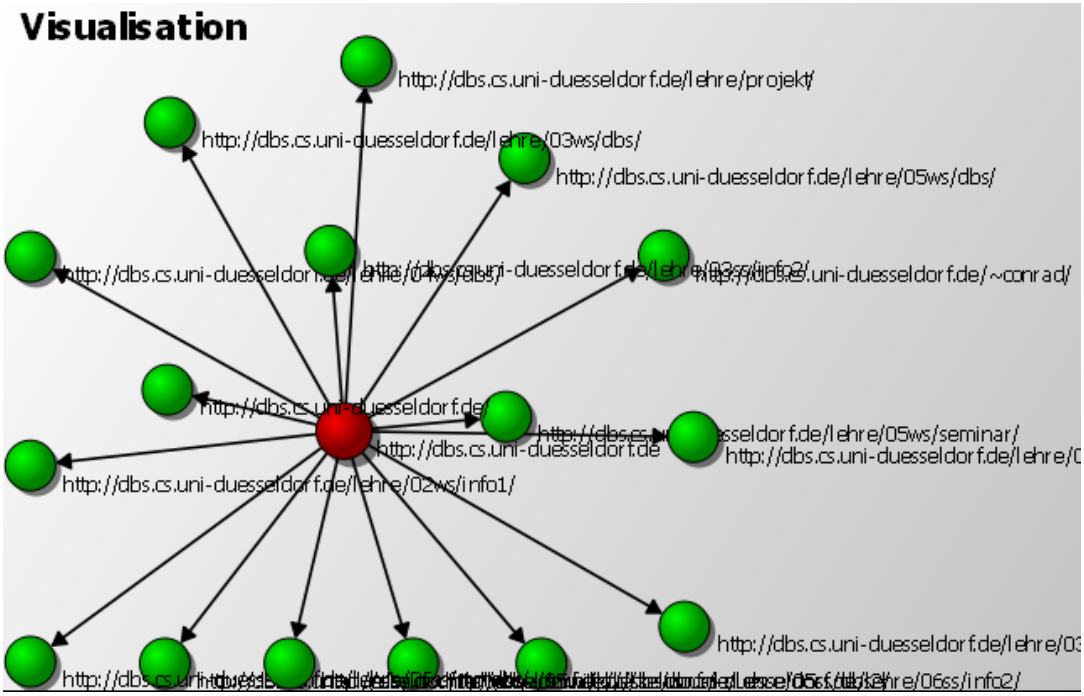

Abbildung 12: *HostNet*-Beispiel erstellt mit *wwwVis*

5.2 WebNet

Abbildung 13 zeigt eine Anwendung von WebNet. Der Host *www.cs.uni-duesseldorf.de* wurde über *WebNet –> Choose Host as Startpoint* als Startpunkt ausgesucht. *WebNet –> Show WebNet* mit Linktiefe 2 ergibt die Darstellung wie im Bild. In der Hostliste kann man erkennen, welche Host von *www.cs.uni-duesseldorf.de* aus über zwei Links erreichbar sind. Offensichtlich linkt der Startpunkt hauptsächlich auf Websites innerhalb der Universität. Ebenso wird deutlich, dass eine hohe Linkdichte zwischen *www.cs.uni-duesseldorf.de* und *www.uni-duesseldorf.de* besteht (rote Kante). Um den Host *www.uni-duesseldorf.de* bildet sich ein Cluster aus Knoten der Linktiefe 2, die nicht direkt von dem Startpunkt aus erreicht werden können.

