Multiple Links im World Wide Web

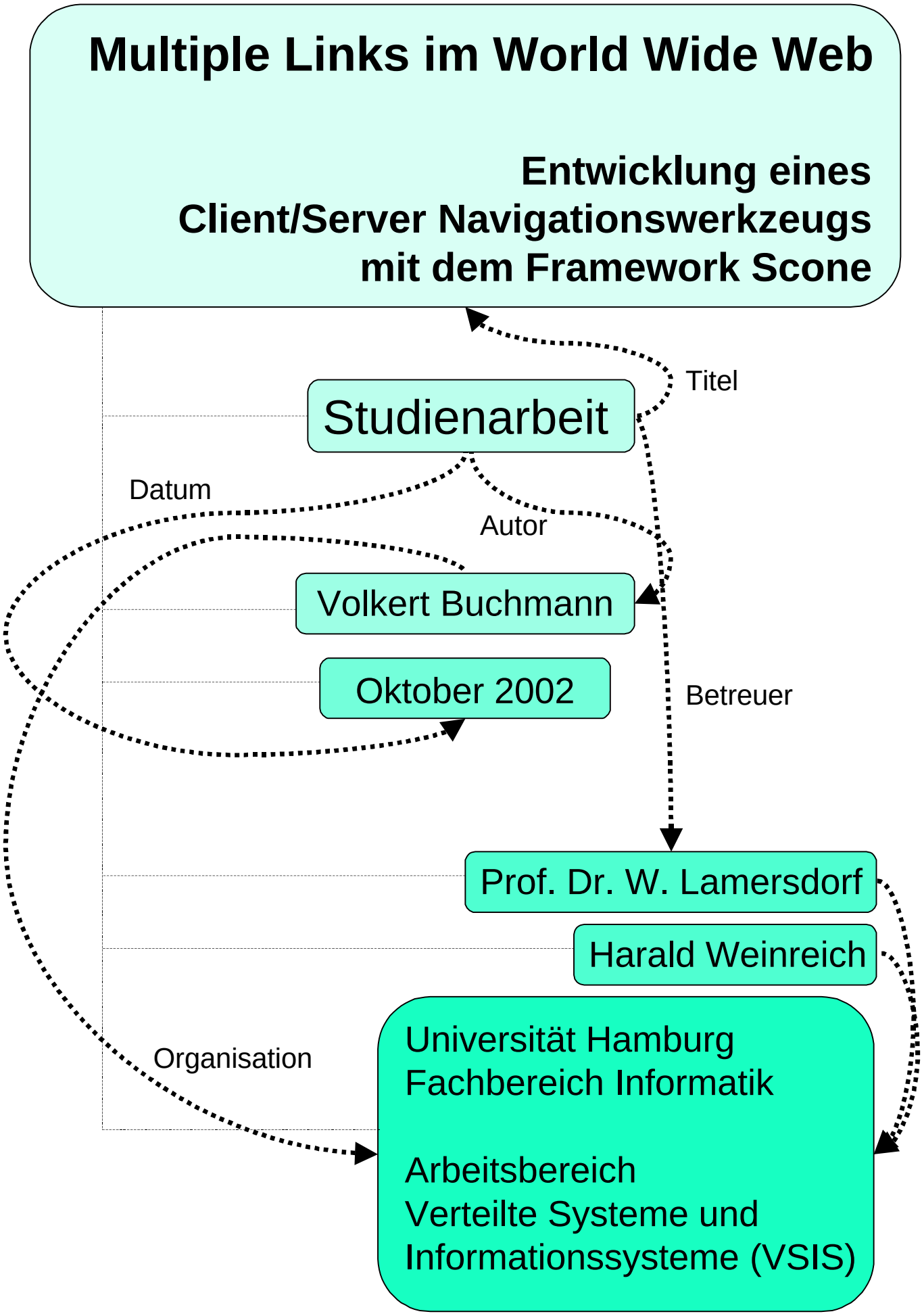
Entwicklung eines Client/Server-Navigationswerkzeugs mit dem Framework Scone

Studienarbeit

Volkert Buchmann
November 2002

Betreuer:
Prof. Dr. Winfried Lamersdorf

Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Verteilte Systeme und Informationssysteme (VSIS)


Multiple Links im World Wide Web
Entwicklung eines
Client/Server Navigationswerkzeugs
mit dem Framework Scone
Titel
Studienarbeit
Datum
Autor
Volkert Buchmann
Oktober 2002
Betreuer
Prof. Dr. W. Lamersdorf
Harald Weinreich
Organisation
Universität Hamburg
Fachbereich Informatik
Arbeitsbereich
Verteilte Systeme und
Informationssysteme (VSIS)

Inhaltsverzeichnis

1 Einleitung

Übt die unkomplizierte weltweite Vernetzung von Informationen im World Wide Web zunächst eine Faszination auf die Anwendenden aus, so folgt alsbald die Ernüchterung. Während sich die Benutzung von wesentlich kleineren, lokalen Hypertexten schon als schwierig erwiesen hat, ist die verteilte, unstrukturierte Informationsflut des WWWs nur schwer zu bewältigen. Die Technik bleibt hierbei weit hinter dem Potenzial des WWWs zurück und unterstützt die AutorInnen und LeserInnen in dieser Hinsicht nur wenig.

Obwohl Probleme mit der Orientierung und Navigation in Hypertexten schon vor der Einführung des WWWs bekannt waren, besitzt das WWW keine inhärenten Mechanismen, die den BenutzerInnen in dieser Beziehung helfen. Zusätzlich wurden viele erprobte oder vorgeschlagene technische Konzepte nicht umgesetzt, wie beispielsweise Kollaborationsunterstützung, vielfältige Linkmodelle oder Versionierung. Diese Studienarbeit befasst sich mit der Erweiterung des World Wide Webs um einige dieser Konzepte. Hierbei stehen die Trennung von Links und Knoten und die Verwendung von Links mit multiplen Zielen im Vordergrund.

Während im heutigen WWW Links in Dokumente eingebunden werden, unidirektional sind und nur auf genau ein anderes Dokument verweisen können, soll erprobt werden, wie ein erweitertes Linkmodell eingesetzt werden kann. Diese Studienarbeit beschränkt sich dabei auf unidirektionale Links, die auf mehrere Zielknoten gleichzeitig verweisen können. In diesem Zuge wird nicht nur eine mögliche technische Realisiserung diskutiert, sondern insbesondere auch eine mögliche Visualisierung. Nur wenn eine Visualisierung gefunden werden kann, die die BenutzerInnen nicht zusätzlich ablenkt oder überfordert, ist die Realisierung eines solchen Konzepts tatsächlich sinnvoll. Um die technische Realisierung zu ermöglichen, werden dynamisch erzeugte, extern verwaltete Links benutzt. Somit wird gleichzeitig mit eigenständigen Linkobjekten experimentiert.

Prototypische Implementationen mit dem Framework Scone sollen die Umsetzung dieser Konzepte in der Praxis erproben. Das Framework Scone wurde am Fachbereich Informatik entwickelt und bietet eine Grundlage für die schnelle Entwicklung und Evaluierung von WWW-Navigations-Prototypen.

Zu Beginn wird in Kapitel 2: „Hypertext und das World Wide Web“ das World Wide Web als ein Hypertext-System beleuchtet. Dabei wird deutlich, welche Konzepte Hypertext beinhaltet und welche dieser Konzepte im WWW fehlen. Mit den Open Hypermedia Systems wird dabei

ein Forschungszweig erwähnt, der sich unter anderem mit der Erweiterung des WWWs um einige dieser Konzepte bemüht.

Anschließend werden in Kapitel 3: „Multiple Links und das World Wide Web“ verschiedene Möglichkeiten zur Umsetzung dieser Erweiterung diskutiert. Hierzu werden verschiedene Ansätze zur Architektur, zur Datenrepräsentation eines multiplen Links, zu Übertragungsprotokollen und schließlich zur Visualisierung erörtert. Zugleich werden die in dieser Arbeit verfolgten Lösungswege in diesem Kontext positioniert.

Nachdem die grundlegenden Fragen unseres Ansatzes geklärt wurden, werden in Kapitel 4: „Multiple Links in der Praxis“ verschiedene experimentelle Implementation vorgestellt, mit denen die Realisierbarkeit der vorgestellten Konzepte erprobt wird. Diese Prototypen wurden mit Hilfe des Frameworks Scone erstellt, das in diesem Abschnitt ausführlich vorgestellt wird. Des Weiteren wird kurz eine Schnittstelle zur Datenbank der Suchmaschine Google beleuchtet, die als Datenquelle für die Implementationen diente.

Anschließend werden in Kapitel 5: „Ähnliche Arbeiten“ Projekte vorgestellt, die sich ebenfalls mit der Erweiterung des WWWs um multiple Links bemühen und auf die eine oder andere Art einen ähnlichen Weg gewählt haben, wie er in dieser Studienarbeit verfolgt wird.

Kapitel 6: „Fazit“ zieht ein Resümee und gibt einen Ausblick auf mögliche Folgeprojekte.

2 Hypertext und das World Wide Web

Das World Wide Web (WWW) als verteiltes Hypertextsystem soll im Folgenden im Kontext der Hypertext-Forschung betrachtet werden. Dabei wird nicht nur deutlich, welche besondere Stellung es in diesem Kontext einnimmt: viele visionäre Konzepte von früheren Hypertextsystemen sind im WWW nicht umgesetzt worden. Einige dieser Konzepte sollen vorgestellt werden, und mit den Open Hypermedia Systems wird ein Forschungszweig beleuchtet, der sich unter anderem um die Erweiterung des WWWs bemüht.

2.1 Hypertext

Hypertext[1] ist ein Medium, dass durch seine Struktur unser Denken und unsere Kommunikation unterstützen soll [Conklin 1987]. Vannevar Bush, der „Grossvater" des Hypertexts, beschreibt die niemals realisierte Maschine „Memex"[2], die eine „vergrößerte, nahe Ergänzung zum (...) Gedächtnis" [Bush 1945] von WissenschaftlerInnen sein sollte. Bush suchte nach einem Weg, die in den Bibliotheken akkumulierte unüberschaubare Wissensmenge zugreifbar zu machen. Die Memex sollte den WissenschaftlerInnen helfen, in dem sie diese Informationen verfügbar machte und dem menschlichen Geist entsprechend organisierte. Bush ging davon aus, dass der menschliche Geist durch Assoziation arbeitet, und so sollte seine Maschine die Möglichkeit bieten, zwischen allen in ihr gespeicherten Dokumenten Verbindungen herzustellen, denen maschinell gefolgt werden kann. Ein ähnliches Ziel verfolgten Douglas Engelbart mit seinem Projekt NLS und Ted Nelson, der 1965 den Begriff „Hypertext" prägte. In [Nelson 1972] wird die Idee der Memex auf die damalige Computertechnik abgebildet und so mit Nelsons bis heute nicht vollständig umgesetztem Hypertextsystem Xanadu[3] verglichen. Er beschreibt, wie Bushs Idee des Trails in die der Links transformiert wird, und wie Dokumente von entfernten Rechnern verlinkt werden. Für eine Geschichte des Hypertexts siehe [Nielsen 1995] und [Müller-Prove 2002].

Der so durch Verweise aufgespannte Hypertext besitzt nach [Conklin 1987] zwei grundlegende Eigenschaften: die *„node-ness"* und die *„linked-ness"*. In Hypertext wird der Inhalt in relativ kleinen Knoten abgelegt, die dann durch Links miteinander verbunden sind. Dies führt dazu, dass der Text nicht mehr sequentiell verläuft. Die LeserIn muss nun Entscheidungen treffen, welchem Verweis sie folgt. Anders als bei traditionellen sequentiellen Text ist der

[1] In dieser Arbeit werden die Ausdrücke „Hypertext" und „Hypermedia" synonym benutzt, d.h. wenn wir von „Text" reden, kann gegebenenfalls auch jedes andere Medium wie Bilder oder sogar Klänge gemeint sein.
[2] „Memory Extender" [Nielsen 1995, S. 33]
[3] An Xanadu wird heute noch gearbeitet, siehe http://www.aus.xanadu.com

Leseverlauf nicht mehr vorgegeben. Während gedruckte Texte häufig Links innerhalb dieser Texte wie Fußnoten[4] ebenso wie Verweise auf externe Texte[5] anbieten, werden sie doch nicht Hypertxt genannt. [Conklin 1987] führt an, dass der Talmud, Aristoteles Schriften oder moderne Enzyklopädien trotz ihrer starken Verlinkung nicht als Hypertexte angesehen werden, da häufig noch eine dritte grundlegende Eigenschaft von Hypertext verlangt wird: Die Navigation, also das Folgen der Links muss von einer Maschine, unterstützt werden (Siehe das Zitat aus [Nielsen 1995] in Fußnote 4).

Einigen AutorInnen reichen diese Anforderungen nicht aus, sie fordern zusätzlich *Guided Tours* oder *Übersichten* der durch die Knoten und Links aufgespannten Netzwerke. [Halasz 1988] beispielsweise stellt sieben Forderungen auf, die Hypertextsysteme erfüllen sollten. Diese Forderungen umfassen Suchmöglichkeiten ebenso wie multiple Links oder Möglichkeiten zur Kollaboration. [Nielsen 1995, S.5] hingegen fordert nur, dass ein System dem für Hypertext typischen „*Look and Feel*“ entspricht. Kurz gefasst gehört zu diesem „Look and Feel“ die interaktive Navigation von (fast) jedem Knoten.

Je nachdem, welche dieser Eigenschaften in den jeweiligen Hypertextsystemen realisiert sind, können große Hyperdokumente so unübersichtlich werden, dass sie eventuell schwer zu benutzen sind. [Conklin 1987] prägte hierfür den Ausdruck „*lost in space*“. Er sieht zwei grundsätzliche Probleme bei der Benutzung von Hypertext: *Desorientation* und *Cognitive Overhead*. Die Desorientation kommt dabei durch die Tendenz zustande, in nichtlinearen Dokumenten den „Ortssinn“ zu verlieren. Der Cognitive Overhead wird durch die Wahlmöglichkeiten und die dadurch entstehende „gleichzeitige“ Verfolgung unterschiedlicher Pfade erzeugt. Hypertext-Projekte sollten – auch wenn sie nicht um eine Lösung dieser Probleme bemüht sind – dies immer im Auge behalten, und versuchen, diese Probleme zumindest nicht zu vergrößern. Das in dieser Studienarbeit vorgestellte Projekt beispielsweise könnte möglicherweise zu mehr Desorientaion und kognitiven Zusatzkosten führen, da es die Anzahl der Wahlmöglichkeiten für zukünftige Pfade erhöht.

In der Literatur sind jedoch nicht nur Beschreibungen der Eigenschaften von Hypertextsystemen zu finden. Auch zugrundeliegende Datenmodelle sind wichtig und werden diskutiert. [Conklin 1987] etwa beschreibt Hypertext als ein Medium, dass traditionelle Informatik-

[4] ...wie beispielsweise diese Fußnote. Sie haben sich beim Lesen des Textes entschieden, dem Verweis zu dieser Fußnote zu folgen, und haben das sequentielle Paradigma dieser Arbeit durchbrochen. Das macht diese Arbeit jedoch nicht nicht zu einem Hypertext, da diese Fußnote keine andere Wahl anbietet, als wieder zum sequentiellen Textfluss zurückzukehren. Enstprechend fordert [Nielsen 1995, S. 4], dass „...wahrer Hypertext den BenutzerInnen das Gefügl gibt, dass sie sich frei durch die Informationen bewegen können.“

[5] Durch einen solchen Verweis (nämlich auf [Nielsen 1995]) bietet Fußnote 4 doch eine andere Wahl, die Fußnote zu verlassen.

Grenzen überschreitet. So ist Hypertext eine *Datenbank-Methode*, ein neuer Weg, auf Daten direkt zuzugreifen. Hypertext ist auch ein *Repräsentations-Schema*, eine Art semantisches Netzwerk, das textuelle Daten mit formalen und mechanisierten Operationen vermischt. Und Hypertext ist eine *Interface-Art*, die Link-Icons[6] an beliebigen Positionen im Text einfügt.

Analog dazu[7] beschreiben [Campbell & Goodman 1988] die *General Purpose Hypertext Abstract Machine* (HAM), ein abstraktes Hypertext-Modell[8], das Hypertext-Systeme in drei entsprechende Abstraktionsschichten einteilt: Die unterste Schichte ist die *Datenbasis-Schicht*[9]. Sie ist damit beauftragt, die Objekte des Systems zu speichern und bietet Mechanismen, um diese Objekte abzurufen. Die *Hypertext Abstract Machine (HAM)-Schicht* beschreibt das Datenmodell des jeweiligen Systems. Hier wird festgelegt, welche Objekte es im System geben soll, beispielsweise Textknoten und einfache, unidirektionale Links und welche primitiven Operationen auf ihnen möglich sind. Die oberste Schicht ist die *Darstellungs-Schicht*[10]. Sie bestimmt, wie die BenutzerInnen auf die Objekte der HAM-Schicht zugreifen können und wie sie dargestellt werden. [Nielsen 1995, S. 178] beschreibt, wie das WWW in dieses Modell eingeordnet werden kann: Die Datenbasis-Schicht enthält die teilnehmenden Computer und das Internet als zugrundeliegende Speicher- und Transportmedien. HTML und HTTP bilden die HAM-Schicht, während die Browser in der Darstellungs-Schicht angesiedelt sind.

Das HAM-Modell basiert nach [Campbell & Goodman 1988] auf fünf verschiedenen Objekten[11]: Graphen, Kontexten, Knoten, Links und Attributen. Ein *Graph* ist das höchste HAM-Objekt. Er soll alle Informationen zu einem bestimmten Thema enthalten, beispielsweise eine komplette Dokumentation. Er ist also das, was [Conklin 1987] ein *Hyperdocument* nennt und was in Intermedia ein *Web* genannt wird [Campbell & Goodman 1988]. Ein *Kontext* partitioniert den Graphen in logische Subgraphen, beispielsweise

---

[6] Conklin benutzt den Ausdruck „Link-Icon“ um die visuelle Repräsentation eines Links auszudrücken. Heute ist eher der Begriff „Link-Marker“ gebräuchlich.

[7] Das HAM-Modell ist keine Antwort auf Conklins Artikel. Es wird von [Conklin 1987] selbst zitiert, ist also älter.

[8] Elaborierter als das HAM-Modell ist das Dexter Hypertext Referenz Modell ([Halasz & Schwartz 1994]). Da jedoch fraglich ist, ob dies ein adäquates Modell für eingebettete Referenzen wie das WWW ist ([Grønbæk & Trigg 1996]) wird hier das HAM-Modell benutzt, dem laut [Nielsen 1995, S. 178] auch das WWW folgt.

[9] Im Original „Database-Level“. Laut [Nielsen 1995, S. 131] auch als Datenbank-Schicht zu verstehen. Dies ist jedoch irreführend. Die Datenbasis-Schicht dient allein der Speicherung und Zurverfügungstellung der Daten. Die HAM-Schicht kann dann analog als DMBS angesehen werden, da erst sie definiert, welche Objekte es gibt, und wie auf sie zugegriffen werden kann.

[10] In [Campbell & Goodman 1988] wird die Darstellungs-Schicht ursprünglich noch in *Application Tools* und *User Interface* getrennt. Wir schliessen uns hier [Nielsen 1995] an, der diese beiden Subschichten zu einer zusammenfügt.

[11] genauer gesagt Klassen

Kapitel[12]. Ein *Knoten* enthält beliebige Daten. Das HAM-Modell sieht Versionierungen und Suchfunktionen auf diesen Knoten vor. Ein *Link* definiert eine Beziehung zwischen Knoten. *Attribute* können zu Kontexten, Knoten und Links hinzugefügt werden, und beschreiben die Semantik dieser Objekte.