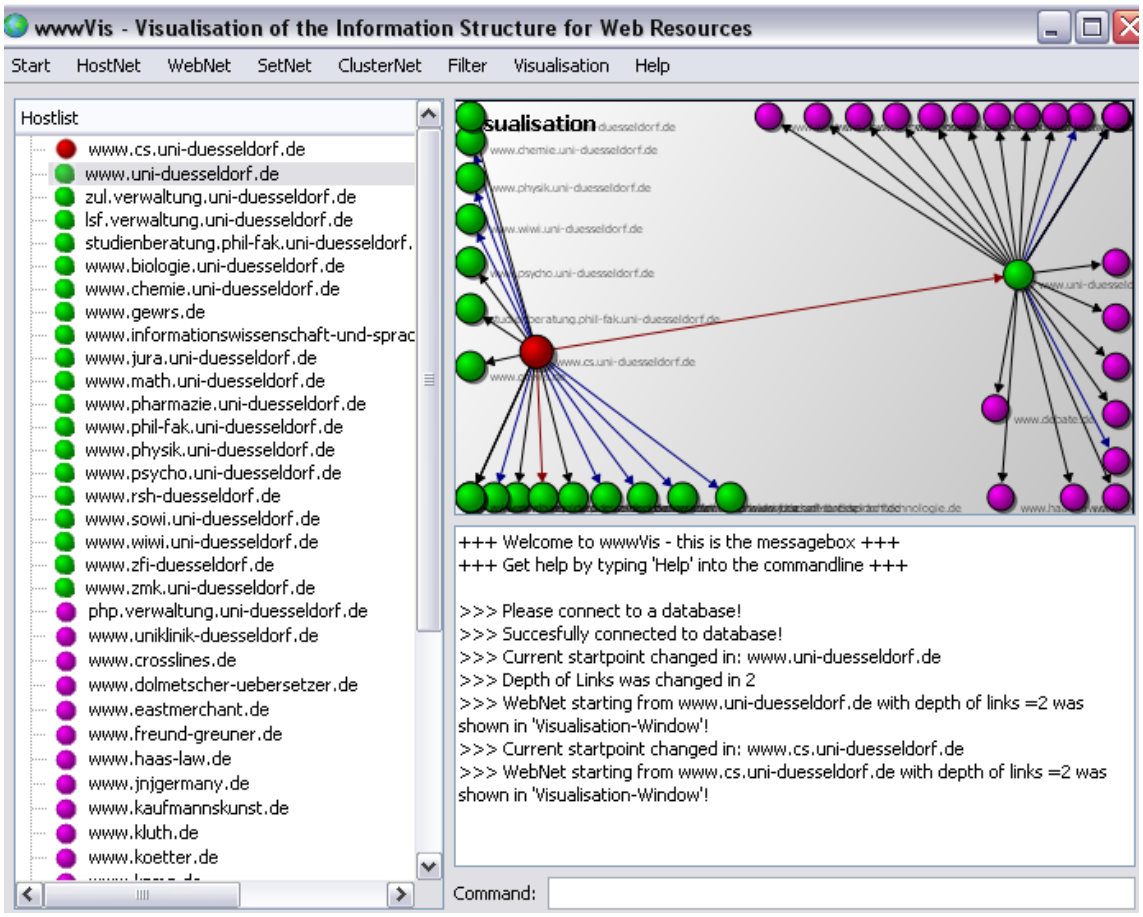

Abbildung 13: *WebNet*-Beispiel erstellt mit *wwwVis*

5.3 SetNet

Die Funktion *SetNet* untersucht nur eine bestimmte, vorher definierte Menge von Hosts. Im Beispiel der Abbildung 14 wird die Menge aller Hosts, die „uni“ enthalten, mit der höchsten Linktiefe, nämlich 4, und dem Startpunkt *dbs.cs.uni-duesseldorf.de* analysiert (*SetNet –> Choose Set of Hosts*, dann *SetNet –> Show SetNet*).
In dieser Visualisierung bilden sich drei Cluster, jeweils um die Hosts *dbs.cs.uni-duesseldorf.de*, *www.cs.uni-duesseldorf.de* und *www.uni-duesseldorf.de*. Zusätzlich ist hier noch die Anzahl der Links in der Hostliste zu sehen. Zwischen *dbs.cs.uni-duesseldorf.de* und *www.cs.uni-duesseldorf.de* gibt es also 662 Links, die in der Basisdatenbank erfasst wurden.

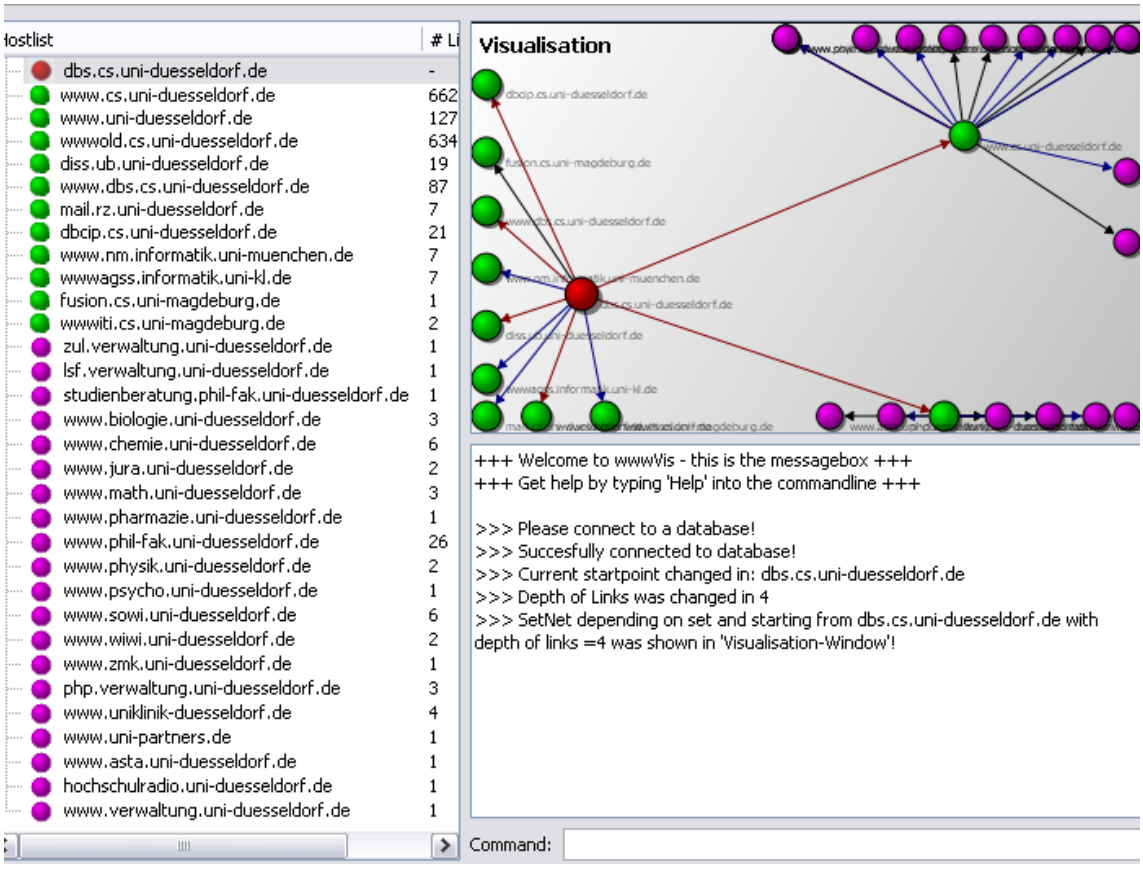

Abbildung 14: *SetNet*-Beispiel erstellt mit *wwwVis*

5.4 ClusterNet

Im Beispiel für die Anwendung der *ClusterNet*-Funktion (Abbildung 15) wurden zunächst vier Startpunkte ausgewählt (*ClusterNet –> Choose Hosts as Startpoints*), von denen beginnend, mit Linktiefe 1, die Linkstruktur angezeigt werden sollte (*ClusterNet –> Show ClusterNet*). Für die Hosts *www.faqs.org* und *www.boutell.com* gibt es jedoch keine Links in der Basisdatenbank (siehe Fehlermeldung im Nachrichtenfenster), so dass nur die Cluster um *www.w3.org* und *www.htmlhelp.com* visualisiert werden.