Das Link-Modell der HAM bietet keine Links mit multiplen Zielen, wir müssen es daher für unsere Zwecke erweitern. Allgemein beschreibt ein Link eine Beziehung zwischen Ressourcen[13]. Diese Ressourcen können Knoten oder Teile von Knoten sein, und werden auch Anker genannt. Hat ein Link eine Richtung, wird von Start- und Ziel-Ankern gesprochen. Zeigt ein Link von einer Ressource weg, wird er in Bezug auf diese Ressource *outgoing Link* (kurz: Outlink) genannt, andersherum wird er *incoming Link* (kurz: Inlink) genannt. Ein Link, dessen Startressourcen und Endressourcen beide nicht im selben Knoten wie die Linkdefinition liegen, werden *3rd Party Links* genannt. Zusätzlich können Links von einem bestimmten *Typ*[14] sein, sie können eine Bezeichnung (einen *Titel*) haben und sie können *annotiert* sein. Hat ein Link genau einen Startanker und genau einen Zielanker, so ist er ein 1-zu-1 oder ein einfacher Link. Hat ein Link genau einen Startanker und möglicherweise mehrere Zielanker, so ist er ein 1-zu-n Link. Dies sind zwei Spezialfälle von n-zu-m Links. Die 1-zu-n Links sind Gegenstand dieser Arbeit. Im Folgenden werden sie der Kürze halber multiple Links oder kurz Multilinks genannt.

Mit diesen Grundlagen können nun einige Defizite des World Wide Webs herausgearbeitet werden.

2.2 Das World Wide Web

Das World Wide Web ist das derzeit bekannteste Hypertextsystem. Es wird von Millionen Menschen weltweit benutzt[15]. Das zugrundeliegende Hypertextmodell ist jedoch recht einfach und kennt die meisten der in den folgenden Abschnitten vorgestellten Konzepte nicht. Nach [Berners-Lee et.al. 1994] soll das WWW ein „*Pool menschlichen Wissens*“ sein, der es entfernten TeilnehmerInnen eines Projekts erlaubt, sich auszutauschen. Auch Querverweise

[12] Beachte, dass [Campbell & Goodman 1988] den Graphen als den „root context“ betrachten, somit ist der Kontext das höchste Objekt, nicht der Graph.
[13] Eine Ressource ist dabei eine beliebige adressierbare Einheit.
[14] Treffender ist hier die Bezeichnung *Rolle*, wie sie XLink verwendet. Partizipieren zwei Ressourcen A und B, die jeweils eine Person repräsentieren, an einem Link, kann durch die Rolle beispielsweise ausgedrückt werden, dass A das Kind von B ist. Genau dies ist hier mit Typ gemeint.
[15] Nach [Wilkens 2002] benutzen heute 28,3 Millionen Menschen allein in Deutschland das Internet (das hier fast synonym fürdas WWW gesehen werden kann).

zwischen diesen Projekten sollen möglich sein. Dem WWW zugrunde liegt die Idee einer „*grenzenlosen Informationswelt*“, in der alle Objekte über ihre Referenz erhältlich sind.

Technisch gesehen basiert das WWW auf den folgenden drei Konzepten:

- Ein einheitliches Adresssystem (*URI*[16]), das über Protokollgrenzen hinweg funktioniert und das diese Welt ermöglichen soll.
- Ein spezielles Protokoll (*HTTP*[17]) zur Übertragung von Hypertext.
- Eine Markup-Sprache (*HTML*[18]), in der Knoten-Inhalte und Links definiert werden, und die jeder WWW-Client versteht.

Die Sprache HTML sieht nur die Verwendung von ASCII-Zeichen vor, HTML-Dokumente können daher leicht zwischen verschiedenen Plattformen ausgetauscht werden. Zudem können sie leicht mit existierender Software bearbeitet werden [Müller-Prove 2002, S. 12]. Links werden in HTML definiert und direkt in HTML-Dokumente eingebettet. Sie sind einfache Outlinks, die einen Titel und einen Typ haben können. Sie können alles referenzieren, was durch eine URI adressierbar ist. In Knoten eingebettete Links werden von vielen AutorInnen als problematsich angesehen. Siehe Abschnitt 3.2: „Architekturen zur Trennung von Links und Knoten“[19].

Das WWW sieht eine einfache Client-Server Architektur vor. Die Clients können die über URIs adressierten Knoten von Servern herunterladen. Proxys können zwischen Client und Server geschaltet werden, und dienen meist dazu, die Knoten so lokal wie möglich vorzuhalten, um die durch die Verteilung des WWWs auftretenden Wartezeiten zu verringern.

HTML beschreibt nur die Struktur eines Dokuments, nicht aber sein Erscheinungsbild[20]. Hierdurch können HTML-Dokumente auf den unterschiedlichsten Plattformen dargestellt werden, von reinen Textkonsolen bis zu hochauflösenden Grafikworkstations. Die Darstellung der HTML-Dokumente wird durch die Clients, sogenannte *Browser*[21] vorgenommen. Diese Browser bieten gleichzeitig ein einziges Interface für die verschiedenen, durch die URIs vereinheitlichten Dienste. Mit dem Erscheinen des einfach zu installierenden und benutzenden

[16] Uniform Ressource Identifier [Berners-Lee et.al. 1998]. Die URI-Spezifikation umfasst URLs und URNs.
[17] Hypertext Transfer Protocol [Fielding et. al. 1999]
[18] Hypertext Markup Language [Raggett et.al. 1999]
[19] [Nelson 1997] geht sogar noch weiter, und warnt davor, überhaupt Markup in den Inhalt eines Knoten einzubetten.
[20] Während dies für die ersten Versionen von HTML noch überwiegend stimmt, so wurden im folgenden immer mehr Layout-Komponenten in den Standard mit aufgenommen. Daher wurde versucht, Struktur und Layout durch Cascading Stylesheets [Bos et.al. 1998] zu trennen.

Browers Mosaic 1993 wurde das WWW außerordentlich populär [Berners-Lee et.al. 1994]. Sicherlich trugen hierzu auch das einfache Konzept des WWWs und die relativ einfach zu administrierenden Webserver bei.

2.3 Open Hypermedia Systems

Open Hypermedia Systems (OHS) sehen im Gegensatz zum WWW eine strikte Trennung von Knoten und Links vor. Sie versuchen, eine Umgebung zu schaffen, in der Dokumente – unabhängig von der Applikation mit der sie erstellt und betrachtet werden – untereinander verlinkt werden können. [Anderson & Sherba 2001] heben drei Eigenschaften von Open Hypermedia hervor: *Fortgeschrittene Hypermedia Datenmodelle* wie beispielsweise n-äre Links und Links zwischen Links, *3rd Party Links*, die in einer Linkbase verwaltet werden und die *Integration von 3rd Party Applikationen*, um die Vorzüge bisheriger Software nicht aufgeben zu müssen. OHS sind für das WWW von Bedeutung, da sie eine wesentlich verbesserte Integration des WWWs versprechen, und zusätzliche Linkmodelle wie multiple Links einführen. Für eine Beschreibung einer OHS Referenz-Architektur siehe [Nürnberg & Leggett 1997].

Ein Beispiel für ein OHS ist das in [Davis et.al. 1992] beschriebene System Microcosm. Microcosm erlaubt es, zwischen Dokumenten, die von verschiedenen Applikationen verwaltet werden, zu verlinken. Hierbei gibt es drei verschiedene Stufen der Zusammenarbeit von Microcosm mit diesen Applikationen: *fully aware*, *partially aware* und *unaware*. Fully aware sind Applikationen, die speziell für Microcosm implementiert wurden. Ihnen ist es problemlos möglich, Links in Dokumenten sichtbar und traversierbar zu machen. Partially aware sind Applikationen wie Microsoft Word, die mit Hilfe ihrer Skripting-Funktionalität dazu gebracht werden können, Links anzuzeigen und traversierbar zu machen. Unaware sind Applikationen, die nicht über die nötigen Schnittstellen und Skripting-Fähigkeiten verfügen, um Links aus einer Linkbase zu laden.

2.4 „Neue“ Konzepte für das WWW

Die oben erwähnten Defizite des WWWs sollen nun etwas genauer beschrieben werden.

[Halasz 1988] beispielsweise forderte Eigenschaften für die (damals) nächste Generation von Hypertext, die das Web auch heute zum großen Teil nicht bietet. Unter anderem zählte er *Guided Tours*, ein *erweitertertes Linkmodell*, *Versionierung* und *Kollaborationsunterstützung* auf. [Bouvin 2000] stellt die Frage „What is wrong with the Web?“ und kritisiert das be-

[21] Zuvor waren „Browser“ Programme, die eine navigierbare Übersicht des Graphen visualisierten (siehe u.a.

schränkte Link-Modell[22]. [Bieber et. al. 1997] beschreiben ihrer Meinung nach wichtige Eigenschaften, die HTML-Links nicht aufweisen. Unter anderem fordern sie, dass LeserInnen von Hypertext ihre *privaten Links* und *Annotationen* zu fremden Knoten erstellen können[23]. [Pam 1995] kritisiert das WWW im Vergleich mit Xanadu und kritisiert u.a. die fehlende Unterstützung für *Metadaten*.

Aus den im vorigen Absatz bemühten Texten wurden nur einige Kritikpunkte aufgeführt, eine vollständige Auflistung der Kritik am WWW würde den Rahmen dieser Arbeit deutlich sprengen. Neben den vielfach geäußerten Kritikpunkten wie fehlende Versionierung und Kollaboration wird meist das simple Linkmodell des WWWs kritisiert. Hierunter können andere oben erwähnte Teilprobleme wie Annotationen [Nielsen 1995, S. 142] und Guided Tours [Grønbæk et.al. 2000] subsummiert werden. Diese Arbeit versucht daher, das WWW um ein erweitertes Linkmodell zu bereichern. Dabei beschränkt sie sich auf multiple 3rd Party Links.

Diesen Ansatz teilt sie mit einer ganzen Reihe von Projekten aus der OHS-Community. [Anderson 1997] etwa beschreibt diverse Ansätze, das WWW und OHS miteinander zu verbinden. Während der simple Export von Daten in ein anderes System kein zufriedenstellendes Konzept sein kann, scheinen hybride Modelle (u.a. von [Whitehead et.al. 1996] favorisiert), die beispielsweise WWW-Server oder -Clients OHS-aware machen, größere Erfolgsaussichten zu haben. In gewisser Weise ist das in dieser Studienarbeit beschriebene Projekt ein solcher hybrider Ansatz. Er verbindet die eingebetteten Links des WWW mit Links, die aus einer Linkbase abgefragt werden. Von einem OHS zu sprechen, wäre hierbei jedoch übertrieben, da die hinzugefügten Links wiederum nur Links in das WWW sind. Es werden also lediglich Techniken verwandt, die Teil eines solchen hybriden Ansatzes sind, wie beispielsweise in [Oberholzer & Wilde 2002] und [Halsey & Anderson 2000] beschrieben.

[Conklin 1987]).

[22] Das Titelbild dieser Studienarbeit wurde vom Titelblatt von [Bouvin 2000] inspiriert.

[23] Sie beschreiben ein elaboriertes Link-Modell, vergessen jedoch multiple Links.

3 Multiple Links und das World Wide Web

Ziel dieser Studienarbeit ist eine experimentelle Erweiterung des beschränkten Link-Modells des World Wide Webs durch ein reichhaltigeres, namentlich durch Links mit mehreren Zielen. Hierzu soll der Webserver die normalen, in den HTML-Dokumenten eingebetteten Links durch solche erweiterten Links ersetzen, die er getrennt von den Dokumenten in einer Datenbasis gespeichert hat. Gegebenenfalls besorgt sich der Webserver weitere Link-Informationen, beispielsweise von Google. Der Client muss zudem in der Lage sein, die vom Server bereitgestellten Links entsprechend zu interpretieren, visualisieren und navigierbar zu machen.

Im Folgenden werden die Grundlagen dieses Ansatzes diskutiert, um das Projekt zu positionieren und genauer zu spezifizieren.

3.1 Generierte Links – lebendiger Hypertext

Die zu dieser Studienarbeit gehörenden Implementationen benutzen generierte Links, um das Modell der multiplen Links zu erproben. Ein kurzer Exkurs zu generierten Links und ihren Auswirkungen auf den Hypertext soll diesen Aspekt beleuchten. [Bodner & Chignell 1999] unterscheiden hierzu, je nachdem, wie Links erstellt bzw genutzt werden, *Static Hypertext*, *Dynamic Hypertext* und *Adaptive Hypertext*. Anhand dieser Kategorien soll nun der dynamische Charakter der in diesem Projekt benutzten multiplen Links untersucht werden.

Im statischen Hypertext ändern sich Links nicht. Verknüpft mit dem *Strong Authoring* Modell von [Bodner & Chignell 1999], in dem Knoten-AutorInnen die Beziehungen in und zwischen Knoten festlegen, beschreibt dies das ursprüngliche Modell des World Wide Webs. HTTP und HTML sind auch heute noch hauptsächlich für statischen Hypertext ausgelegt, die CGI-Spezifikationen von HTTP zielen hauptsächlich auf die dynamische Generierung von Texten, nicht auf das dynamische Einbinden von Verweisen.

Im dynamischen Hypertext können sich Links automatisch verändern. Beispielsweise können zwei Absätze in verschiedenen Dokumenten anhand von Textähnlichkeiten verlinkt werden. Wird ein solcher Link nicht nur einmal im Voraus berechnet, hat dies den Vorteil, dass beispielsweise in bestimmten Intervallen auf Grund einer aktualisierten Datenbasis der Link erneut berechnet oder auf seine Konsistenz überprüft werden kann. Ein grosser Vorteil im dynamischen Hypertext ist die Möglichkeit, das UserInnenverhalten in die Generierung der Links miteinzubeziehen, um wesentlich gezielter zu verlinken.

Einen etwas anderen Weg verfolgt der adaptive Hypertext. Er orientiert sich an einem Modell der Vorlieben der jeweiligen AnwenderIn, berechnet aber keine entsprechenden Links, sondern manipuliert schon vorhandene. Beispielsweise können je nach Vorlieben der AnwenderInnen Links ausgeblendet, Link-Listen sortiert oder annotiert werden. Hierbei muss es sich nicht um dynamisch erstellte Links handeln, so dass dynamischer und adaptiver Hypertext durchaus verschieden sind. [Miller & Waltz 1998] beschreiben das Projekt COOL-Links, das entsprechend userInnenspezifischen Vorgaben Links automatisch aus einer Liste auswählt. Ein guter Einstieg zu adaptivem Hypertext ist in [Brusilovsky 1997] zu finden[24].

[Ashman et.al. 1997]beschreiben analog die verschiedenen Arten von Links. Sie können *handgemacht* sein oder *berechnet* worden. Bei den berechneten Links kann wiederum zwischen im Voraus berechneten (*precomputed*) und *dynamischen* Links unterschieden werden. [Ashman et.al. 1997] weisen explizit auf den Unterschied zwischen dynamischen und precomputed Links hin: werden dynamische Links gespeichert und wiederverwendet, so können sie als „gecachete“ Links bezeichnet werden, nicht jedoch als precomputed Links, diese ändern sich nie. Die möglichen Kombinationen der verschiedenen Arten von Links und Hypertext-Modellen sind vom Autoren dieser Arbeit in der Tabelle 3.1.1 aufgetragen[25].

Tabelle 3.1.1: Die verschiedenen Hypertext-Modelle und ihre Links

Hypertext / Links	handmade	precomputed	dynamic
Statisch	+	+	
Adaptiv	+	+	+
Dynamisch			+

Diese verschiedenen Hypertext-Modelle haben unterschiedliche Auswirkungen auf das Surfverhalten (und somit wahrscheinlich auch auf den Surferfolg) der AnwenderInnen. Im statischen Hypertext haben die AnwenderInnen nur die von den AutorInnen vorgegebenen Linkstrukturen zur Verfügung, und können Probleme haben, diesen zu folgen. Zwischen allen möglichen Pfaden zwischen den einzelnen Knoten muss den vorgegebenen Pfaden gefolgt werden, Pfade, die nicht unbedingt für die AnwenderInnen nachvollziehbar sind [Bodner & Chignell 1999]. Bei größeren Hypertext-Systemen mit handgemachten Links werden die Links von verschiedenen AutorInnen erstellt, die zum Teil signifikant unterschiedliche Kriterien für ihre Links haben [Wilkinson und Smeaton 1999]. Innerhalb einer Website kann die

[24] Dort wird u.a. zwischen adaptiver *Präsentation* und adaptiver *Navigationsunterstützung* unterschieden. Es ist anzumerken, dass auch für statischen und dynamischen Hypertext der Inhalt und dessen Präsentation jeweils statisch bzw. dynamisch erzeugt werden kann. Da diese Studienarbeit sich vorrangig mit Links befasst, haben wir diese wichtigen Bereiche vernachlässigt.

Desorientierung der AnwenderInnen durch Sitemaps oder Menüs gemindert werden, siteübergreifend hilft dies jedoch nicht.

[Bodner & Chignell 1999] beschreiben zwei Modi des Surfverhaltens: Suchen und Browsen. Diese beiden Modi werden beim Surfen ständig iteriert, jedesmal auf Kosten der Kontinuität. Sie sind optimistisch genug, zu behaupten, dass diese beiden Modi des Surfens im dynamischen Hypertext in einen überführt werden können, wenn in die Berechnung der dynamischen Links nicht nur der Kontext des aktuellen Texts, sondern auch der persönliche Kontext der AnwenderIn miteinbezogen wird. Sie weisen außerdem darauf hin, dass dynamischer Hypertext den AnwenderInnen wie statischer Hypertext erscheinen kann. Einerseits ist dies durchaus wünschenswert, um eine möglichst einfache Schittstelle zu haben, andererseits bringt dies auch die Nachteile des statischen Hypertexts mit sich: Es wird wieder ein bestimmter Pfad durch den Hyperspace vorgegeben. Auch wenn dieser nun besser an der Zielsetzung der AnwenderInnen orientiert ist, mag er sich nicht mit deren Vorstellungen decken. Andere Probleme, die sich mit berechneten Links ergeben können sind falsche oder paradox erscheinende Links, zu viele Links oder auch die teure Berechnung von dynamischen Links zur Laufzeit. Auch die sich für die BenutzerInnen eventuell aus unersichtlichen Gründen verändernden Links im Adaptiven oder Dynamischen Hypertext könnten zu Verwirrungen führen. [Tebbutt 1999] untersucht die Auswirkungen von precomputed semantischen Links in einem großen Online-Manual, in dem zuvor hauptsächlich mit Hilfe einer Suchmaschine navigiert wurde. Er fand heraus, dass Links, die semantische Ähnlichkeit zwischen Dokumenten repräsentieren, dazu führen, dass die gesuchten Informationen schneller gefunden werden konnten als zuvor. Er beobachtete aber auch sehr unterschiedliche Reaktionen der BenutzerInnen, die von „sehr positiv, über Konfusion, zu negativ“ (ebd.) reichten. Insgesamt muss noch einiges an Forschungsarbeit geschehen, bis wir berechnete oder gar dynamische Links tatsächlich in Produkten wiederfinden.

Das in dieser Arbeit beschriebene Projekt stellt eine Hybridform aus statischem und dynamischen Hypertext dar. Ausgehend von wahrscheinlich statischem Hypertext mit von Menschen erstellten Links wird dynamisch zu jedem Link ein multipler Link berechnet. Diese Multilinks beinhalten den ursprünglichen statischen Link und die dynamisch dazu berechneten Alternativlinks. Sie befinden sich außerdem genau an den jeweils von der AutorIn vorgesehenen Stelle im Text. Durch die Hybridform der so berechneten Multilinks sollen die Vorteile der verschiedenen Hypertext-Modelle genutzt werden. In dieser Arbeit wird davon ausgegangen,

[25] Hierbei sind auch Mischformen denkbar. Adaptiver Hypertext mit dynamisch erzeugten Links ist auch gleichzeitig dynamischer Hypertext.

dass die AutorInnen meist an sinnvoller Stelle einen Link in eine sinnvolle Richtung[26] setzen. Ausgehend von dieser sinnvollen[27] Vorgabe werden dann die Vorteile der dynamischen Links genutzt, und die statischen Links durch Alternativen angereichert. Der von Hand erstellte statische Link soll dabei die Orientierung der BenutzerInnen in dem Dokument erleichtern, und das oben angesprochene Problem der sich mit der Zeit scheinbar spontan ändernden Links vermeiden. Sicher wäre es zudem sinnvoll, die Vorteile des dynamischen bzw. adaptiven Hypertexts auszunutzen, und die Berechnung der alternativen Links auf den Kontext im Text und den Kontext der BenutzerIn zu stützen. Dies liegt jedoch weit ausserhalb des Bereiches einer Studienarbeit.

3.2 Architekturen zur Trennung von Links und Knoten

Ob die Links in die Knoten fest eingebettet werden, wie dies beispielsweise in HTML geschieht, oder ob sie als eigenständige Objekte gespeichert und erst bei Bedarf explizit gemacht werden, hängt von der Architektur des jeweiligen Hypertextsystems ab.

Wie oben gezeigt, kennt die HAM des WWWs nur die Knoten als eigenständige Objekte, Links tauchen nur als Teile eines Knotens auf. [Brailsford 1999] argumentiert, dass eingebettete, hardkodierte Links, wie sie in HTML definiert sind, schwer zu warten und auf dem neusten Stand zu halten sind[28]. Linkbases, die eine Menge von 3rd Party Links vorhalten, könnten hingegen leichter gewartet werden und vielmehr noch von unterschiedlichen Hypertext-Systemen genutzt werden, solange die in den Linkbases gespeicherten Links in einem möglichst abstrakten Format gehalten werden[29].

[Davis 1995] unterscheidet drei Stufen der Einbindung von Links in die Knoten. In die erste Stufe kann das WWW eingeordnet werden, hier werden alle Informationen über Start-und Zielressourcen und der Link selbst in die Knoten eingebunden. In der zweiten Stufe werden Start- und Zielressourcen in den Knoten definiert, die Links aber extern gespeichert und verwaltet. Die dritte Stufe letztendlich sieht die Links und die Angaben über Start- und Ziel-Ressourcen physikalisch losgelöst von Start- und Zielknoten. Architekturen, die der dritten Stufe

[26] Diese Terminologie ist eigentlich falsch. Ein Link kann zwar eine Richtung haben, aber nicht in eine Richtung zeigen. Wir benutzen diesen Ausdruck hier trotzdem, um deutlich zu machen, dass das verlinkte Dokument nunmehr als „Vorschlag“ oder eben „Richtung“ verstanden wird.
[27] Im WWW ist vielen Links zweifelsohne nicht anzusehen, ob sie „sinnvoll“ angebracht wurden. Wir gehen in dieser Arbeit der Einfachheit halber trotzdem davon aus.
[28] Das WWW wird von ihnen in dieser Hinsicht sogar als Rückschritt angesehen: „Systems which supported link separation were available more than 15 years ago but today's World Wide Web has a document architecture, in HTML, which is an uneasy combination of structural and layout features and where every document has to have its hyperlinks hard-coded within itself.“ [Brailsford 1999]
[29] Tatsächlich sind heute ca. 5% der Links im WWW korrupt [LinkAlarm 2002].

zuzuordnen sind, bieten den Vorteil, dass auch die BenutzerInnen Dokumente verlinken können, für die sie keine Schreib-Rechte besitzen. Hierzu sind zwei Vorbedingungen zu erfüllen [Brailsford 1999]: Die Dokumente müssen strukturiert sein, so dass Start- und Zielressourcen addressiert werden können – beispielsweise mit Xpointer [DeRose et.al. 2002]. Die Namen der Knoten müssen persistent sein, damit Links konsistent bleiben können, dies kann beispielsweise durch URNs erreicht werden[30].

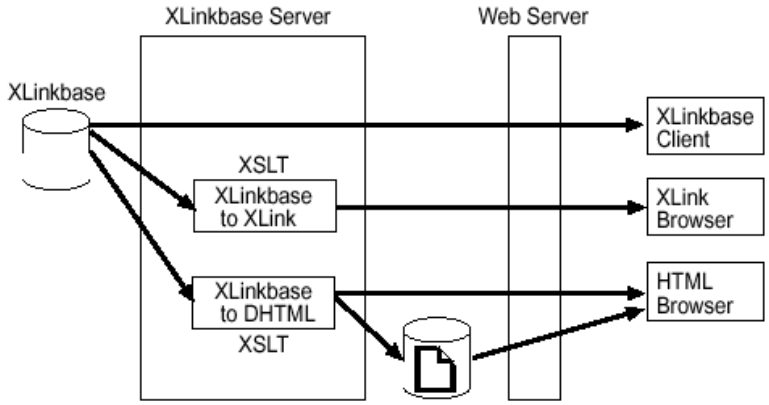


Abbildung 3.1: Architektur in [Oberholzer & Wilde 2002]

[Oberholzer & Wilde 2002] beschreiben eine Architektur, die es verschiedenen Clients ermöglicht, auf Multilinks zuzugreifen (siehe Abbildung 3.1). Heutigen HTML-Browsern werden DHTML-Popups zugeführt, zukünftige XML- und XLink-fähige Browser und spezielle Clients können direkt auf XLinks zugreifen. Ein ähnlicher Ansatz wird in dieser Arbeit verfolgt. Das in dieser Arbeit vorgestellte Projekt ist eine Mischform aus der ersten und zweiten Stufe der Linkeingebundenheit. Ausgehend von den vollständig eingebetteten Links des HTML-Standards wird aus einer Linkbase heraus ein Multilink erstellt. Dieser Multilink wird wiederum in das HTML-Dokument eingebettet, indem er den ursprünglichen einfachen Link ersetzt. [De Roure et. al. 2000] erproben u.a. den Ansatz, den ursprünglichen Link in HTML-Dokumenten durch einen `A`-Tag zu ersetzen, der kein `href`- sondern nur ein `name`-Attribut hat. Der dazugehörige Link könnte dann von einer Linkbase angefordert werden. Diese Lösung würde der zweiten Stufe entsprechen und wäre wesentlich sauberer.

Es wäre von Vorteil, wenn auch im World Wide Web Links als eigenständige Objekte behandelt werden. Hierzu müssten Links separat von den Dokumenten gespeichert und verwaltet werden. Werkzeuge zur Erstellung von HTML-Dokumenten[31] sollten in der Lage sein, die Dokumente und Links getrennt zu verwalten und in einem geeigneten Format an den Webser-

[30] Interessanterweise fordert [Berners-Lee et.al. 1994] dies für das WWW und nennt es als eine der nächsten Aufgaben.

ver zu übergeben. Aus alten oder per Hand erstellten HTML-Dokumenten können die eingebetteten Links wie oben beschrieben automatisch extrahiert werden[32]. [Wilde 2002] schlägt vor, dass Web-Browser jedesmal, wenn sie ein Dokument anfordern, simultan bei einer Linkbase zugehörige Links anfordern, an denen das Dokument partizipiert. In dieser Studienarbeit wird als Alternative vorgeschlagen, dass der Web-Server selbst diese Links zusammen mit dem Dokument an den Browser zurückliefert. Diese Links holt sich der Web-Server im Voraus anstelle des Browsers und speichert sie zwischen. Dies hat die Vorteile, dass durch die dezentrale Haltung der Links an den jeweiligen Webservern die Linkbases entlastet werden, und eventuell Zugriffszeiten verringert werden können. Zudem ist davon auszugehen, dass die meisten Links, an denen eine Ressource partizipiert, von der selben Person erstellt wurden, wie die Ressource. Es ist daher naheliegend, Links und Ressourcen in einer einheitlichen Datenbank zu verwalten, und so auch leicht möglich, über eine einheitliche Schnittstelle auf diese Objekte zuzugreifen. Hierbei muss jedoch für die Aktualität der von externen Linkbases geladenen Links gesorgt werden. Auch können eventuell nicht die spezifischen Wünsche der AnwenderInnen erfüllt werden. So sind ohne weiteres spezialisierte Linkbases denkbar, beispielsweise in Bezug auf Informatik oder Filmstars. Abbildung 3.2 veranschaulicht diese Architektur. Der Dokumentenserver (DS) teilt sich eine lokale Datenbank mit einem Linkserver (LS). Der Browser kommuniziert mit dem Dokumentenserver, der auf Anfrage in der Datenbank vorhandende Links in die Dokumente einfügt. Der Browser hat jedoch auch die Möglichkeit, bei beliebigen Linkservern Links anzufordern. Die Linkserver haben wie oben beschrieben die Möglichkeit, untereinander Links auszutauschen. Nach dieser Architektur gibt es zwei verschiedene Arten von Servern im WWW. Im Folgenden wird mit „Webserver“ jedoch stets der Dokumentenserver gemeint sein.

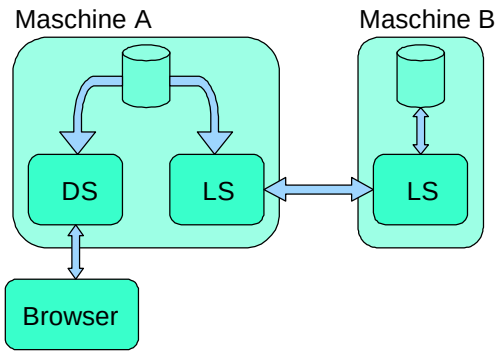


Abbildung 3.2: Die in dieser Studienarbeit angenommene Architektur

[31] Wahrscheinlich werden zukünftige Webserver eher auf XHTML- oder XML-Dokumenten arbeiten. Auch wenn in diesem Text nur HTML-Dokumente erwähnt werden, sollen die Webserver jedoch keinesfalls auf diese eingeschränkt werden.
[32] Das in [Pitkow & Jones 1996] beschriebene ATLAS-System sieht eine ähnliche Architektur vor.

3.3 Datenrepräsentation von multiplen Links

Die multiplen Links, die in dieser Arbeit behandelt werden, haben eine Struktur, die nicht einfach auf bestehende Standards abzubilden ist. Sie haben ein ausgezeichnetes Ziel und mehrere Alternativ-Ziele. Während die HTML- und XHTML-Standards nur sehr einfache Links zulassen, bietet XLink Möglichkeiten, multiple Links zu definieren. Auch wenn die in dieser Arbeit behandelten Verweise nicht ideal auf XLink-Datenstrukturen abgebildet werden können, ist Xlink die Sprache der Wahl, um möglichst kompatibel zu etwaigen anderen Produkten zu bleiben. XLink wird zu diesem Zweck unter anderem auch von [Oberholzer & Wilde 2002], [Wilde 2002] und [Halsey & Anderson 2000] verwandt.

Zunächst soll geklärt werden, wie die Struktur in den von uns behandelten multiplen Links beschaffen ist. In unserem Fall ist ein multipler Link ein Link, der eine Menge von unterschiedlichen einfachen Links enthält. Diese einfachen Links müssen jeweils nur einen Endpunkt definieren. Die Richtung der Links ist damit festgelegt. Einer der einfachen Links wird ausgezeichnet, während die anderen als Alternative zu ihm angesehen werden. Siehe Abbildung 4.5 a) zur Struktur eines solchen Links.

Links in HTML

HTML [Raggett et.al. 1999] sieht eine ganze Reihe von Arten von Verweisen vor. Explizit werden jedoch nur das `A`- und das `LINK`[33]-Element als Links bezeichnet, Nach der natürlichsprachigen Definition eines Links in HTML definieren jedoch auch Elemente wie `IMG` und `FRAME` Links. Links werden als gerichtete Verbindung von zwei Web-Ressourcen definiert. Sie besitzen zwei Enden – Anker genannt – und eine Richtung. Der Quell-Anker ist dabei das Element, das den Link definiert, der Ziel-Anker kann eine beliebige Web-Ressource oder ein Element eines HTML-Dokuments sein[34]. Standardmäßig dienen die `A`-Elemente dazu, die verlinkte Ressource in den Browser zu laden, wenn sie beispielsweise angeklickt wird. Es können jedoch auch andere Aktionen in einer Skriptsprache – beispielsweise JavaScript – angegeben werden. Das `LINK`-Element dient dazu, automatisch Dokumente wie Stylesheets einzubinden, oder um Beziehungen zu anderen Dokumenten explizit zu machen[35]. Geschach-

[33] Im folgenden ist ein `LINK`-Element ein Element mit dem Namen `LINK`, und ein Link-Element ein Element, das einen Link definiert.
[34] Tatsächlich sagt die HTML-Spezifikation nichts darüber aus, was genau der Quell-Anker ist. Manchmal scheint die oben genannte Definition sinnvoll, manchmal muß der Quell-Anker auch als das HTML-Dokument, in dem der Link definiert wurde, interpretiert werden.
[35] Insofern ist es inkonsequent definiert.

telte Links sind explizit illegal. Es ist somit festzuhalten, dass es in HTML generell nicht möglich ist, multiple Links zu definieren.

Links in XHTML

XHTML [McCarron et.al. 2002] definiert in Bezug auf Links im Wesentlichen den HTML-Standard in XML [Bray et.al. 2000]. Dabei werden keine neuen Konzepte eingeführt, sondern vielmehr Begriffe aus dem HTML-Standard verwandt, ohne sie zu definieren. Link-Elemente können auch hier nicht geschachtelt werden, somit ist auch der Nachfolger von HTML nicht in der Lage, multiple Links auszudrücken.

Links in XLink

XLink [DeRose et.al. 2000] definiert Link-Elemente in XML und beschreibt ein wesentlich reicheres Linkmodell als HTML. Ein Link in XLink ist eine explizite Beziehung zwischen Ressourcen. Explizit wird diese Beziehung durch die Verwendung eines Link-Elements gemacht. XLink definiert keine Elemente, sondern Attribute. So kann jedes Element ein XLink-Link sein, wenn es ein Attribut `type` aus dem XLink-Namensraum mit dem entsprechenden Wert (`simple` oder `extended`) besitzt. Elemente vom Typ `simple` und `extended` sind die beiden Arten von Links, die XLink bietet. Simple Links ähneln den `A`-Elementen in HTML, während Extended Links mehrere Ressourcen miteinander in Beziehung setzen können, diese „partizipieren“ dann am Link. Die Ressourcen eines `extended`-Links können durch Elemente vom Typ `locator` bzw. `resource` beschrieben werden. `Locator`-Typ Elemente beschreiben nach XLink Sprachgebrauch anhand einer URI entfernte Ressourcen, während `resource`-Typ Elemente selbst lokale Ressourcen sind[36]. Ressourcen werden also sozusagen von `locator`-Typ Elementen „by reference“ beschrieben und vom `resource`-Typ Element „by value“. Elemente vom Typ `Arc` (Kante, Bogen) legen fest, wie Ressourcenpaare traversiert werden sollen.

Multilinks in XLink

Mit XLink ist es möglich, einen Multilink zu definieren, der unseren Anforderungen genügt. Abbildung 3.3 zeigt einen solchen Link (siehe Anhang für die zugehörige DTD). Das Element `multilink` ist von Typ `extended`. Das Element `starting_ressource` ist vom Typ `resource` und ist selbst die Startressource. Die Elemente `original` und `alternative`

[36] D.h. Elemente vom Typ `resource` definieren selbst eine Ressource innerhalb des extendes Links.

sind vom Typ `locator` und zeigen auf den urpsrünglich verlinkten Anker und die jeweiligen Alternativen. Die Elemente `outbound` und `related` sind vom Typ `arc` und definieren die Beziehungen zwischen den einzelnen Elementen.

```
<multilink>
  <starting_resource>Apache Web server</starting_resource>
  <original xlink:href="http://www.apache.org/">
    <title>Apache Web server</title>
  </original>
  <alternative xlink:href="http://www.mysql.com/">
    <title>MySQL</title>
  </alternative>
  <alternative xlink:href="http://www.php.net/">
    <title>PHP: Hypertext Preprocessor</title>
  </alternative>
  <outbound/>
  <related/>
</multilink>
```

Abbildung 3.3: Ein Multilink in XLink definiert

Abbildung 3.4 zeigt den oben definierten Link als Diagramm. Hier ist zu sehen, dass die einzelnen Elemente zusammen an einem Link partizipieren. Die geschwungenen, gerichteten Kanten sind durch die `arc`-Elemente definiert. `outbound` definiert dabei die Richtung der Links zwischen der Startressource und den Zielankern. `related` definiert die Beziehung zwischen den Zielankern[37]. Der ursprüngliche Anker ist allein durch die Semantik, den Namen des Elements `original` ausgezeichnet, es wäre jedoch denkbar, eine solche Auszeichnung zusätzlich entsprechend der XLink-Syntax vorzunehmen.

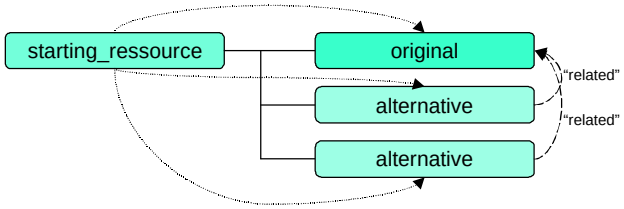


Abbildung 3.4: Graphische Darstellung des in XLink definierten Multilinks

Es ist sicherlich sinnvoller, XPointer [DeRose et.al. 2002] zu verwenden, um die Startressource zu referenzieren, anstatt die Startressource wie oben beschrieben, in den Link einzubetten. In dieser Studienarbeit wurde XPointer nicht benutzt, da dies zu einer wesentlich

komplexeren Implementation geführt hätte. Es ist wesentlich unkomplizierter, die Multilinks an den durch die usprünglichen Links markierten Positionen im Dokument einzufügen, als tatsächlich 3rd-Party Links zu verwenden, die aufwändig wieder in das Dokument eingefügt werden müssten. Dies würde jedoch zu einer saubereren Trennung von Ressourcen und Links führen, und sollte bei einer Weiterführung des Projekts auf jeden Fall in Betracht gezogen werden.

OHIF

[Grønbæk et.al. 2000] beschreiben das *Open Hypermedia Interchange Format* (OHIF), das ein reicheres Linkmodell als XLink bietet. Guided Tours und Hierarchien von Link-Typen werden beispielsweise unterstützt. [Christensen & Hansen 2002] untersuchen, wie OHIF auf XLink abgebildet werden kann. Für zukünftige Weiterführungen dieses Projekts wäre es sicherlich ebenfalls interessant, die Verwendung von OHIF-Links zu untersuchen.

3.4 Protokolle zur Übertragung von Link-Objekten

Der Webserver soll verschiedene Möglichkeiten bieten, die Link-Objekte an den Client zu liefern. So soll analog zu dem Vorschlag von [Wilde 2002] der Webserver XML-Dokumente liefern können, die die Links für eines seiner Dokumente enthält. Er soll aber auch wie oben beschrieben in der Lage sein, die Links für den Transport an den Client in die HTML-Dokumente einzubetten[38].

HTTP

Im HTTP-Standard gibt es drei verschiedene Möglichkeiten, eine Menge von zusammengehörigen Dokumenten (beispielsweise ein HTML-Dokument und die zugehörigen Bilder und Stylesheets) herunterzuladen. Die erste Möglichkeit bietet der HTTP/1.0-Standard [Berners-Lee et.al. 1996]. Hier wird für jedes Dokument eine neue TCP/IP-Verbindung geöffnet, die Dokumente werden nebenläufig heruntergeladen. Werden nun das HTML-Dokument und das Dokument, das die Link-Objekte enthält, nebenläufig heruntergeladen, so kann es passieren, dass den BenutzerInnen im Browser schon der Text angeboten wird, die Links, die von diesem Text ausgehen jedoch noch nicht heruntergeladen wurden. Dies kann beispielsweise ein nicht funktionierendes Site-Navigationsmenü zur Folge haben, für die BenutzerInnen sicher-

[37] Diese Beziehung, eine „arcrole“, ist nicht, wie in Abbildung 3.4 verkürzt angegeben „related“, sondern „http://www.scone.de/xlink/related“, da der XLink-Standard hierfür eindeutige URLs vorsieht.
[38] Ein Protokoll für den Zugriff auf Linkserver wird in dieser Studienarbeit nicht thematisiert. Siehe [Wilde 2002b] für eine ausführliche Diskussion.

lich ein unakzeptabler Zustand. Der Client muss also in der Lage sein, die Anzeige des HTML-Dokuments und der dazugehörigen Links zu synchronisieren. HTTP/1.1 [Fielding et. al. 1999] bietet persistente Verbindungen, hierbei können Objekte nacheinander innerhalb derselben TCP/IP-Verbindung heruntergeladen werden. Es ist auch möglich, mehrere Requests abzusetzen, ohne auf die Response der vorigen Requests warten zu müssen, die Dokumente werden jedoch auch hier nacheinander geliefert. Auch hier muss der Client wieder für eine synchrone Darstellung sorgen. Werden die Links auf Serverseite direkt in das Dokument eingefügt, entfällt die Notwendigkeit der Synchronisation. Die in Abschnitt 4.5 beschriebenen Implementationen weisen daher keine Mechanismen zur Synchronisation auf.