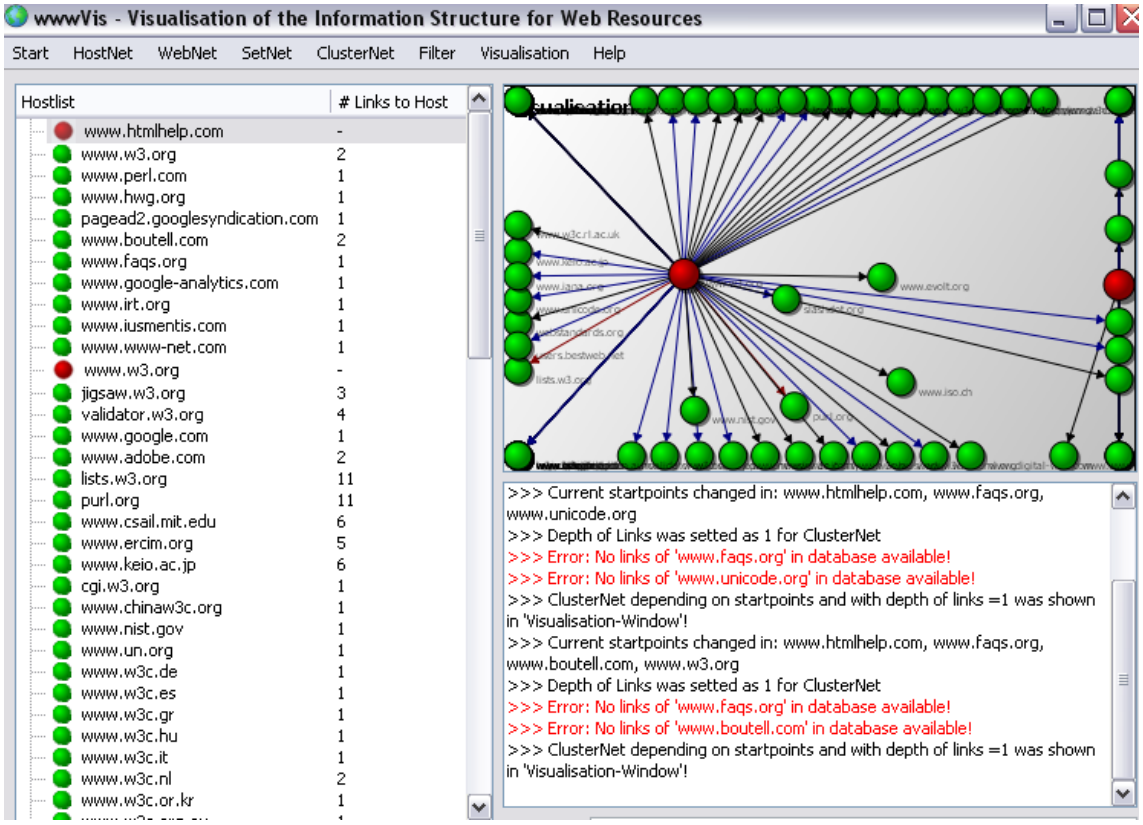

Abbildung 15: *ClusterNet*-Beispiel erstellt mit *wwwVis*

6 Abschluss

6.1 Fazit

In dieser Arbeit wurde mit *wwwVis* ein Tool zur Visualisierung der Informationsstruktur von Webressourcen entwickelt und implementiert. Die Entwicklung geschah auf Basis von Erkenntnissen über die Anforderungen an eine optimale Visualisierung, sowie mit Hilfe von den Möglichkeiten und Techniken einer Daten- bzw. Informationsvisualisierung.
Beim Entwurf und der Realisierung wurde zudem auf die Benutzbarkeit, Angemessenheit und Effektivität der Anwendung geachtet. Trotz der großen Grundlage an Daten sollte es aufgrund der *Focus-and-Context-Visualiesierung* gelungen sein, ein nützliches Tool zu implementieren, das in Zukunft auch für verschiedene Zwecke und Analysen eingesetzt werden kann.
Im Vergleich zu klassischen Informationssystemen liefert *wwwVis* eine bessere Möglichkeit der Übersicht über Strukturen im Web. So sollte es zumindest eine gute Ergänzung und Unterstützung zu Visualisierungen mit gängigen Informationssystemen sein.
Darüber hinaus zeigen die Beispielanalysen der Arbeit, dass sinnvolle Visualisierungen und Analysen mit *wwwVis* leicht durchführbar sind und der Nutzer von der Anwendung profitiert.

6.2 Ausblick auf zukünftige Arbeiten

In zukünftigen Arbeiten wäre es sinnvoll, die beiden Bachelorarbeiten *Extraction and Storage of Web Structures* und *Focus-and-Context Visualisierung der Informationsstruktur von Webressourcen* in einer Anwendung zu verbinden. Da für den Web-Crawler zur Ermittlung der Basisdaten noch kein *GUI* existiert und die Erweiterung der Basisdaten für künftige Analysen mit *wwwVis* erforderlich ist, sollten beide Anwendungen über ein gemeinsames Tool ausführbar sein.