Die oben beschriebenen zwei Möglichkeiten, Links von einem Webserver zu beziehen, sollten durch kleine Erweiterungen des HTTP-Protokolls ermöglicht werden. Das HTTP-Protokoll ist ein einfaches Request-Response Protokoll. Die Nachrichten haben einen Header und einen Daten-Teil. Der Header beginnt mit einer Zeile, die eine Methode, eine URI und die Protokoll-Version benennt. Danach können Feldname:Wert-Zeilen folgen, die den Request genauer spezifizieren. Als Methoden sind `GET`, `POST`, `HEAD`, `PUT` und `DELETE` vorgesehen, wobei die beiden letzteren von den meisten Webservern nicht angeboten werden. `GET` fordert das unter der URI zu findende Dokument an, `POST` tut dies auch, weist aber explizit darauf hin, dass der Request einen Daten-Teil besitzt. `HEAD` liefert nur den HTTP-Response-Header, nicht aber das Dokument zurück. Die Feldname:Wert-Zeilen können noch zusätzliche Einschränkungen definieren, wie „Lade das Dokument nur herunter, wenn es jünger als x ist.“ oder „Ich akzeptiere die folgenden Dokument-Typen...“.

HTTP für 3rd Party Links

Um dem Server zu signalisieren, dass er Link-Objekte in das angeforderte Dokument einbinden soll, genügt es, einen neuen Feldnamen einzuführen: `Accept-Links`. Als Wert sollte dann eine Spezifikation der gewünschten Links übergeben werden. Denkbar ist beispielsweise Werte wie „`*`“ für alle in der Linkbase des Servers vorhandenen Links, „`role = related`“, eventuell kombiniert mit Pattern-Matching oder „author“ für die Links von der AutorIn der Ressource. „`none`“ würde das Einbinden von Links unterbinden. Die Definition einer geeigneten Abfragesprache ist leider zu komplex für diese Studienarbeit. Abbildung 3.5 ist ein Beispiel für einen HTTP-Request, der den Webserver dazu veranlässt, die oben definierten Multilinks in das Dokument „mydocument.xml“ einzubinden.

```
GET http://www.example.org/mydocument.xml HTTP/1.0
Accept-Links: role=http://www.scone.de/xlink/related
```

Abbildung 3.5: Die Verwendung des Accept-Links Feldes

Um die Kompatibilität mit älteren (also heutigen) Browsern zu gewährleisten, sollten `GET` und `POST` beim Fehlen des `Accept-Links` Feldes wie bisher funktionieren. Nur wenn der Wert des `Accept-Linkroles`-Feldes explizit auf „none“ gesetzt ist, sollte ein Dokument ohne eingebettete Links zurückgeliefert werden[39].

Analog zum bisherigen HTTP-Standard könnte es außerdem eine weitere Methode `LINKS` geben, die ein Dokument mit Link-Objekten zu einem Dokument anfordert. Über `Accept-Links` könnte dabei spezifiziert werden, welcher Art diese Links sein sollen.

```
LINKS http://www.example.org/mydocument.xml HTTP/1.0
Accept-Links: role=http://www.scone.de/xlink/related
```

Abbildung 3.6: Ein möglicher Request mit der Links-Methode

Abbildung 3.6 ist ein Beispiel für einen solchen HTTP-Request. Da hierbei keine Rücksicht auf ältere Clients genommen werden muss, ist bei der `LINK`-Methode die Verwendung von SOAP möglich. Praktisch kann dieser Request auch sofort an den mit dem Webserver assoziierten Linkserver gestellt werden. Hierfür wird dann das entsprechende Protokoll benutzt.

SOAP

Alternativ wäre die Verwendung von SOAP [Box et.al. 2000] möglich. SOAP benutzt HTTP als Trägerprotokoll und definiert, wie ein XML-Dokument im Datenteil der HTTP-Requests und -Responses beispielsweise als entfernter Prozeduraufruf interpretiert werden kann. [Wilde 2002b] favorisiert SOAP, um auf Linkserver zuzugreifen, da so eine wesentlich höhere Ausdrucksfähigkeit zur Verfügung steht, als durch die Verwendung neuer Feldname:Wert-Tupel im HTTP-Header. Um die Kompatibilität mit älteren Browsern und die Verwendung von POST-Requests zu gewährleisten, wird in dieser Arbeit jedoch das HTTP-Protokoll erweitert.

[39] Tatsächlich bedeutet dies, dass die Webserver für die „alte“ Verwendung von `GET` und `POST` einfache Links wieder als HTML-Links in die Dokumente einfügen müssen, um den herkömmlichen Standard zu emulieren.

3.5 Visualisierungen von multiplen Links

Die Visualisierung von multiplen Links ist ein zentraler Teil dieser Studienarbeit. Ohne eine gute Visualisierung werden Multilinks in der Praxis nicht einzusetzen sein. [Kahn et.al. 1995] beschreiben die ihrer Meinung nach drei fundamentalen Elemente der „visuellen Rhetorik im Hypertext“: *link presence*, *link destination* und *link mapping*.

Link Presence

Wichtig ist demnach zuerst die Kenntlichmachung von Links, die in dem gerade visualisierten Knoten präsent sind. Als Beispiel ziehen die Autoren hierfür das Hypertextsystem *Storyspace* heran: Dieses System gab den LeserInnen zunächst keine Hinweise, ob Links in einem Knoten präsent sind; erst wenn sich der Mauszeiger über einer der unsichtbaren Linkpräsenzen befand, wurde eine Interaktionsmöglichkeit visualisiert. Einige HypertetxtautorInnen veranlasste dies dazu, in ihren Storyspace-Texten Links beispielsweise durch fette Schrift sichtbar zu machen. Siehe [Kahn et.al. 1995] und [Weinreich et.al. 2001] für eine Übersicht von verschiedenen historischen bzw. möglichen zukünftigen Visualisierungen von Linkpräsenzen. In den Visualisierungen von HTML wurden Links[40] ursprünglich durch Unterstreichungen von Text und durch Rahmen um Bilder kenntlich gemacht. Heute gibt es durch die Einführung von CSS keine durchgängig einheitliche Kennzeichnung mehr. So ist es unter anderem möglich – und auf einigen Websites auch Realität – dass Links erst sichtbar werden, wenn sich der Mauszeiger über der Startressource befindet[41]. [Oberholzer & Wilde 2002] reichern HTML-Dokumente mit multiplen Links an. Die Präsenz eines solchen multiplen Links wird durch ein kleines Icon neben dem ursprünglichen Link visualisiert (siehe Abbildung 3.9). Für unser Projekt ist eine solche Kennzeichnung nicht notwendig, da alle eingebetteten Links zu multiplen Links erweitert werden. Desweiteren ist es bei [Oberholzer & Wilde 2002] notwendig, auf dieses Icon zu klicken, bevor die Alternativen visualisiert werden. [Cooper 1995, S. 172] unterscheidet zwischen *revenue tasks*, die direkt dazu beitragen, Probleme zu lösen, und *excise tasks*, die nicht direkt zur Lösung dieser Probleme beitragen, aber trotzdem ausgeführt werden müssen. Der Klick auf das Icon ist ein solcher excise task, er schafft zusätzlichen Aufwand – vor allem, wenn es darum geht, bei mehreren Links auf einer Seite schnell die Alternativen Ziele zu vergleichen – und sollte weggelassen werden.

[40] oder treffender „Startressourcen“ nach der in dieser Arbeit verwandten Notation.
[41] Dieser Umstand wird in der weiter unten beschriebenen Visualisierung von multiplen Links tatsächlich ausgenutzt. In diesem Fall ist dies jedoch unproblematisch, da dies in einem Kontext geschieht, der sichtbar macht, dass es sich um Links handelt.

Die Implementierung dieses Projekts wird daher die Visualisierungen der jeweiligen Linkpräsenzen nicht verändern. Dies erhält zusätzlich das Seitenlayout der veränderten Dokumente.

Link Destination

Das nächste wichtige Element der visuellen Rhetorik ist das Ziel eines Links. Den LeserInnen sollte deutlich und prägnant visualisiert werden, wohin der Link sie führen kann. Nützliche Informationen in dieser Hinsicht können URLs, Linktitel oder so elaborierte Popups wie bei HyperScout [Weinreich & Lamersdorf 2000] sein. Handelt es sich bei einem Link um einen Link mit mehreren möglichen Zielen, sollten diese Ziele ebenfalls sichtbar sein. *COOL Links* [Miller & Waltz 1998] beispielsweise verzichten auf eine Visualisierung der unterschiedlichen Alternativen eines multiplen Links. Hier entscheidet ein Algorithmus anhand der Vorlieben der BenutzerInnen und den an den Links annotierten Meta-Informationen automatisch, welcher der alternativen Links traversiert werden soll. Hierdurch können Linkziele von LeserIn zu LeserIn oder in der Zeit differieren, ein möglicherweise verwirrender Ansatz[42]. Die Visualisierung von multiplen Zielen variiert stark von System zu System. [Landow & Kahn 1992] beschreiben die Visualisierung von multiplen Links in *Intermedia* und *WorldView*. WorldView unterstützt keine multiplen Links, daher werden multiple Ziele in sogenannten *Crossroads Dokumenten* dargestellt (siehe Abbildung 3.7). Ebenso geht der PortalMaximizer vor [Oberholzer & Wilde 2002]. Dies ist ein Bruch der Navigation, und reisst die Visualisierung der Ziele eines multiplen Links aus ihrem Kontext.

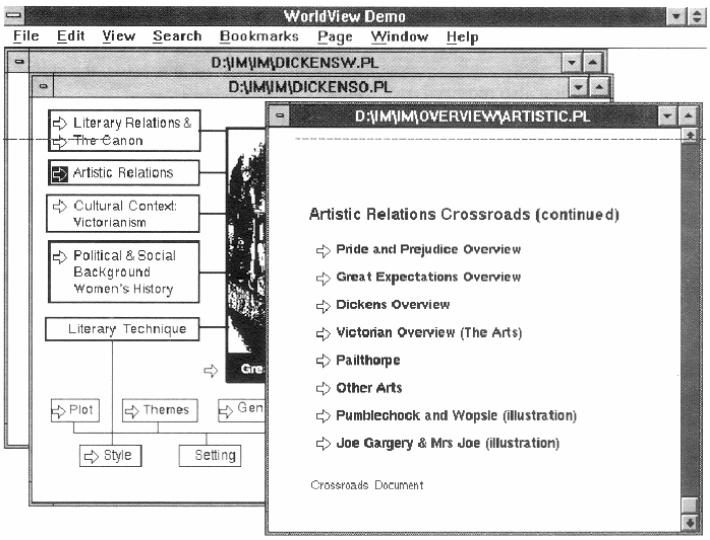


Abbildung 3.7: Ein Crossroads Dokument in WorldView (aus [Landow & Kahn 1992])

Ein besserer Ansatz ist in Intermedia zu sehen. Hier wird zu einem multiplen Link ein Dialog geöffnet, der die möglichen Ziele auflistet (siehe Abbildung 3.8).

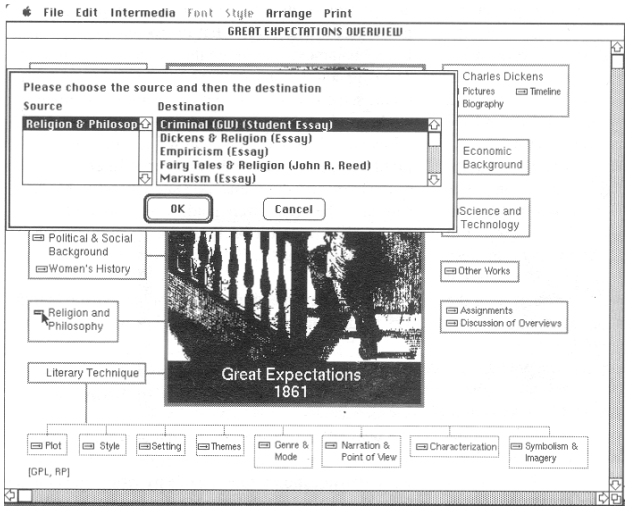


Abbildung 3.8: Visualisierung eines multiplen Links in Intermedia
(aus [Landow & Kahn 1992], Ausschnitt)

Hierdurch werden die möglichen Ziele im Kontext der Startressource dargestellt, sehr benutzungsfreundlich ist dieses Interface jedoch nicht. So ist die Startressource („Religion and Philosophy“) offensichtlich nicht besonders gekennzeichnet. Wird der Mauszeiger zu dem Dialogfenster mit den möglichen Zielen bewegt, gibt es keine visuelle Orientierungshilfe, wo sich die Startressource befindet. Desweiteren befindet sich das Dialogfenster in einiger Entfernung von der Startressource und bietet ein verwirrendes Interface für einen 1-zu-n Link.

[Oberholzer & Wilde 2002] stellen reichhaltige, hierachisch aufgebaute multiple Links dar. Die möglichen Ziele werden wie ein Kontextmenü in der Nähe der betreffenden Startressource dargestellt (siehe Abbildung 3.9).

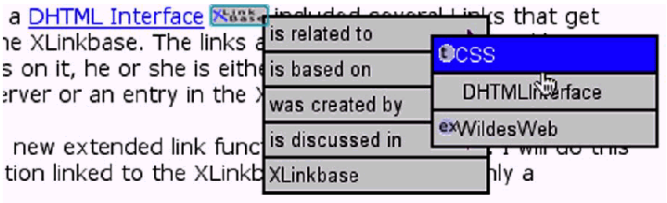


Abbildung 3.9: Visualisierung eines multiplen Links in [Oberholzer & Wilde 2002]

Allerdings ist diese Lösung noch keineswegs optimal. Wie schon kritisiert sind die Menüs nicht direkt über den Link, sondern über einen kleinen, neben dem Link eingefügten Button zu erreichen. Das Kontextmenü selbst scheint unabhängig über dem Text zu schweben, die Verbindung zur Startressource, die nicht speziell als „ausgewählt“ gekennzeichnet ist, wird

[42] Dies ist die subjektive Meinung des Autors dieser Arbeit, der noch niemals mit der COOL Links Umgebung gearbeitet hat. Er ist jedoch der Meinung, dass den BenutzerInnen mehr Entscheidungsfreiheiten eingeräumt und nicht gleich wieder genommen werden sollten.

nur durch die offensichtliche Nähe hergestellt. Dies wird bei der Visualisierung der hierarchischen Menüstruktur konsequent weiterverfolgt, so dass auf den ersten Blick nicht ersichtlich ist, dass das gerade dargestellte Untermenü zu „is related to“ gehört. Die Tiefe der Menühierarchie wird auf zwei Ebenen beschränkt, um die mentale Belastung der AnwenderInnen nicht unnötig zu erhöhen.

Abbildung 3.10 zeigt ein Kontextmenü in der Textverarbeitung Microsoft Word 2000. Zum zuvor automatisch korrigierten Wort „das“ erscheint ein Button, der, wenn er angeklickt wird, ein Kontextmenü öffnet. Hier wird das Kontextmenü nicht nur in der räumlichen Nähe der jeweiligen „Startressource“ dargestellt, sondern auch durch die Ausrichtung klar zugeordnet. Störend wirkt hier nur der zwischengeschaltete Button.

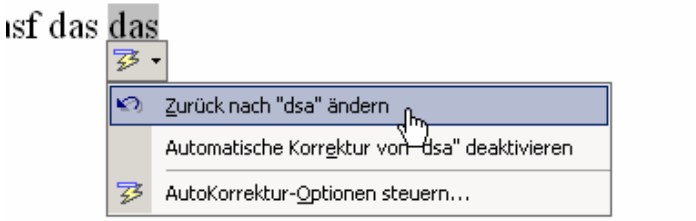


Abbildung 3.10: Ein Kontextmenü in Word 2000

Abbildung 4.13 (auf Seite 28) zeigt eine mögliche Visualisierung eines multiplen Links, die aufgrund der oben aufgeführten Überlegungen und Kritiken entworfen wurde. Diese Visualisierung wird sofort sichtbar, wenn sich der Mauszeiger über dem betreffenden Link befindet. Ein Klick auf den Link ist hierzu nicht notwendig[43]. Von der üblichen Visualisierung eines Kontextmenüs wurde Abstand genommen und stattdessen eine Form gewählt, die aus dem Menüleisten mit ihren Untermenüs bekannt ist. So soll die enge Beziehung zwischen der Startessource und den multiplen Zielen verdeutlicht werden. Allerdings wurde in diesem Fall die Funktionalität einer Menüleiste nicht vollständig übernommen. Die Startressource ist hier nicht der Titel des aufgeklappten Submenüs, sondern ein funktional gleichwertiges Item, d.h. es kann ebenso wie die anderen Einträge als Ziel ausgewählt werden. Um den textuellen Kontext der Startressource (die in dem hier beschriebenen Projekt eine Sonderrolle als Repräsentant des ursprünglichen und des multiplen Links einnimmt) zu erhalten und nicht durch das Submenü zu verdecken, wird es wie ein Menütitel dargestellt. Durch Tests mit BenutzerInnen müsste festgestellt werden, ob diese Art der Darstellung verwirrend ist, dies kann im begrenzten Rahmen einer Studienarbeit jedoch leider nicht erfolgen.

[43] Dies würde wiederum dazu führen, dass zwei Mausklicks benötigt werden, um einem Link zu folgen. Dies wäre nicht nur die Einführung eines excise tasks, sondern würde zusätzlich eine vorher sehr einfache Routinetätigkeit erschweren. [Johnson 2000] weist darauf hin, dass gerade Routinetätigkeiten so einfach wie möglich gehalten werden müssen (S. 36).

Jedem möglichen Alternativziel wird in der oben beschriebenen Visualisierung die Art der Assoziation zum ursprünglich verlinkten Dokument als Text voransgestellt. In diesem Fall ist dies „related“. In diesem Fall wurde kein prägnantes Icon gewählt, da es sehr viele verschiedene mögliche Arten von Beziehungen gibt. Ein Text kann hier im Zweifel aussagekräftiger und auch erweiterbarer sein. Andererseits mag es wenig sinnvoll sein, in diesem Fall die Beziehung „related“ zu visualisieren, da der zugehörige Prototyp keine anderen Beziehungen kennt. Auch dies sollte letztendlich durch Tests mit BenutzerInnen geklärt werden.

Link Mapping

Als drittes fundamentales Element der visuellen Rhetorik des Hypertexts zählen [Kahn et.al. 1995] das Link Mapping auf. Darunter verstehen sie eine flächige Repräsentation der Verbindungen des Texts als ein Graph. Eine solche Übersichtsdarstellung halten sie für eine grundlegend wichtige Unterstützung der LeserInnen, unabhängig davon, ob sie vollständig absorbiert oder einen bestimmten Pfad entlang geführt werden sollen. Ein solcher Übersichtsgraph kann jedoch selbst sehr schnell unübersichtlich werden. Abbildung 3.11 zeigt, wie das System Storyspace ein gesamtes Web als ein Dokument anzeigt.

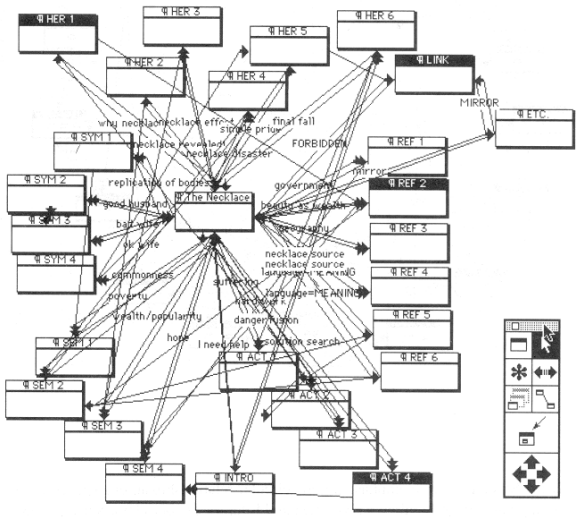


Abbildung 3.11: Die Ansicht eines Webs in *Storyspace*
(aus [Kahn et.al. 1995])

Es sind nur relativ wenige Knoten zu sehen, doch der Graph ist schon jetzt sehr unübersichtlich. Eine graphische Ansicht des gesamten WWW zu erstellen ist sicher noch weniger hilfreich. Hier würde sich eher der Ansatz von Intermedia anbieten. Das Intermedia System zeigt zu jedem Knoten ein „Web View“ an (siehe Abbildung 3.12).

Abbildung 3.12: Ein *Intermedia* Web View
(aus [Landow & Kahn 1992], Ausschnitt)

Der Web View kombiniert eine navigierbare History und eine navigierbare Liste der über den aktuellen Knoten erreichbaren Knoten [Utting & Yankelovich 1989]. Die Art der angezeigten Knoten wird dabei durch ein Icon visualisiert. Wird ein Link im aktuellen Knoten selektiert, werden die über diesen Link erreichbaren Knoten durch dicke Linien im Web View hervorgehoben. Abbildung 3.12 zeigt also ein Beispiel für die Visualisierung eines multiplen Links.

Der Versuch, multiple Links zu visualisieren, kann unter dem Gesichtspunkt des Link Mappings gesehen werden. Indem bei Bedarf die möglichen Ziele eines multiplen Links visualisiert werden, wird ein Teil eines Umgebungsgraphen visualisiert. Konsequent wietergedacht bedeutet dies, dass möglicherweise statt wie oben beschrieben eines Untermenüs mit ähnlichen Zielknoten ein Graph aufklappt, der den Zielknoten in eine Linkumgebung einbettet. Abbildung 4.10 (auf Seite 28) zeigt einen solchen Graphen, der über Inlinks verbundene, ähnliche und über Outlinks verbundene Knoten visualisiert. Der Graph ist dabei intuitiv „von links nach rechts“ zu lesen: Die Knoten in der linken Spalte sind über Inlinks zu erreichen, die Knoten in der mittleren Spalte Links zu ähnlichen Knoten und die Knoten in der rechten Spalte die über Outlinks zu erreichenden.

Durch die graphische Struktur wird eine geschachtelte Hierarchie wie in [Oberholzer & Wilde 2002] vermieden. Leider wird dadurch auch wesentlich mehr Platz auf dem Bildschirm eingenommen. Wie in Abbildung 4.10 ersichtlich, führt dies dazu, dass die angezeigten Dokumententitel bis zur Unkenntlichkeit gekürzt werden müssen. Auch die eindeutige Assoziation mit dem ursprünglichen Zielknoten des Links (hier der Link „Apache Web Server“ und der ausgezeichnete Knoten „Welcome! The Apac“) ist noch nicht gelungen. Um einen zusätzlichen Cognitive Overhead zu vermeiden, beschränkt sich die endgültige Implementation der Studienarbeit allerdings auf die in Abbildung 4.13 dargestellte Version, die Visualisierung als

Graph wurde daher auch nicht weiter verbessert. [Cockburn & Jones 1997] beschreiben einige ältere Beispiele von Link Mappings für das World Wide Web. Browsing Icons [Mayer & Bederson 2001] ist eine ausgeklügelte neuere Variante. Abbildung 3.13 zeigt einen „Browse Ahead Graph“ der Browsing Icons Oberfläche, der die momentane History zusammen mit früheren Pfaden (in der Graphik hellgrau) graphisch darstellt und auch zur Navigation genutzt werden kann.

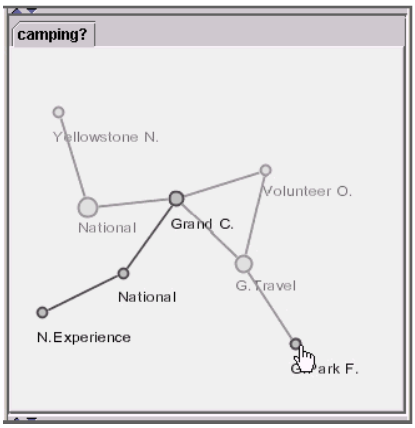


Abbildung 3.13: Ein Browse Ahead Graph in Browsing Icons
(aus [Mayer & Bederson 2001], Ausschnitt)

Menüs

Die Repräsentation von multiplen Zielen eines Links als ein Menü mag elegant erscheinen, wirft allerdings auch einige Fragen auf. So merken [Oberholzer & Wilde 2002] an, dass ihr Kontextmenü eigentlich gar kein Kontextmenü sei. Es biete nicht unterschiedliche Aktionen, sondern nur eine Aktion (einen Link zu verfolgen) und unterschiedliche Wahlmöglichkeiten (die verschiedenen Linkziele). Sie gehen jedoch davon aus, dass diese Visualisierung den BenutzerInnen „intuitiv klar“ (ebd.) erscheinen würde. Das selbe Problem gilt auch für die in dieser Studienarbeit vorgeschlagene Visualisierung. Zusätzlich wird hier die übliche Semantik eines Menüs durchbrochen, indem ein Menüeintrag als Titel visualisiert wird, aber trotzdem wählbar ist[44]. Sollte sich herausstellen, dass die Visualisierung eines multiplen Links als Menü missverständlich ist, könnte eventuell durch die Kennzeichnung der einzelnen Menüeinträge als Links für Klarheit sorgen (siehe Abbildung 3.14). In dieser weniger eleganten Darstellung sind die multiplen Ziele durch Unterstreichung eindeutig als Links identifizierbar.

[44] [Cooper 1995] führt Beispiele an, die belegen sollen, dass Konsistenz für User Interfaces nicht unbedingt notwendig ist, und auch inkonsistente Lösungen zu guten Ergebnissen führen können (S. 377). Es ist also durchaus denkbar, dass diese inkonsistente Anwendung eines UI-Idioms durchaus sehr gut benutzbar ist. Allerdings gibt [Shneiderman 1998] Konsistenz als erste seiner „acht goldenen Regeln des Interface Designs“ an (S. 74).

Abbildung 3.14: Visualisierung als alternative Links

Im Vergleich zu den in [Eberts 1994, S. 549] diskutiert Eigenschaften von Menüs werden noch andere Unterschiede zu herkömmlichen Menüs deutlich. So kann Redundanz zwischen Untermenüs zu Verwirrung führen und sollte daher vermieden werden, in unserem Fall jedoch kann Redundanz erwüscht sein und sinnvolle Informationen liefern. [Eberts 1994, S. 548ff] diskutiert auch die Frage von Breite und Tiefe eines Menüs und kommt zu dem Schluss, dass Menüs mit mittlerer Breite und Tiefe von BenutzerInnen bevorzugt werden und schneller benutzt werden können. Betrachten wir analog zur Menüleiste das HTML-Dokument und die Visualisierungen der multiplen Links als Untermenüs, so sehen wir, dass die Breite/Tiefe-Relation von Dokument zu Dokument erheblich schwanken kann. Aufgrund der oben genannten Unterschiede zwischen Menüs und multiplen Links muss hier allerdings die Frage gestellt werden, ob hier ein Vergleich überhaupt angestellt werden darf. [Eberts 1994, S. 551] geht des weiteren darauf ein, dass die Beschränkungen des Kurzzeitgedächtnisses um drei bis 9 Informationsbrocken liegen[45]. Demnach sollten nicht mehr als 9 Menüeinträge gleichzeitig auf dem Bildschirm zu sehen sein. Konsequent weitergedacht, bedeutet dies, dass nicht mehr als 9 Links für ein Dokument visualisiert werden sollten. Die meisten HTML-Dokumente beinhalten jedoch schon deutlich mehr Links, und durch die Visualisierung mutlipler Links wird die Anforderung an die LeserInnen zusätzlich erhöht. Es wird durch Tests zu bewerten sein, ob das Visualisieren von multiplen Links zur kognitiven Überforderung der LeserInnen führen wird.

[45] Dies stimmt beispielsweise mit den Beobachtungen von [Miller 1956] überein.

4 Multiple Links in der Praxis

Zur Erprobung der oben diskutierten Techniken und Konzepte wurden im Laufe dieser Studienarbeit verschiedene Prototypen entwickelt, die auf dem vom Autoren dieser Arbeit mit entwickelten Framework Scone aufbauen. In diesem Rahmen sollte gleichzeitig erprobt werden, wie Scone mit multiplen Links arbeiten kann und wie Scone für den Einsatz auf Serverseite optimiert werden kann. Die Daten für die Multilinks wurden über die Google API angefordert, die für diese Zwecke in Scone integriert wurde.

4.1 Scone

Scone[46] ist ein in Java entwickeltes Framework zur schnellen Entwicklung von Prototypen von WWW – Navigationswerkzeugen ([Weinreich et.al. 2003]). Die Grundidee des Frameworks ist es, die Darstellung der HTML-Dokumente weiterhin dem Browser zu überlassen und zusätzliche Hilfen entweder in die Dokumente einzubringen oder in einer externen Applikation zugänglich zu machen. Mit Hilfe eines Proxys klinkt Scone sich in die Datenströme des Browsers ein, ist so über die Aktionen der Anwendenden informiert und kann die Inhalte der vom Browser angeforderten Dokumente manipulieren. Weitere Komponenten von Scone sind eine persistente Datenbasis, ein elaboriertes UserTracking, ein Remote Access Service und ein konfigurierbarer Robot (siehe Abbildung 4.1). Zusätzlich bietet Scone seinen Plugins komfortable Konfigurationsmöglichkeiten.

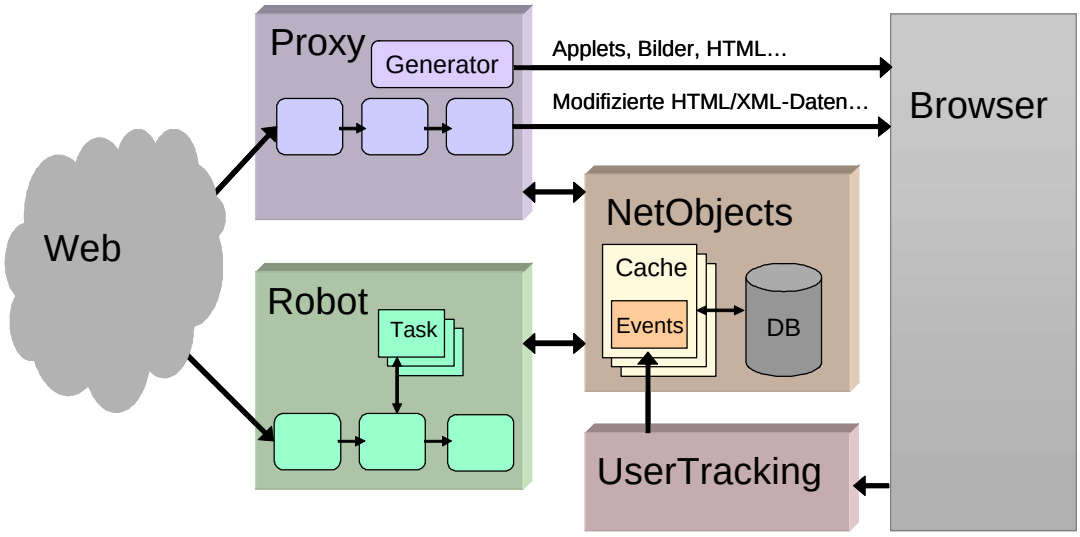


Abbildung 4.1: Die Kernkomponenten von Scone
(aus [Weinreich et.al. 2003])

[46] http://www.scone.de

Zielsetzung

Das Framework wurde entsprechend den folgenden Anforderungen entwickelt:

- **Prototypen sollten schnell und einfach zu entwickeln sein.**

  Dies ist durch Scones elegante Java API gelöst. Java ist eine mittlerweile weitverbreitete und leicht zu handhabende Sprache. Scones API geht auf die Bedürfnisse von Navigationswerkzeugen ein und bietet oft benötigte Funktionalität. Gängige Browsertypen (Internet Explorer und Netscape in Form von Mozilla) bieten zwar eine API, um eigenen Code (zumindest in Form von Plugins) zu integrieren. Dies ist jedoch weder ein schneller noch ein einfacher Ansatz.

- **Prototypen sollten auf möglichst vielen Plattformen getestet werden können.**

  Scone im Proxybetrieb hat den Vorteil, dass es mit jedem beliebigen Browser auf jedem beliebigen Betriebssystem zusammen arbeiten kann[47]. Ein Prototyp als Browsererweiterung hat den Nachteil, dass er nur auf einem bestimmten Betriebssystem funktioniert und außerdem ein anderer Browser (oder zumindest ein anderes Programm) als gewohnt benutzt werden muss. Die aktuelle Version gestattet es den Anwendenden, ihre gewohnten Browser zu benutzen, der Kreis der möglichen Testpersonen wird dadurch erheblich erweitert.

- **Prototypen sollten möglichst viel Kontrolle über den Browser haben.**

  In Bezug auf dieses Kriterium sind in die Browser integrierte Lösungen eindeutig besser. Sie könnten eine wesentlich bessere Kontrolle über die Funktionalität des Browsers haben und jede noch so winzige Aktion der Anwendenden beobachten. Scone ist inzwischen jedoch so weit ausgereift, dass hier nur noch wenige relevante Unterschiede bestehen. Navigationserkzeugprototypen können problemlos die Navigationsleiste des Browsers verdekken und eine eigene Navigationsleiste anzeigen. Die wichtigsten Aktionen der Anwendenden können ebenfalls beobachtet werden.

- **Scone sollte möglichst flexibel (verteilt) eingesetzt werden können.**

  Scone soll nicht nur Einzel- sondern auch Workgroup- oder noch grössere Anwendungen unterstützen. Scone als Proxy hat den Vorteil, dass es selbst nicht an die Rechner der Benutzenden gebunden ist. Es kann überall laufen und von überallher angesprochen werden.

[47] Nicht alle Basisfunktionalitäten von Scone arbeiten mit allen Browsern zusammen. Das UserTracking beispielsweise benötigt JavaScript- und Java-Unterstützung des Browsers. Mit den gängigsten Browsern wie dem Internet Explorer, Netscape und Opera funktioniert es jedoch.

Auch die benutzte Datenbank muss nicht auf dem selben Rechner liegen wie Scone. So kann Scone nicht nur (ein bisschen) skaliert sondern auch je nach Anwendung mehr oder weniger verteilt eingesetzt werden. Ein weiterer Vorteil ist hier, dass Anwendende Scone für Tests nicht selbst zu installieren brauchen, sondern ein entferntes Scone benutzen können. Dieser Vorteil wird in dieser Arbeit ausgenutzt.

- **Scone sollte auf bewährten Techniken aufsetzen.**

  Die Darstellung der HTML-Dokumente sollte weiterhin dem Browser überlassen sein. Selbst wenn wir in einer selbst entwickelten Lösung die Darstellung von HTML erreichen würden, so müssten zusätzlich noch JavaScript, Java Applets und diverse andere Anwendungen wie Flash unterstützt werden. Ein völlig neu entwickeltes WWW-Navigationswerkzeug fällt somit in den Bereich des Unmöglichen. Ist es für die Navigation hilfreich, von der ursprünglichen Darstellung der Dokumente im Browser abzuweichen, zum Beispiel um Links anders zu visualisieren [Weinreich et.al. 2001], reicht es (zumindest für einen Prototypen) aus, DHTML zu verwenden. So behalten die dargestellten Dokumente ihr bekanntes Aussehen, und Scone und die Navigationsprototypen können sich auf die Fragen der Navigation beschränken.

Der Proxy

Eine wichtige Komponente von Scone ist ein HTTP-Proxy (siehe [Fielding et. al. 1999], S. 9), der u.a. dazu benutzt wird, die Visualisierung von HTML-Dokumenten im Browser zu beeinflussen, obwohl hierdurch nicht der Grad von Kontrolle über das User Interface möglich ist, wie dies beispielsweise durch die Entwicklung eines eigenen Browsers oder durch die Erweiterung eines bestehenden Browsers möglich wäre. Dies kann zudem zu Performanzverlusten führen, bietet aber so wertvolle Vorteile, dass wir einen Proxy für die wesentlich bessere und vielfältigere Wahl halten. Scone benutzt WBI als Grundlage für seinen Proxy, um sich in die Datenströme des Browsers einzuklinken. WBI ist ein erweiterbarer Proxy vom IBM Almaden Research Center, dessen Monitor-Editor-Generator (kurz: MEG) Konzept das Überwachen und Verändern der HTTP-Ströme erlaubt[48] [Maglio & Barrett 2000].

Monitore überwachen und analysieren Daten, Generatoren erzeugen Daten, und Editoren vereinen beide Funktionalitäten in sich. MEGs können leicht programmiert werden und

[48] WBI kann nicht nur auf HTTP-Ströme zugreifen, in der zu Forschungszwecken frei erhältlichen Version befindet sich jedoch nur ein HTTP-Sublayer.

klinken sich als Intermediaries[49] in den Datenstrom des HTTP-Klienten ein. Das WBI-Framework sorgt für den reibungslosen Ablauf und transportiert die Daten des Stroms byteweise zwischen den MEGs.

Scone läuft bei Bedarf als ein WBI-Plugin, das Scone-Plugins Zugriff auf die Datenströme des Browsers gewährt. Dazu müssen die Scone-Plugins MEGs zur Verfügung stellen. Ein standardmässig arbeitendes MEG von Scone – das UserTracking – protokolliert die Aktionen der BenutzerIn, und stellt die damit verbundenen Daten als persistente Objekte zur Verfügung. Auf Wunsch werden auch HTML-Dokumente analysiert und Werte wie die Sprache, Anzahl der Links etc. in diesen Objekten gespeichert.

Es kommt oft vor, dass mehrere MEGs, die an einem Markup-Dokument im Datenstrom interessiert sind, das Dokument parsen, um zum Beispiel auf einem DOM zu arbeiten. Nach getaner Arbeit muss das Dokument wieder in einen Bytestrom umgewandelt werden, damit WBI es an das nächste MEG weitergeben kann. Dies ist ineffizient und durch Scones Token-Stream-Konzept inzwischen behoben.

Das TokenStream-Konzept

Die der ursprünglichen Implementation zugrundeliegende Version von WBI erlaubte nur den Transport von Bytes zwischen den MEGs. Arbeiten zwei oder mehr MEGs auf höheren Datenstrukturen, muss jedes dieser MEGs zunächst die Bytes in höhere Datenstrukturen parsen. Dann muss es die Daten wieder in Bytes umwandeln, um sie an das nächste MEG weiterreichen zu können. Können hingegen auch Objekte zwischen MEGs transportiert werden, wird teures wiederholtes Parsen vermieden.

Sind beispielsweise zwei benachbarte MEGs A und B im Datenstrom, die beide auf einem DOM arbeiten wollen, so kann A die Bytes in ein DOM-Objekt parsen und nach der Manipulation an B weitergeben. MEG C, das danach ausgeführt wird, erwartet aber einen Bytestrom, also muss das Objekt wieder in einen Bytestrom umgewandelt werden. In diesem Szenario ist einmal Parsen eingespart worden.

Es standen zwei mögliche Ansätze für die Implementation zur Verfügung. Der erste betrachtet einen Strom von Objekten (beispielsweise Markup-Elemente) zwischen MEGs, der zweite geht davon aus, daß nur ein Objekt zwischen MEGs ausgetauscht werden soll, zum Beispiel ein DOM. Beide Ansätze sind gleich mächtig. Wir entschieden uns für den zweiten Ansatz, so

[49] Intermediaries sind Komponenten, die sich in Datenströme einklinken und diese gegebenenfalls verändern, siehe [Maglio & Barrett 2000].

daß nun ein MEG statt eines Bytestroms auch ein Objekt anfordern und zurückgeben kann: das MegObject. Soll damit ein Objekt-Strom implementiert werden, wird einfach der Strom als dieses MegObjekt übergeben.

Die folgenden Bedingungen mußten erfüllt werden, um das TokenStream-Konzept in das WBI-Framework zu integrieren:

- Kompatibilität: alte MEGs sollten ohne Veränderung weiterbenutzt werden können
- Konsistenz: MegObjects sollen nur an kompatible MEGs weitergegeben werden
- Transparenz: es soll für MEGs verborgen sein, in welchem Format die benachbarten MEGs den Datenstrom bearbeiten.