Weiterentwicklungen von *wwwVis* könnten z.B. eine Exportfunktion der Visualisierungen als Bild bzw. HTML-Datei oder mehrere Graphik-Widgets für Vergleiche zwischen Visualisierungen sein. Ferner wäre eine Anbindung von *wwwVis* an das WWW eine weitere sinnvolle Erweiterung zum bestehenden Programm. So könnte die Anwendung von mehreren Benutzern gleichzeitig verwendet werden, was allerdings zu starken Performanceverlusten führen könnte. Eine Anbindung von *wwwVis* an mehrere verschiedene Datenbanken, die eine Erweiterung der Basisdaten zur Folge hätte, könnte auch die Effizienz der Analyse mit Hilfe des Programms steigern.

Literatur

[Aba06] ABALENKOVS, Maksims: *Extraction and Storage of Web Structures*. Bachelorarbeit an der Heinrich Heine Universität Düsseldorf, Institut für Informatik, Lehrstuhl Datenbanken und Informationssysteme, 2006

[Ada99] ADAMIC, Lada A.: *The Small World Web*. Proceedings of the 3rd European Conference on Digital Libraries, 1999

[AJB99] ALBERT, Reka ; JEONG, Hawoong ; BARABASI, Albert-Laszlo: *The diameter of the world wide web*. In: Nature, 410:130-131, September 1999

[BKM+00] BRODER, Andrei ; KUMAR, Ravi ; MAGHOUL, Farzin ; RAGHAVAN, Prabhakar ; RAJAGOPALAN, Sridhar ; STATA, Raymie ; TOMKINS, Andrew ; WIENER, Janet: *Graph structure in the web*. In: Computer Networks, 2000

[BL90] BERNERS-LEE, Tim: *Information Management: A Proposal*. CERN, Genf, May 1990

[BLFM05] BERNERS-LEE, Tim ; FIELDING, R. ; MASINTER, L.: *Uniform Resource Identifier (URI): Generic Syntax, IETF RFC 3986*. Version: January 2005. `http://www.ietf.org/rfc/rfc3986.txt?number=3986`. – Online Ressource, Abruf: 18. 08 2007

[BYRN99] BAEZA-YATES, Ricardo ; RIBEIRO-NETO, Berthier: *Modern Information Retrieval*. Addison Wesley Verlag, 1999