Die Aufgaben zur Realisierung des Konzepts können in zwei Bereiche getrennt werden: Die Aufgaben des Frameworks und die Aufgaben der MegObjects. Das WBI-Framework soll MEGs erlauben, entweder einen Bytestrom oder ein MegObject als Ausgabe zu benutzen. Ausserdem sollen MEGs als Eingabe einen Bytestrom oder – wenn vorhanden – stattdessen auch ein MegObject wählen können. Das Framework selbst deckt damit also die Anforderung der Kompabilität ab. Für Transparenz und Konsistenz müssen allerdings die MegObject-Implementationen selbst sorgen. Nur sie wissen, wie die Byteströme in geeignete Repräsentationen umgewandelt werden können, und nur sie wissen, wie daraus wieder Byteströme erzeugt werden können. Sie verdecken auch die Funktionalität des Frameworks:

Für eine Subklasse von MegObject wird eine Klasse implementiert, die aus einem RequestEvent[50] die Daten in Bytestrom- oder MegObject-Form entnimmt und in ein MegObject vom gewünschten Typ umwandelt. Ebenso gibt es eine Klasse, die das MegObject dann auf Wunsch als Ausgabe eines MEGs akzeptiert und an das RequestEvent weiterleitet. Arbeitet ein MEG beispielsweise auf einem Strom von XML-Elementen – sogenannten Token – sähe die Implementation foglendermaßen aus:

Ein `XMLTokenInputStream` würde das Einlesen der Daten aus dem RequestEvent kapseln. Er wäre in der Lage, Token aus einem Bytestrom zu erzeugen, oder weitergereichte Token zu akzeptieren. Die Token wären in diesem Fall XML-Tags oder Text. Der MEG würde dann auf den Token arbeiten, und sie an einen `XMLTokenOutputStream` weiterreichen. Dieser `XMLTokenOutputStream` würde die Token dann an einen Puffer weiterreichen. Dieser Puffer ist das MegObject, das an das Framework zurückgegeben wird.

[50] Ein RequestEvent im WBI-Framework kapselt die HTTP-Request und –Response-Header sowie die Request- und Response-Daten. Die Response-Daten können dabei entweder als Bytestrom oder als MegObject vorliegen.

Der Einfachheit halber ist der `XMLTokenOutputStream` meist selbst das MegObject. Siehe Abbildung 4.2 für eine schematische Übersicht.

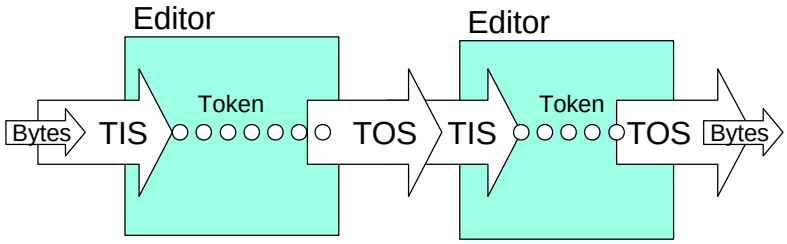


Abbildung 4.2: Zwei Editoren arbeiten auf einem XMLTokenStream
TIS: XMLTokenInputStream, TOS: XMLTokenOutputStream

Das MegObject-Konzept wurde während eines Praktikums im IBM Almaden Research Center im Sommer 2000 vom Autoren dieser Arbeit in WBI integriert.

Das Datenmodell

Grundlegend für die Funktionalität von Scone ist das objektorientierte Datenmodell (siehe Abbildung 4.3). Ein Plugin soll auf alle relevanten Objekte in der Scone-Domäne zugreifen können. Außerdem soll die Information über ein Objekt nicht verloren gehen, sondern persistent und wie bei einer Datenbank abfragbar sein.

Die von uns als relevant ausgezeichneten Objekte sind

- **Server**
  Enthält die wichtigsten Daten zu einem Server (Besitzer, mittlere Zugriffszeit,...)
- **NetNode**
  Repräsentiert eine über eine URI adressierbare Ressource und das letzte Anfrageergebnis an diese URI (HTTP-Status-Code, MimeType,...)
- **HtmlDocument**
  Repräsentiert ein HTML – Dokument, und enthält allgemeine Angaben über Inhalt und Format (Anzahl der Bilder, Sprache, Titel,...)
- **Link**
  Ein gerichteter, typisierter Verweis zwischen zwei NetNodes, wie er in HTML beschrieben werden kann.
- **Inclusion**
  Hat im Prinzip dieselbe Struktur wie ein Link, der Zielknoten (zum Beispiel ein Bild) wird jedoch in den Ausgangsknoten eingebunden.
- **Keyword**

Ordnet Schlüsselwörter Netzknoten zu (diese können bei HTML beispielsweise aus einem Meta-Tag entnommen werden)

- **Access**

  Repräsentiert den Zugriff eines Users auf einen Netzknoten.

- **User, Person**

  Repräsentieren Menschen, die Scone benutzen.

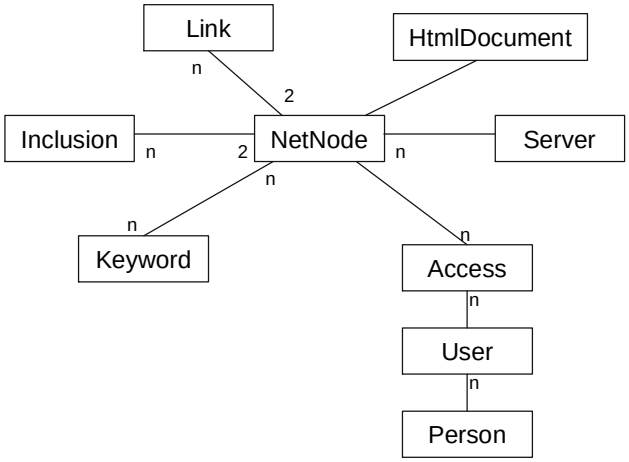


Abbildung 4.3: Das Datenmodell von Scone

Implementation des Datenmodells

Die oben beschriebenen Objekte, die NetObjects, werden vom Scone Framework erzeugt, gefüllt und transparent persistent gehalten. Jede der NetObject-Klassen hat ihren speziellen Container, der diese Transparenz durch einheitlichen Zugriff auf im Hauptspeicher und auf der Festplatte vorhandene Instanzen herstellt.

Diese Container sind Caches, die eine theoretisch unbegrenzte Menge von Objekten aufnehmen können. Durch die Verwendung der Klassen `java.lang.ref.WeakReference` bzw. `java.lang.ref.SoftReference`[51] wird dafür gesorgt, dass momentan nicht benutzte NetObjects aus dem Speicher entfernt werden können. So wird der Hauptspeicher nicht unnötig belastet, und ein unnötiges Überlaufen des Speichers wird vermieden.

[51] Werden Objekte in Java nur noch über diese Klassen referenziert (sind also nicht mehr über normale Variablennamen erreichbar), so kann der Garbage Collector des Java Runtime Environments diese Objekte aus dem Speicher entfernen, wenn neuer Speicherplatz benötigt wird. Die beiden Klassen unterscheiden sich darin, wie schnell die Objekte aus dem Speicher entfernt werden.

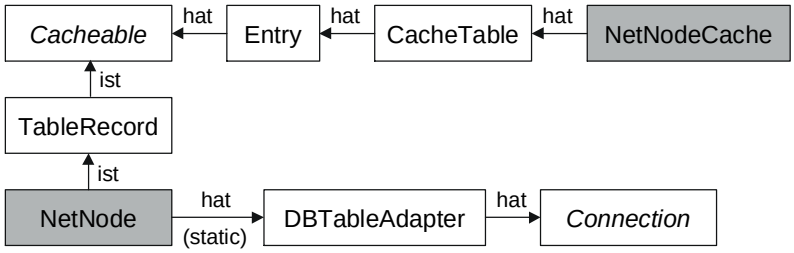


Abbildung 4.4: Klassendiagramm am Beispiel von NetNode und NetNodeCache

Dabei muss der Cache sicherstellen, dass die Inhalte eines Objekts persistent gemacht werden, bevor das Objekt aus dem Speicher entfernt wird. Zur persistenten Speicherung wird das relationale Datenbanksystem MySQL[52] benutzt. Die Klassen, die die NetObjects beschreiben, werden dazu auf SQL-Tabellen abgebildet. Referenzen zwischen NetObjects werden als Joins realisiert. Die Abbildung der Klasse auf die Datenbank geschieht in der Klasse DBTableAdapter. Jede NetObject-Klassendefinition besitzt eine DBTableAdapter – Klasse, in der die Definitionen der Felder gespeichert sind. Diese Definitionen werden zu Beginn einer Sitzung aus einer SQL-Datei gelesen, die auch benutzt wird, um die Datenbank in MySQL zu definieren. Gleichzeitig wird überprüft, ob diese Version dem in MySQL verwandten Datenbankschema entspricht. So wird sichergestellt, dass alle Komponenten auf denselben Definitionen arbeiten. Eine NetObject-Instanz kann sich dann mit Hilfe des DBTableAdapters in die Datenbank schreiben, bzw. seine Daten aus der Datenbank auslesen.

Das Klassendiagramm in Abbildung 4.4 zeigt die wichtigsten beteiligten Klassen am Beispiel der NetNodes. Die im Text nicht behandelten Klassen verwalten Verwaltungsinformationen, z.B. ob das Objekt ein aktuelles Bild in der Datenbank hat. Nur Kenntnisse der beiden grau hinterlegten Klassen sind nötig, um NetNodes zu benutzen.

NetObjects können über eine eindeutige Kennung vom Cache angefordert werden. NetNodes können beispielsweise über ihre URI angefordert werden. Kompliziertere Anfragen, die für viele Navigationshilfen erforderlich sind, können mit der momentanen Implementation nur durchgeführt werden, wenn das Kapselungskonzept umgangen wird, und SQL-Anfragen auf der eigentlich verdeckten Datenbankstruktur durchgeführt werden. Es existiert eine experimentelle Version von Scone, die auf dem objektorientierten Datenbanksystem POET aufbaut, um komplexere Anfragen zu erlauben [Stephan 2002].

[52] http://www.mysql.com

Events

Die Caches können nach dem Observable-Pattern Events erzeugen, wenn beispielsweise neue Objekte erstellt werden, oder aus der Datenbank gelesen werden. So können Plugins beispielsweise davon unterrichtet werden, daß gerade eine neue Seite besucht wurde, oder dass ein HTML-Dokument fertig geparst wurde.

RAS

Der Remote Access Service (RAS) von Scone ermöglicht die Kommunikation mit Scone über TCP/IP. Die RAS-Implementation ist speziell auf Applets ausgerichtet. Kommunizieren Applets, die in ein HTML-Dokument eingebunden sind, mit Scone, so kann das User Interface (in diesem Fall also die dargestellte Webseite) dynamisch reagieren. Beispielsweise können veraltete Informationen aktualisiert werden, ohne dass die ganze Seite neu geladen werden muss. Oder Scone kann aktiv Befehle an den Browser schicken: mit einem reinem HTTP-Klienten kann Scone erst aktiv werden, wenn es ein Request empfangen hat. RAS wird somit in verschiedenen Scone-Anwendungen eingesetzt. In Hyperscout etwa dient es dazu, die Inhalte der Popups aktuell herunterzuladen. Im UserTracking Paket ist sogar eine Browser-Fernsteuerung integriert.

Eine RAS-Verbindung wird von einem Client aufgebaut, bei dem es vorkommen kann, dass er über lange Zeit nicht aktiv ist. Hierzu kann der Client die Verbindung „einschlafen“ und „aufwachen“ lassen, um Ressourcen zu sparen. Wird beispielsweise ein Applet in einem Browser dargestellt und anschließend eine neue Seite dargestellt, so wird das Applet nicht beendet und aus dem Speicher entfernt. Es wird nur benachrichtigt, dass es nicht mehr angezeigt wird, ist aber noch aktiv. Wird die Seite, die das Applet enthält, wieder angezeigt, wird das Applet wieder benachrichtigt. Ein solches Applet könnte in der Zeit, in der es nicht angezeigt wird, die RAS-Verbindung einschlafen lassen, und danach wieder aufwecken. Die aufgeweckte RAS-Verbindung benutzt zwar eine neue Socket-Verbindung, dies wird jedoch für das Applet gekapselt. Es ist sogar möglich, eine zustandsbehaftete RAS-Verbundung über mehrere Socket-Verbindungen hinweg zu benutzen.

Beim Verbindungsaufbau gibt der Client den Klassennamen eines ConnectionHandlers an. Dieser Handler wird auf Serverseite instanziiert, um mit dem Client zu kommunizieren. RAS bietet den beiden das Versenden und Empfangen von UTF-Zeichenketten. Die Connection-Handler definieren das Protokoll für ihre spezifische Anwendung. Grundsätzlich gibt es zwei

verschiedene Arten von RAS-Verbindungen bzw. ConnectionHandlern: Singleton und Dedicated.

Singleton-ConnectionHandler werden nur einmal instantiiert. Diese Instanz wickelt die Anfragen aller Clients in ihrer `handle()` - Methode ab. Es wird davon ausgegangen, dass die Methode vollständig durchlaufen wurde, wenn der Client die Verbindung beendet oder einschlafen lässt. Nach dem Aufwecken einer Verbindung wird die `handle()`-Methode erneut durchlaufen. Dieser Typ von ConnectionHandlern kann für nicht zustandsbehaftete Kommunikation verwandt werden, wie das Anfordern von Informationen über ein HTML-Dokument.

Dedicated-ConnectionHandler werden für jede neue Verbindung einmal instantiiert. Die Instanzen wickeln in ihren `handle()` – Methoden jeweils eine gesamte Verbindung ab. Es wird erwartet, dass genau dann die Verbindung vom Client endgültig beendet wird, wenn die `handle()` – Methode vollständig durchlaufen wurde. Dieser Typ von ConnectionHandler kann für zustandsbehaftete Kommunikation eingesetzt werden.

Robot

[Wollenweber 2002] beschreibt den in Scone integrierten programmierbaren Robot. Der Robot verwendet *Filter* und *Classifier*, um zu entscheiden, welche Dokumente als nächstes bearbeitet werden. Er benutzt ausserdem den DocumentParser von Scone, so dass die gewonnenen Daten nahtlos von Scone-Plugins genutzt werden können. Der Robot kann beispielsweise Links im Voraus traversieren, um den BenutzerInnen Informationen über die Zielknoten zu vermitteln. Dieser Ansatz wird mit dem HyperScout Prototypen [Weinreich & Lamersdorf 2000] verfolgt.

4.2 Repräsentation von Multilinks als NetObjects

Während der Implementation der Prototypen wurde deutlich, dass die Scone-Datenbank speziell auf HTML-Links zugeschnitten ist. Die Beziehungen zwischen dem ursprünglichen und den verwandten Dokumenten werden jeweils als gerichtete Links in der Datenbank persistiert. Das Attribut „`rel`“ wird dabei auf „`bowtie:related`“ gesetzt[53]. Dies ist im HTML-Standard [Raggett et.al. 1999] nicht vorgesehen[54]. Durch diese Vorgehensweise wer-

[53] Dieses Vorgehen ist insofern unsauber, als dass damit eventuell schon bestehende normale HTML-Links in der Datenbank überschrieben werden. Für die vorgestellten Prototypen ist dies unproblematisch, da sie nicht auf diesen Links arbeiten, für die Zukunft wäre eine sauberere Lösung vorzuziehen.

[54] Wird der Wert des Attributes „rel“ eines `LINK`-Tags auf „fontdef“ gesetzt, so zeigt der Link auf eine herunterladbare Schriftart (siehe http://www.truedoc.com). Es scheint also durchaus gängige Praxis zu sein, nicht dem Standard entsprechende Werte einzusetzen.

den zwar korrekte Links persistiert (sie sind – abgesehen vom Attribut – legal, und spiegeln einen richtigen Sachverhalt wieder). Ein echter Multilink wird so jedoch nicht korrekt auf die `NetObjects` abgebildet. Abbildung 4.5 a) ist ein Multilink mit Start-Anker „*“ und dem ausgezeichneten Zielanker „1“. „2“ und „3“ sind alternative Zielanker. Abbildung 4.5 b) stellt denselben Multilink als `NetObjects` persistiert dar. Jeder der drei Pfeile repräsentiert hier ein `Link` Objekt, die Kreise `NetNode` Objekte. Die „related“-Beziehung wurde wie oben beschrieben als Links zwischen „1“ und „2“ und „3“ realisert. Die Alternativen sind nun nicht mehr mit dem Start-Anker (-Knoten) verbunden, und es ist Aufgabe der jeweiligen Applikation, aus dieser Anordnung wieder einen Multilink zu konstruieren.

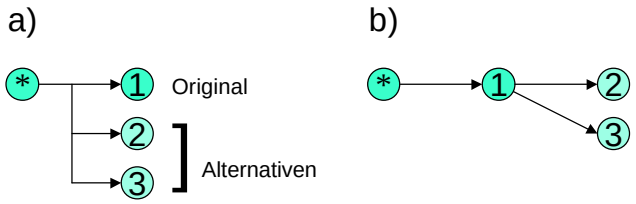


Abbildung 4.5: Die Persistierung von Multilinks als NetObjects

Durch das getrennte Speichern von diesen Links und den dazugehörigen Knoten wird Scone leicht zur Plattform einer Linkbase. Leider kennt Scone nur Links von Knoten zu Knoten, lediglich der Ziel-Anker in HTML-Schreibweise, etwa „#ziel_anker“, wird persistiert. Für ein OHS wird Scone daher in der gegenwärtigen Version nicht benutzt werden.

4.3 Das ServerSide - Plugin

Grundsätzlich ist der Einsatz von Scone auf Serverseite ohne Eingriffe in das Framework möglich. Beispielsweise kann Scone leicht unter der Adresse http://host2 als transformierte Kopie des Servers unter der Adresse http://host2 eingesetzt werden (siehe Abbildung 4.6).

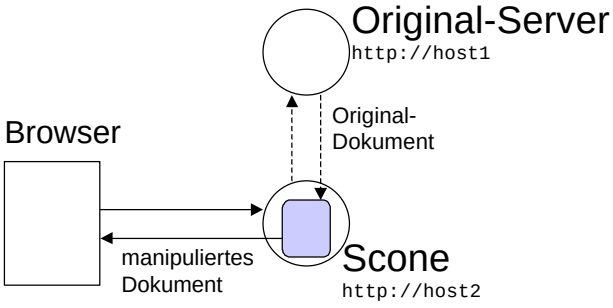


Abbildung 4.6: Scone als serverseitige Erweiterung

Benötigt wird hierfür nur ein Plugin, das das folgende gewährleistet:

- Ein `RewriteRequestEditor` muss die URL des Requests übersetzen. Für das oben genannte Beispiel müsste etwa http://host2/index.html in http://host1/index.html übersetzt werden.
- Ein RewriteResponseEditor muss die im Response-Document enthaltenen URLs in die entgegengesetzte Richtung übersetzen. Hierbei müsste analog zum obigen Beispiel http://host1/index.html nach http://host2/index.html übersetzt werden.

Ein Browser, der auf dem Server host2 surft, bekommt so die Dokumente von host1 geliefert, ohne dass dies für ihn ersichtlich wäre. Da die Dokumente durch Scone geladen werden, können sie von MEGs manipuliert werden. Somit ist es möglich, Erweiterungen auf Serverseite mit Scone zu testen.

Das Plugin, das die auf Serverseite zur Verfügung stehende Funktionalität bereitstellen soll, kann auf zwei mögliche Arten mit dem ServerSide Plugin kombiniert werden. Erstens kann dieses Plugin so eingesetzt werden, dass es auf den ursprünglichen Links des Servers host1 arbeitet, und zweitens kann es so eingesetzt werden, dass es auf den transformierten Links des Servers host2 arbeitet.

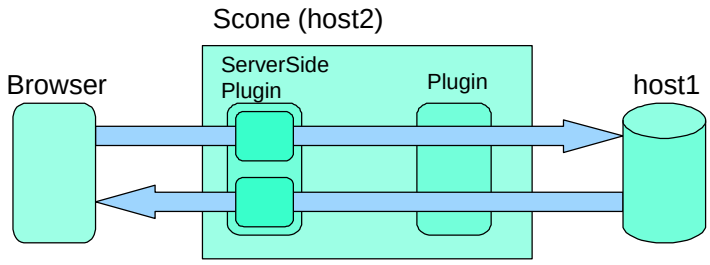


Abbildung 4.7: Das Plugin arbeitet auf host1.

Abbildung 4.7 illustriert den ersten Fall: Das ServerSide-Plugin wird so eingesetzt, dass das auf Serverseite einzusetzende Plugin auf den unbearbeiteten Links arbeitet. Es sieht die HTML-Dokumente so, wie sie von host1 heruntergeladen werden und manipuliert sie dementsprechend. Das ServerSide-Plugin übersetzt dann die vom Plugin manipulierten Seiten so, als kämen sie von host2. Entsprechend illustriert Abbildung 4.8 den zweiten Fall. Hier werden die in den Dokumenten eingebetteten Links zuerst vom ServerSide-Plugin manipuliert und dann erst vom eigentlichen Plugin.