[CER07] CERN: *Was sind die größten Erfolge des CERN? - Das World Wide Web*. Version: 2007. `http://public.web.cern.ch/Public/Content/Chapters/AboutCERN/Achievements/WorldWideWeb/WWW-de.html`. – Online Ressource, Abruf: 15. 08 2007

[Chi00] CHI, Ed H.: *A Taxonomy of Visualization Techniques using the Data State Reference Model*. Proc. of the IEEE Symposium on Information Visualization, IEEE Computer Society Press, 2000

[Day99] DAY, Rebecca: *Think Research: Web hounds*. Version: 1999. `http://domino.research.ibm.com/comm/wwwr_thinkresearch.nsf/pages/webhounds399.html`. – Online Ressource, Abruf: 20. 08 2007

[Däs99] DÄSSLER, Rolf: *Informationsvisualisierung - Stand, Kritik und Perspektiven*. In: Methoden/Strategien der Visualisierung in Medien, Wissenschaft und Kunst, Wissenschaftlicher Verlag, Trier, 1999

[IAN07] IANA: *MIME Media Types*. Version: 2007. `http://www.iana.org/assignments/media-types/`. – Online Ressource, Abruf: 19. 08 2007

[IBM07] IBM: *IBM DB2 Informationszentrale*. Version: 2007. `http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp`. – Online Ressource, Abruf: 21. 08 2007

[Kei02] KEIM, Daniel A.: *Datenvisualisierung und Data Mining*. In: Datenbank-Spektrum, Februar 2002

[KL01] KLEINBERG, Jon ; LAWRENCE, Steve: *The Structure of the Web*. In: Science, November 2001

[KMH02] KOSARA, Robert ; MIKSCH, Silvia ; HAUSER, Helwig: *Focus and Context Taken Literally*. In: IEEE Computer Graphics and its Applications, Special Issue: Information Visualization, Januar/Februar 2002

[Krü03] KRÜGER, Guido: *Handbuch der Java-Programmierung, 3. Auflage*. Addison Wesley Verlag, 2003

[KR05] KUROSE, James F. ; ROSS, Keith W.: *Computer Networking - A Top-Down Approach Featuring the Internet: 3rd Edition*. Addison Wesley Verlag, 2005

[Lew05] LEWANDOWSKI, Dirk: *Web Information Retrieval. Technologien zur Informationssuche im Internet*. DGI Schrift, Frankfurt am Main, 2005

[MS04] MEINEL, Christoph ; SACK, Harald: *WWW - Kommunikation, Internetworking, Web-Technologien*. Springer-Verlag, Berlin Heidelberg New York, 1.Auflage, 2004

[Nor99] NORTH, Klaus: *Wissensorientierte Unternehmensführung: Wertschöpfung durch Wissen.* Gabler Verlag, Wiesbaden, 1999

[PRR99] PROBST, G. ; ROMHARDT, K. ; RAUB, S.: *Wissen managen. Wie Unternehmen ihre wertvollste Ressource optimal nutzen*. Gabler Verlag, Wiesbaden, 1999

[QJA07] *Trolltech Qt Jambi API Javadoc Documentation*. Version: 2007. `http://doc.trolltech.com/qtjambi/index.html`. – Online Ressource, Abruf: 21. 08 2007

[QJD07] *Trolltech Qt Jambi Reference Documentation*. Version: 2007. `http://doc.trolltech.com/qtjambi/com/trolltech/qt/qtjambi-index.html`. – Online Ressource, Abruf: 21. 08 2007

[QJW07] *Trolltech Qt Jambi Whitepaper*. Version: 2007. `http://trolltech.com/pdf/QtJambi_4.3.0_Whitepaper_A4.pdf`. – Online Ressource, Abruf: 21. 08 2007

[QTD07] *Trolltech Qt Reference Documentation (Open Source Edition)*. Version: 2007. `http://doc.trolltech.com/4.3/index.html`. – Online Ressource, Abruf: 21. 08 2007