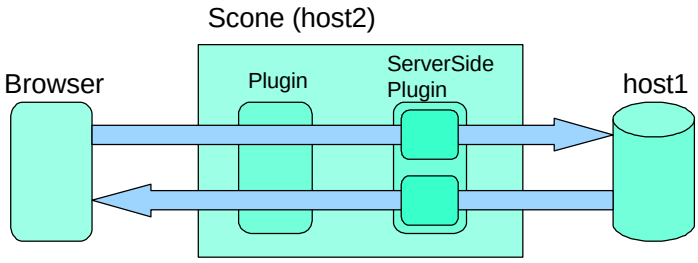


Abbildung 4.8: Das Plugin arbeitet auf host2

Die unkontrollierte Vermischung beider Varianten führt zu Inkonsistenzen, muss also vermieden werden. Welche Variante ausgewählt wird, hängt stark von vom jeweiligen Plugin ab. Für das in Abschnitt 4.5 beschriebene ServLink Plugin ist es sinnvoll es, auf den Links zu host1 zu arbeiten, da es nur so korrekte Anfragen an Google stellen kann (Google wird die Dokumente auf host2 höchstwahrscheinlich nicht kennen). Das Plugin bettet Links im XLink-Format in die Dokumente ein, die dann vom ServerSide-Plugin zurückübersetzt werden können, da es jegliche Tags automatisch nach dem Attribut `href` durchsucht. Würde das ServLink Plugin diese Links in JavaScript-Code einbetten, hätte das ServerSide-Plugin keine Möglichkeit, diese Links zurückzuübersetzen. Die ServerSide-Plugin-Funktionalität kann also nicht immer vollständig vor den serverseitigen Plugins versteckt werden.

Optimierungen

Bisher wurde Scone meist im Proxy-Modus eingesetzt. Hier stellte sich nicht die Frage, welche Daten nicht durch Scone geladen werden sollten. Grafiken und Klänge werden so durch den Proxy geladen, ohne dass sie transformiert werden (dies ist selbstverständlich auch denkbar, für WBI IBM-intern teilweise auch schon implementiert). Zur Verbesserung der Performanz wäre es besser, darauf zu verzichten, nicht zu transformierende Daten durch Scone zu laden. Setzen wir Scone auf Serverseite ein, haben wir die Möglichkeit, dies zu verhindern.

Dazu wird dem RewriteResponseEditor mitgeteilt, dass die URLs in bestimmten Tags wie `<img>` nicht zu kodieren seien[55]. Außerdem kann Scone auch noch eine relocation-Response versenden, wenn aus dem HTTP-Request ersichtlich ist, dass ein solches Dokument angefordert wird. So wird der Browser dazu veranlasst, nicht transformierbare Daten vom originalen Server zu laden. Dies ist noch nicht im ServerSide-Plugin implementiert, ist aber für die Zukunft angedacht.

[55] Der Encoder darf übrigens nicht einfach bestimmte Tags übergehen. Ist die Quelle eines Bildes z.B. durch eine relative URL angegeben, so muss diese in eine absolute URL umgewandelt werden, da sie sonst nicht mehr auf den originalen Server zeigt.

4.4 Die Google-API

In diesem Jahr hat Google[56] den kostenlosen Zugriff[57] auf seinen Suchdienst über eine SOAP-Schnittstelle zugelassen. Für die sogenannte „Google-API“ stellt Google unter http://www.google.com/api clientseitige Schnittstellen zur Verfügung, unter anderem auch eine Reihe von Java-Klassen, die für dieses Projekt verwandt wurden.

Die Google-API bietet grosse Teile der Funktionalität der Google-Suchmaschine. Es ist u.a. möglich, Suchbegriffe zu verwenden, Dokumente, die auf ein bestimmtes Dokument verlinken zu abzufragen, die Schreibweise eines Wortes zu überprüfen und verwandte Dokumente anzufordern. Damit kann Google eine Grundlage für die von [Bodner & Chignell 1999] beschriebene Verschmelzung der zwei Modi des Surfens darstellen (siehe Seite 13).

Für diese Studienarbeit wurde die Google API in der Klasse `scone.util.Google` gekapselt. Die Ergebnisse von Google werden hier in NetObjects umgewandelt und entsprechend wietergereicht. Der Zugriff auf Google ist ausreichend schnell, um eine akzeptable Benutzbarkeit der Prototypen zu gewährleisten.

4.5 Implementation der Prototypen

Zur Umsetzung und Erprobung der oben beschriebenen Ideen wurden zwei Prototypen jeweils als Scone-Plugins realisiert. Der erste Prototyp, das *Bowtie*-Plugin visualisiert komplexe Links in einem Java-Applet, während der *ServLink*-Protoyp auf Serverseite eingesetzt werden kann und sich auf die Visualisierung einfacher Multilinks beschränkt

Der Bowtie-Prototyp

Dieser Prototyp erprobt grundlegende Konzepte wie die Visualisierung und die Einbindung der Informationen von Google. Er experimentiert noch nicht mit Erweiterungen der Serverseite und Xlink, die Applet-Scone Kommunikation kann jedoch mit minimalen Änderungen schon als Erweiterung des Clients und des Servers aufgefasst werden.

[56] http://www.google.com
[57] beschränkt auf 1000 Abfragen pro Account pro Tag

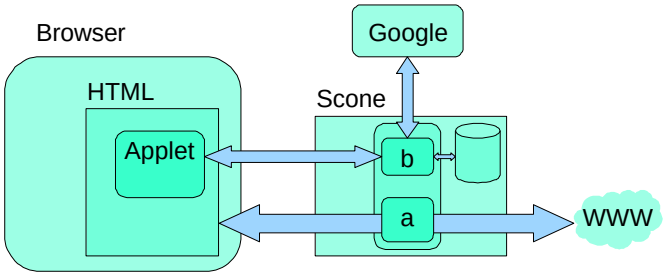


Abbildung 4.9: Die Architektur des Bowtie-Prototypens

Wie in Abbildung 4.9 ersichtlich, schaltet sich der Prototyp als ein Scone-Plugin im Proxybetrieb zwischen den Browser und das World Wide Web. „a“ ist ein `HtmlTokenEditor`, der ein (nicht sichtbares) Applet in jedes HTML-Dokument einfügt[58]. Zusätzlich werden JavaScript-Events zu den in das HTML-Dokument eingebetteten Links hinzugefügt, die über LiveScript mit dem Applet kommunizieren. Befindet sich der Mauszeiger über einem Link, wird dem Applet mitgeteilt, dass es ein entsprechendes Popup für diesen Link auf dem Bildschirm darstellen soll. Hierzu kommuniziert es über den RAS-Service mit „b“, einem `ConnectionHandler`. Diese Komponente fragt erst bei Scones Objektdatenbank nach, ob schon geeignete Dokumente bekannt sind. Ist dies nicht der Fall, wird über die Google API bei Google nach entsprechenden Dokumenten geforscht. „b“ liefert dann nach einem simplen Protokoll eine Reihe von Datensätzen an das Applet zurück. Diese Datensätze bestehen aus URL und Titel der betreffenden Dokumente und dem Namen der Beziehung zum ursprünglich verlinkten Dokument. Diese Daten werden dann wie in Abbildung 4.10 visualisert[59]. Das Applet ist somit in der Lage, über Inlinks oder Outlinks erreichbare und ähnliche Dokumente darzustellen. Die so dargestellten Dokumente können über das Applet selbstverständlich auch navigiert werden.

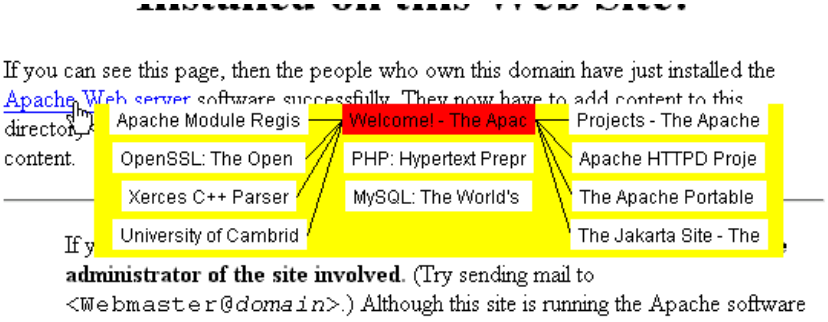


Abbildung 4.10: Ein Multipler Link als Graph visualisiert

Durch die mächtige Unterstützung des Frameworks Scone konzentrierte sich die Anstrengung der Implementation auf die Visualisierung des multiplen Links. Diesen Graphen zu visualisie-

[58] Das Applet und eine JavaScript-Code-Datei lädt der Browser dann von Scone herunter. Von diesem Vorgang wurde in Abbildung 4.9 abstrahiert.
[59] Von dieser Visualisierung stammt auch der Name „Bowtie“

ren stellte sich als schwieriger heraus, als erwartet. Da die betreffende Klasse in einem Browser ausgeführt werden sollte, standen neuere GUI-Konzepte wie Swing nicht zur Verfügung, auch musste auf einige Eigenheiten der älteren Java-Version eingegangen werden.

Der zweite Bowtie-Prototyp

Nachdem die ersten Erfahrungen mit dieser Art der Visualisierung zeigten, dass es anscheinend nicht sinnvoll ist, zu jedem Link einen so grossen Graphen zu visualisieren, wurde mit einer wesentlich beschränkteren Visualisierung experimentiert, wie sie in Abbildung 4.11 zu sehen ist.

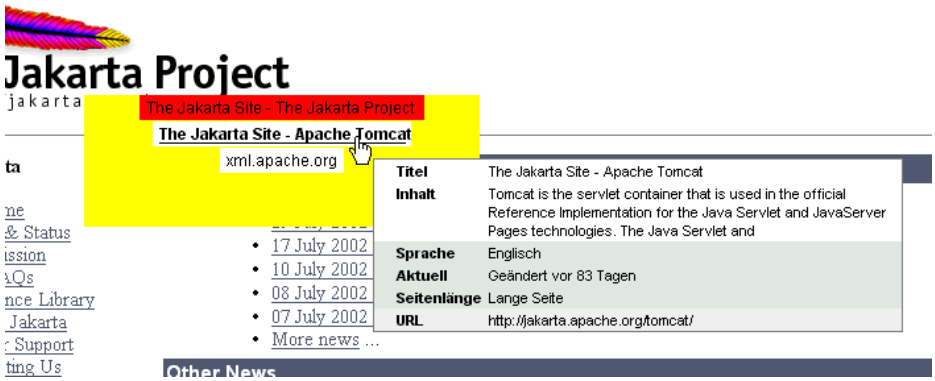


Abbildung 4.11: Die eingeschränkte Variante des Bowtie-Prototypen mit HyperScout

Diese Variante beschränkt sich auf verwandte Dokumente. Dies hat den Vorteil, dass die BenutzerInnen nicht mit zu vielen verschachtelten Informationen belastet werden. Zusätzlich konnten die Daten schneller visualisiert werden, da weniger Anfragen über das Netzwerk gestellt werden mussten. Schnell zeigte sich, dass dies eine sehr brauchbare Variante war. Ebenso schnell zeichnete sich ab, dass die im Popup dargestellten Dokumententitel nicht ausreichten, um die Zieldokumente klar zu beschreiben, da diese Dokumente nicht mehr unbedingt in den Kontext des ursprünglichen Links passten. Daher wurde probeweise eine angepasste Version von HyperScout benutzt, um zu diesen verwandten Dokumenten detailliertere Versionen anzuzeigen. Da HyperScout Informationen zu Links visualisiert und nicht, wie es hier benutzt wird, zu Dokumenten, musste dem HyperScout-Popup jeweils eine Link-Id übergeben werden. Diese Link-Id stammt von den wie in Abbildung 4.5 gezeigten zusätzlich in die Datenbank eingefügten Links.

Insgesamt ließ die Arbeit am Bowtie-Prototypen darauf schliessen, dass es eine interessante, möglicherweise wertvolle, Surferfahrung ist, Zugriff auf verwandte bzw. alternative Dokumente zu haben. Dabei zeichnete sich zudem ab, dass Informationen, wie sie HyperScout liefert, wertvolle Entscheidungshilfen bei der Orientierung und Navigation sind. Deutlich stellte

sich auch heraus, dass einige Informationen, die im Vorfeld als nützlich angesehen wurden, in der Praxis eher zu mehr Verwirrung und Overhead führten, als von Nutzen zu sein.

Der ServLink-Prototyp

Der ServLink-Prototyp erweitert nicht nur die Client- sondern auch die Serverseite, indem er die Aufgaben, die im Bowtei-Prototypen noch von einem einzigen Plugin übernommen wurden, nun auf zwei verschiedene Plugins aufteilt.

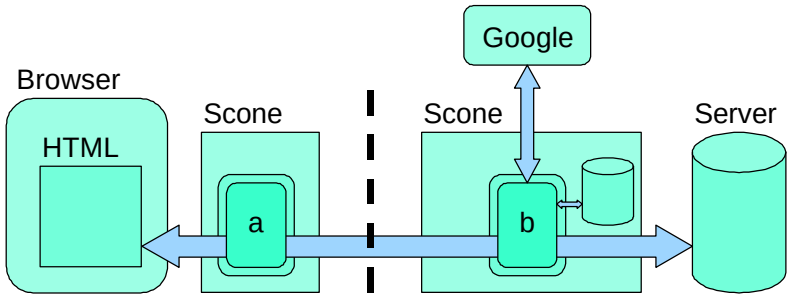


Abbildung 4.12: Die Architektur des ServLink-Prototypen

Wie in Abbildung 4.12 ersichtlich, emuliert eine clientseitige Scone-Instanz im Proxymodus einen Browser, der vom Server Multilinks anfordert und der AnwenderIn visualisiert. Eine serverseitige Scone-Instanz im Servermodus emuliert mit Hilfe des ServerSide Plugins einen Server, der die einfachen HTML-Links dementsprechend zu Multilinks erweitert.

Abbildung 4.13: Die Visualisierung eines multiplen Links im ServLink Prototypen

Das serverseitige Plugin „b“ klinkt sich dazu in den TokenStream ein und ersetzt A-Elemente durch `multilink`-Elemente (siehe Abschnitt 3.3). Die dazu notwendigen Daten holt sich das Plugin aus der Scone-Datenbank, oder über die Google-API direkt von Google. Die bei Google angefragten Informationen werden dann zum wiederholten Zugriff als `NetObjects` persistiert. Das clientseitige Plugin „a“ klinkt sich ebenso in den TokenStream ein und ersetzt `multilink`-Elemente durch JavaScript-Anweisungen, die Popups wie in Abbildung 4.13 generieren. Die Benutzbarkeit dieses Ansatzes war wesentlich besser als die des Bowtie-Plugins. Die Seiten bauten sich nun zwar langsamer auf, dafür konnten nun die Multilinks leichter erkundet werden.

Probleme

Die Positionierung der Menüs gestaltete sich als problematisch. Während es für absolut positionierte DHTML-Objekte problemlos möglich ist, die Position zu erfragen, ist es nicht einfach, die Position eines im Text eingebundenen Objekts zu erfragen. Nur im Internet Explorer war es dem Autoren bisher möglich, die Position eines Links zu erfragen, um das Menü wie in Abbildung 4.13 direkt an diesen Link anzuschliessen. Und auch hier kann es noch zu Fehlpositionierungen kommen, wenn sich der Link in einer Tabelle befindet. Die bisherigen Implementationen (siehe Abschnitt 5) positionieren daher ihre Popups in der Nähe des Mauszeigers, dies ist auf allen Plattformen unproblematisch.

Des weiteren stellte sich das Konzept des TokenStreams als unzureichend für dieses Plugin heraus. Da das Plugin auf `multilink`-*Elementen* arbeitet, mussten die Token bei Bedarf zunächst in Baumstrukturen geparst werden. Um Ressourcen zu schonen, wurden nur die `multilink`-Elemente in eine solche Baumstruktur geparst, nicht das komplette Dokument. Scones TokenStreams sollten eine Lösung zum automatisierten Parsen solcher partiellen Baumstrukturen bieten, um entpsrechende Anwendungen zu unterstützen.

5 Ähnliche Arbeiten

Die interessantesten ähnlichen Projekte stammen eindeutig aus der OHS-Community, so auch die unten vorgestellten. Diese Projekte sind sehr jung, teilweise so jung, dass sie dem Autoren erst kurz vor Abgabe dieser Studienarbeit zugänglich waren.

[Oberholzer & Wilde 2002] wollen das WWW in ein OHS transformieren. Hierzu verwenden sie XLinks und beschreiben die Implementation ihres XLinkbase-Servers und die verschiedenen möglichen Clients: Spezielle XLinkbase-Clients, XLink-fähige Clients und heutige WWW-Browser. Für die heutigen Browser sollen die Webserver die in der XLinkbase gespeicherten Links als DHTML-Menüs in die HTML-Dokumente einfügen. Zur Architektur ihres Ansatzes siehe Abbildung 3.1, eine DHTML-Visualisierung eines multiplen Links ist in Abbildung 3.9 zu sehen.

[Christensen & Hansen 2002] benutzen einen Python-basierten Web-Server, um OHIF-Links in XLinks umzuwandeln, und dann für die HTML-Browser in DHTML umzuwandeln. Sie versuchen die gesamte Breite von OHIF inklusive Multilinks und Guided Tours abzubilden. Abbildung 5.1 zeigt einen Screenshot ihres Xspect-Systems, das Guided Tours auf SVG-Imagemaps und Multilinks auf DHTML-Popups abbildet. Sie kommen in ihrer Arbeit zu dem Schluss, dass OHIF problemlos auf XLink abbildbar ist. Der Ansatz, XLinks zu DHTML zu konvertieren wird von ihnen zwar als einfach und zugleich mächtig eingestuft, in großen Anwendungen aber als zu mühsam angesehen. Wie [Oberholzer & Wilde 2002] hoffen sie auf XLink-fähige Clients.

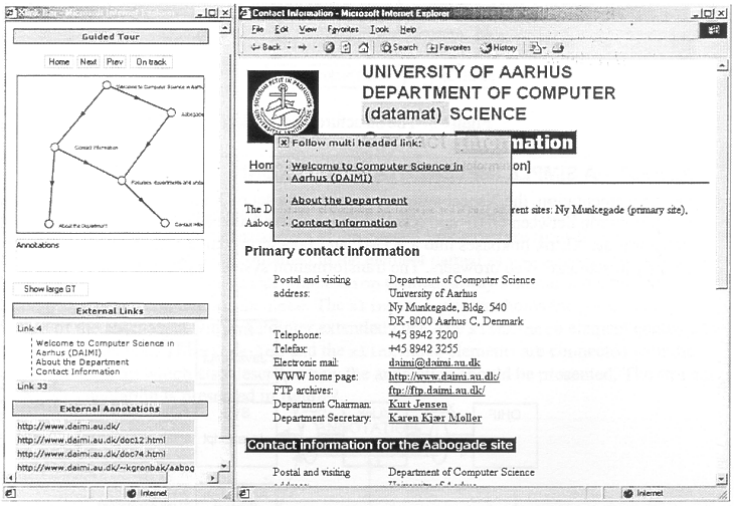


Abbildung 5.1: Eine Sitzung mit Xspect
(aus [Christensen & Hansen 2002])

[Martin & Ashman 2002] versuchen „high-level“ Linking (bidirektionale Links, n-äre Links und flexible Zielanker Spezifikation) durch „low-level“ Linking, wie es in HTML zu finden

ist (unidirektionale 1-zu-1 Links), zu emulieren. So kann ein bidirektionaler Link durch zwei unidirektionale ersetzt werden. Aus ähnlichen Gründen wie die Entwickler von Scone implementieren sie einen Proxy „Goate“, der es erlaubt, verschiedene Linksprachen als Plugins einzubinden. Goate parst die durch ihn geladenen HTML-Dokumente in eine Art DOM und erlaubt es den Plugins, „high-level“ Links in diese HTML-Dokumente einzubringen. Goate erstellt für diese Links eine DHTML- oder Text-basierte Repräsentation wie in Abbildung 5.2 zu sehen.