[QTW07] *Trolltech Qt 4.3 Whitepaper*. Version: 2007. `http://trolltech.com/pdf/qt43-whitepaper-a4.pdf`. – Online Ressource, Abruf: 21. 08 2007

[RWS00] RAUSCHENBACH, Uwe ; WEINKAUF, Tino ; SCHUMANN, Heidrun: *Interactive Focus and Context Display of Large Raster Images*. In: Proceedings WSCG 2000, The 8-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media, Plzen, Czech Republic, February 2000

[Sch04] SCHUMANN, Heidrun: *Konzepte und Methoden der wissenschaftlichen Visualisierung*. Kartographische Bausteine Band 26, TU Dresden, Juni 2004

[SH00] SAAKE, Gunter ; HEUER, Andreas: *Datenbanken - Konzepte und Sprachen*. mitp-Verlag, 2.Auflage, Januar 2000

[Shn96] SHNEIDERMAN, Ben: *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations*. In: IEEE Visual Languages, College Park, Maryland/USA, July 1996

[SK03] SCHUMANN, Heidrun ; KREUSELER, Matthias: *Fokus und Kontext-Darstellung im geographischen Kontext*. In: Proceedings GeoVis 2003, Hannover, Februar 2003

[SM04] SCHUMANN, Heidrun ; MÜLLER, Wolfgang: *Informationsvisualisierung: Methoden und Perspektiven*. In: it - Information Technology, Oldenbourg Verlag, März 2004

[SS04] SAAKE, Gunter ; SATTLER, Kai-Uwe: *Algorithmen und Datenstrukturen - Eine Einführung mit Java*. dpunkt.verlag, 2.Auflage, 2004

Abbildungsverzeichnis

1 Client/Server-Modell für ein verteiltes Hypertextsystem (aus [BL90]) . . . 6
2 Ein Webgraph erstellt mit *wwwVis* . . . 9
3 Beispiel für eine Datenvisualisierung: Globale Verteilung von Erdbebenepizentren (aus [Däs99]) . . . 14
4 Beispiel für eine Informationsvisualisierung: Hyperbolic Tree - Visualisierung des Dateibaums einer Festplatte (aus [Däs99]) . . . 15
5 *Information Visualization Data-State-Reference Model* nach [Chi00] (aus [Sch04]) 16
6 Beispiele für *Focus-and-Context Visualisierungen* mit Hilfe einer *Perspective Wall* (aus [SK03]), mit Hilfe des *Fish-Eye-View*-Effekts (aus [SM04]) und aus der Photographie (aus [KMH02]) . . . 20
7 *Entity-Relationship Modell* der Basisdatenbank (nach [Aba06]) . . . 23
8 Die graphische Benutzeroberfläche von *wwwVis* . . . 27
9 Paketdiagramm von *wwwVis* . . . 34
10 Klassendiagramm des Pakets `wwwVis.visualisation` . . . 35
11 Klassendiagramm des Pakets `wwwVis.graph` . . . 36
12 *HostNet*-Beispiel erstellt mit *wwwVis* . . . 39
13 *WebNet*-Beispiel erstellt mit *wwwVis* . . . 40
14 *SetNet*-Beispiel erstellt mit *wwwVis* . . . 41
15 *ClusterNet*-Beispiel erstellt mit *wwwVis* . . . 42

Tabellenverzeichnis

1 Beispiele für Uniform Resource Identifier (URI) . . . 8
2 Beispiele für Uniform Resource Locator (URL) . . . 8
3 Beispiele für MIME-Typen . . . 8
4 Beispieleinträge der Tabelle *Hosts* aus der Basisdatenbank . . . 23
5 Beispieleinträge der Tabelle *Resources* aus der Basisdatenbank . . . 23
6 Beispieleinträge der Tabelle *Links* aus der Basisdatenbank . . . 24