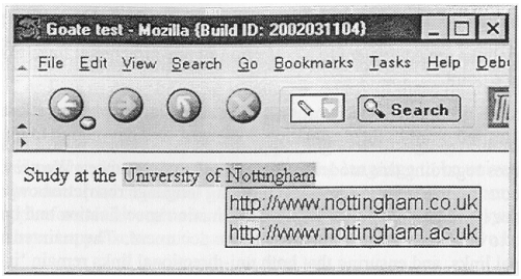


Abbildung 5.2: Ein durch Goate eingefügter Multilink
(aus [Martin & Ashman 2002])

Einen weniger eleganten Ansatz verfolgen [Lekova & De Troyer 2001]. Sie visualisieren extended XLink Links graphisch in Applets, die allerdings nicht in Popups sondern in eigenen Fenstern geöffnet werden[60] (siehe Abbildung 5.3). Als Hilfestellung bieten sie innerhalb des Applets kleine Popups, die die URL der Zielressource visualisieren.

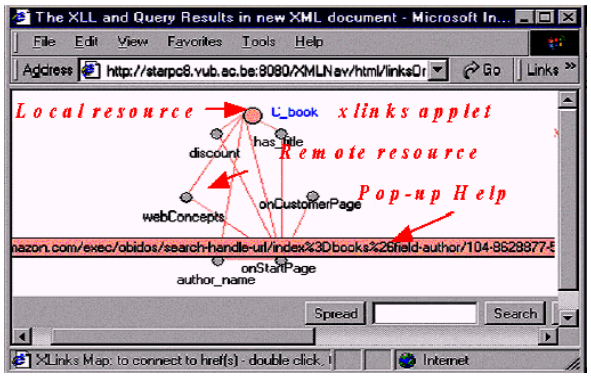


Abbildung 5.3: XLink-Visualisierung in [Lekova & De Troyer 2001]
(Ausschnitt)

[60] Dies erinnert an die in Abschnitt 3.5 beschriebenen Crossroads Dokumente.

6 Fazit

Im Laufe dieser Studienarbeit wurde untersucht, wie multiple und 3rd Party Links im World Wide Web eingesetzt werden können. Hierzu wurden Architektur, Protokolle und die Datenrepräsentation und Visualisierung von multiplen Links umrissen. Dies fand im Kontext der Forschung auf diesen Gebieten statt. Begleitend wurden experimentelle (Teil-) Implementationen der zuvor erörterten Konzepte realisiert.

Die Positionierung der Studienarbeit im aktuellen wissenschaftlichen Kontext zeigte die Aktualität des behandelten Themas. Besonders durch die Projekte aus dem OHS-Bereich (beispielsweise die in Abschnitt 5 vorgestellten) wird deutlich, dass die Integration von multiplen 3rd Party Links in das WWW zur Zeit von einer Reihe von verschiedenen Projekten erforscht wird. Trotz diverser Projekte, denen eine ähnliche Fragestellung wie dieser Studienarbeit zugrunde liegt, gelang es, eigene Ergebnisse zu erzielen. So wurde ein System erörtert, das es ermöglicht, extended XLink Links auf Serverseite in die Knoten einzubringen und auf Clientseite wieder zu extrahieren. Des weiteren wurde eine Reihe von Visualisierungen von multiplen Links experimentell erprobt und auch auf Grundlage anderer Arbeiten wurde eine verbesserte und vereinfachte Visualisierung entwickelt. Während sich die meisten anderen Projekte[61] mit einer Erweiterung des Webservers beschäftigten, um XLink-fähige Browser zu simulieren, wurde hier Server- und Clientseite erweitert.

Im Laufe dieser Studienarbeit wurde der serverseitige Einsatz von Scone weiter erprobt und Scones Möglichkeiten in Hinblick auf ein komplexeres Linkmodell untersucht.

Ausblick

Der Autor dieser Arbeit betrachtet es als sinnvoll, mit einem weiteren Projekt auf dem durch diese Studienarbeit erarbeiteten Stand aufzubauen.

Es ist sinnvoll, die Ergebnisse der OHS-Forschung noch stärker miteinzubeziehen. Hier bietet sich eine Erprobung und Einsatz des OHIF-Formats an. Interessant wäre auch eine Kombination aus dem in [Oberholzer & Wilde 2002] beschriebenen Linkserver mit dem in dieser Arbeit entwickelten Webserver entsprechend der in Abbildung 3.2 skizzierten Architektur. Auch der Einsatz von XPointer und eine Verfeinerung und Bewertung des beschriebenen Protokolls sollte untersucht werden.

[61] bis auf [Oberholzer & Wilde 2002], hier wird der Internet Explorer mit Hilfe einer COM-Komponente XLinkserver-fähig gemacht.

Vielversprechend ist sicherlich auch die Evaluation der verschiedenen möglichen Visualisierungen. Besonders interessant ist hier die Frage, ob es überhaupt sinnvoll ist, überaus komplexe Links zuzulassen, wie sie in einigen Projekten untersucht werden, oder ob dies die Benutzbarkeit nicht wieder einschränkt.

Die Erweiterung von Scone in Bezug auf multiple oder noch komplexere Links sollte zudem vorangetrieben werden, um Scone für den Einsatz in OHS-Forschungen interessant zu machen. Die Erweiterung der TokenStreams in Bezug auf baumförmig strukturierte Elemente steht zusätzlich an.

Auch die weitere Nutzung der Google API in Bezug auf dynamisch erstellte Links könnte lohnenswerts ein. Besonders die in [Bodner & Chignell 1999] beschriebene Verschmelzung von Suchen und Browsen könnte hiermit realisiert werden.

7 Literatur

[Anderson 1997] Anderson, K.: „Integrating Open Hypermedia Systems with the World Wide Web“, Proceedings of the eighth ACM conference on Hypertext, pages 157-166, Southampton United Kingdom, April 1997

[Anderson & Sherba 2001] Anderson, K. und Sherba, S.: „Using Open Hypermedia to Support Information Integration“ in Reich, S. et.al. (ed.): „Hypermedia: Openness, Structural Awareness and Adaptivity“, S. 8-16, Aarhus, 2001

[Ashman et.al. 1997] Ashman, H., Garrido, A. and Oinas-Kukkonen, H.: „Hand-made and Computed Links, Precomputed and Dynamic Links“, in Proceedings of Multimedia '97 (HIM '97), S. 191-208, 1997

[Berners-Lee et.al. 1994] Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H., Secret, A.: „The World Wide Web“ in Communications of the ACM 37 (8), S. 76-82, 1994

[Berners-Lee et.al. 1996] Berners-Lee, T., Fielding, R. und Frystyk, H.: „Hypertext Transfer Protocol -- HTTP/1.0“, RFC 1945, 1996

[Berners-Lee et.al. 1998] Berners-Lee, T., Fielding, R., Irvine, U., und Masinter, L.: „Uniform Resource Identifiers (URI): Generic Syntax“, RFC 2396, 1998

[Bieber et. al. 1997] Bieber, M., Vitali, F., Ashman, H., Balasubramaniam, V. und Oinas-Kokkonen, H.: „Forth Generation Hypermedia: Some Missing Links for the World Wide Web“ in International Journal of Human Computer Studies 47, 1997, S. 31-65

[Bodner & Chignell 1999] Bodner, Richard and Chignell, Mark: „Dynamic Hypertext: Querying and Linking“, ACM Computing Surveys 31(4), December 1999

[Bouvin 2000] Bouvin, N.: „Augmenting the Web through Open Hypermedia“ Ph.D. Thesis, Universität Aarhus, November 2000

[Bos et.al. 1998] Bos, B., Lie, H., Lilley, C. und Jacobs, I.: „Cascading Style Sheets, Level 2 CSS2 Specification“, World Wide Web Consortium Recommendation, Mai 1998, http://www.w3.org/TR/REC-CSS2/

[Box et.al. 2000] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Frystyk Nielsen, H., Thatte, S. und Winer, D.: „Simple Object Access Protocol (SOAP) 1.1“ World Wide Web Consortium Note, Mai 2000, http://www.w3.org/TR/SOAP

[Brailsford 1999] Brailsford, D. F.: „Separable hyperstructure and delayed link binding“, ACM Computing Surveys 31(4), December 1999, http://www.acm.org/surveys/

[Bray et.al. 2000] Bary, T., Paoli, J., Sperberg-McQueen, C.M. and Maler, E. (eds.): „Extensible Markup Language (XML) 1.0 (Second Edition)“, World Wide Web Consortium, Oktober 2000, http://www.w3.org/TR/REC-xml

[Brusilovsky 1997] Brusilovsky, P.: „Efficient Techniques for Adaptive Hypermedia“ in Nicholas, C., Mayfield, J.: „Intelligent Hypertext – Advanced Techniques for the World Wide Web“, Berlin 1997, S. 12-30

[Bush 1945] Bush, V.: „As We May Think“, Atlantic Monthly, Juli 1945

[Campbell & Goodman 1988] Campbell, B. und Goodman, J.M.: „HAM: A General Purpose Hypertext Abstract Machine.“ In: Communications of the ACM 31 (7), S.856-861 1988

[Christensen & Hansen 2002] Christensen, B. G. und Hansen, F. A.: „XLink – Linking the Web and Open Hypermedia“ in „Proceedings of the International Workshop on Open Hypermedia Systems at HT‘02“, 2002, S. 9-18

[Cockburn & Jones 1997] Cockburn, A. and Jones S.: „Design Issues for World Wide Web Navigation Visualisation Tools“. RIAO'97: The Fifth Conference on Computer-Assisted Research of Information. McGill University, Montreal, Quebec, Canada, 25th-27th June 1997. pages 55-74.

[Conklin 1987] Conklin, J.: „Hypertext: An Introduction And Survey“, IEEE Computer 20 (9), September 1987, S. 17-41

[Cooper 1995] Cooper, A.: „About Face: The Essentials of User Interface Design“, Foster City, CA, 1995

[Davis 1995] Davis H.: "To Embed or Not to Embed..." in Communications of the ACM (CACM), 38(8), 108-109, August 1995.

[Davis et.al. 1992] Davis, H., Hall, W., Heath, I. and Hill, G.: „Towards An Integrated Information Environment With Open Hypermedia Systems“ in „Proceedings of the ACM Conference on Hypertext“, 1992, S. 181-190

[DeRose et.al. 2000] DeRose, S., Maler, E. and Orchard, D. (eds.): „XML Linking Language (XLink)“, World Wide Web Consortium Proposed Recommendation, December 2000, http://www.w3.org/TR/xlink

[DeRose et.al. 2002] DeRose, S., Daniel, R., Grosso, P., Maler, E., Marsh, J. und Walsh, N.: „XML Pointer Language (XPointer)“ World Wide Web Consortium Working Draft, August 2002, http://www.w3.org/TR/xptr/

[De Roure et. al. 2000] De Roure, D. C., Walker, N. G. und Carr, L. A. : „Investigating Link Service Infrastructures“ in Proceedings of the 11th ACM Conference on Hypertext and Hypermedia, 2000, pp. 67-76

[Eberts 1994] Eberts, R. E.: „User Interface Design“, Englewood Cliffs, New Jersey, 1994

[Fielding et. al. 1999] Fielding, R, Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. und Berners-Lee, T.: „Hypertext Transfer Protocol -- HTTP/1.1“ veröffentlicht als RFC 2616, Juni 1999

[Grønbæk & Trigg 1996] Grønbæk, K. und Trigg, R.H.: „Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects“ in Proceedings of the Seventh ACM Conference on Hypertext 1996, S. 149-160

[Grønbæk et.al. 2000] Grønbæk, K., Sloth, L. und Bouvin, N.: „Open Hypermedia as User Controlled Meta Data for the Web“ in „Proceedings of the 9th International World Wide Web Conference“, S. 553-566, Amsterdam 2000

[Halasz 1988] Halasz, F. G.: „Reflections On NoteCards: Seven Issues For The Next Generation Of Hypermedia Systems“, CACM 31 (7), Juli 1988, S. 836-852

[Halasz & Schwartz 1994] Halasz, F.G., Schwartz, M. :“The Dexter Hypertext Reference Model“ in CACM 37 (2), S. 30-39

[Halsey & Anderson 2000] Halsey, B., Anderson, K.: „Xlink and Open Hypermedia Systems: A Preliminary Investigation“ in Proceedings of the 11th ACM Conference on Hypertext and Hypermedia, 2000, pp. 212-213

[Johnson 2000] Johnson, J.: „GUI Bloopers: Don’ts and Do’s for Software Developers and Web Designers“, San Francisco 2000

[Kahn et.al. 1995] Kahn, P., Peters, R. and Landow, G.P.: „Three Fundamental Elements of Visual Rhetoric in Hypertext“ in „Designing User Interfaces for Hypermedia“, Springer Verlag Berlin, S. 167-178

[Landow & Kahn 1992] Landow, G. P., Kahn, P.: „Where’s the Hypertext? The Dickens Web as a System-Independent Hypertext“ in „Proceedings of the ACM Conference on Hypertext“, 1992, S. 149-160

[Lekova & De Troyer 2001] Lekova, A. und De Troyer, O.: „Improved Navigation Through Extended XML Links“ in Proceedings of the Third International Conference on Information Integration and Web-based Applications & Services (IIWAS 2001), 10-12 September 2001, Linz Austria

[LinkAlarm 2002] LinkAlarm Inc.: “Web Integrity Benchmark”. Juni 2002, http://linkquality.com/

[Maglio & Barrett 2000] Maglio, P. und Barrett R.: “Intermediaries Personalize Information Streams” in CACM 43 (8), S. 96-101, 2000

[Martin & Ashman 2002] Martin, D. und Ashman, H.: "Goate: An Infrastructure for new Web linking" in „Proceedings of the International Workshop on Open Hypermedia Systems at HT'02", 2002, S. 19-25

[Mayer & Bederson 2001] Mayer, M., Bederson, B.: " Browsing Icons: A Task-Based Approach for a Visual Web History." Technical Report des HCIL (Human-Computer Interaction Lab, University of Maryland, USA), 2001

[McCarron et.al. 2002] McCarron,S. et.al.(eds.): „XHTML 2.0 Working Draft", World Wide Web Consortium, August 2002, http://www.w3.org/TR/2002/WD-xhtml2-20020805

[Miller 1956] Miller, G. A.: „The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information" in The Psychological Review, 1956, vol. 63, S. 81-97

[Miller & Waltz 1998] Michael Miller, L. Jay Wantz: „COOL Links: Ride the Wave." WWW7 / Computer Networks 30(1-7): 663-665 (1998)

[Müller-Prove 2002] Müller-Prove M.: „Vision and Reality of Hypertext and Graphical User Interfaces", Bericht 237 des Fachbereichs Informatik der Universität Hamburg, Februar 2002

[Nelson 1972] Nelson, T.: „As We Will Think", September 1972 in Nyce, J. und Kahn, P.: „From Memex to Hypertext – Vannevar Bush and the Mind's Machine", London, 1991 S. 245-260

[Nelson 1997] Nelson, T.: „Embedded Markup Considered Harmful", Oktober 1997 http://www.xml.com/pub/a/w3j/s3.nelson.html

[Nielsen 1995] Nielsen, J.: „Multimedia and Hypertext, The Internet and Beyond", London, 1995, AP Professional

[Nürnberg & Leggett 1997] Nürnberg, P. und Leggett, J.: „A Vision for Open Hypermedia Systems" Journal of Digital information, Volume 1, Issue 2 http://jodi.ecs.soton.ac.uk/

[Oberholzer & Wilde 2002] Oberholzer, G. and Wilde, E. : „Extended Link Visualization with DHTML:The Web as an Open Hypermedia System", TIK Report 125, January 2002

[Pam 1995] Pam, A.: „Where World Wide Web Went Wrong", Xanadu Australia, http://www.aus.xanadu.com/archive/6w-paper.html, 1995

[Pitkow & Jones 1996] Pitkow, J., Jones, R. : „Supporting the Web: A Distributed Hyperlink Database System", Proceedings of the Fifth International World Wide Web Conference, May 6-10, 1996, Paris, France

[Raggett et.al. 1999] Raggett, D., Le Hors, A. and Jacobs, I. (eds.): „HTML 4.01 Specification“, World Wide Web Consortium Recommendation, December 1999, http://www.w3.org/TR/html

[Shneiderman 1998] Shneiderman, B.: „Designing the User Interface – Strategies für Effective Human-Computer Interaction“, dritte Auflage, Reading, MA, 1998

[Stephan 2002] Stephan, B.: „Vergleich von Objektpersistenzmechanismen für Java: Integration der Object Server Suite von Poet in Scone“, Studienarbeit am Fachbereich Informatik der Universität Hamburg, http://vsis-www.informatik.uni-hamburg.de/publications/readstu.phtml/SA/165/Studienarbeit.pdf

[Tebbutt 1999] Tebbutt, J.: "User Evaluation of Automatically Generated Semantic Hypertext Links in a Heavily Used Procedural Manual", Information Processing and Management, 35(1) (1999), pp. 1-18.

[Utting & Yankelovich 1989] Utting, K und Yankelovich, N.: „Context and Orientation in Hypermedia Networks“ in ACM Transaction on Information Systems, 7 (1), S. 58-84, 1989

[Weinreich et.al. 2001] Weinreich, H., Obendorf, H., Lamersdorf, W.: „The Look of the Link - Concepts for the User Interface of Extended Hyperlinks“ in „Proceedings of the 12th ACM Conference on Hypertext and Hypermedia“, New York, 2001, S. 19-28

[Weinreich et.al. 2003] Weinreich, H., Buchmann, V. und Lamersdorf, W.: „Scone: Ein Framework zur evaluativen Realisierung von Erweiterungen des Webs“ in: Tagungsband zu Kommunikation in Verteilten Systemen 2003 (erscheint 2003)

[Weinreich & Lamersdorf 2000] Weinreich, H. und Lamersdorf W.: „Concepts for Improved Visualization of Web Link Attributes“ Computer Networks 33, S. 403-416, 2000

[Whitehead et.al. 1996] Whitehead, E.J., Jr., Fielding, R.T., Anderson, K.M. „Fusing WWW and Link Server Technology: One Approach“ in Proceedings of the 2nd Workshop on Open Hypermedia Systems - Hypertext '96, pp. 81-86, 1996

[Wilde 2002] Wilde, E. : „Linkbase Access Protocol Design“, in „Proceedings of the 11th International World Wide Web Conference“, Honululu, 2002

[Wilde 2002b] Wilde, E.: „Protocol Considerations for Web Linkbase Access“ TIK Report 143, 2002

[Wilkens 2002] Wilkens, A: „44 Prozent der Deutschen gehen ins Netz“, 05.09.2002, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/anw-05.09.02-005

[Wilkinson und Smeaton 1999] Wilkinson, R. und Smeaton, A. F.: „Automatic Link Generation“, ACM Computing Surveys 31(4), December 1999

[Wollenweber 2002] Wollenweber, F.: „Entwicklung eines generischen Robots für das Scone-Framework“, Studienarbeit am Fachbereich Informatik der Universität Hamburg http://vsis-www.informatik.uni-hamburg.de/publications/readstu.phtml/SA/173/studienarbeit.pdf

8 Anhang

Eine DTD für multiple Links

```
<!ELEMENT multiple_link
   ((starting_resource|original|alternative|outbound|related)*)>
<!ATTLIST multiple_link
   xmlns:xlink     CDATA       #FIXED "http://www.w3.org/1999/xlink"
   xlink:type      (extended)  #FIXED "extended">

<!ELEMENT starting_resource ANY>
<!ATTLIST starting_resource
   xmlns:xlink     CDATA       #FIXED "http://www.w3.org/1999/xlink"
   xlink:type      (resource)  #FIXED "resource"
   xlink:label     (start)     #FIXED "start">

<!ELEMENT original (title)>
<!ATTLIST original
   xmlns:xlink     CDATA       #FIXED "http://www.w3.org/1999/xlink"
   xlink:type      (locator)   #FIXED "locator"
   xlink:label     (end)       #FIXED "end">

<!ELEMENT alternative (title)>
<!ATTLIST alternative
   xmlns:xlink     CDATA       #FIXED "http://www.w3.org/1999/xlink"
   xlink:type      (locator)   #FIXED "locator"
   xlink:label     (end)       #FIXED "end">

<!ELEMENT outbound EMPTY>
<!ATTLIST outbound
   xmlns:xlink     CDATA       #FIXED "http://www.w3.org/1999/xlink"
   xlink:type      (arc)       #FIXED "arc"
   xlink:from      (start)     #FIXED "start"
   xlink:to        (end)       #FIXED "end">

<!ELEMENT related EMPTY>
<!ATTLIST related
   xmlns:xlink     CDATA       #FIXED "http://www.w3.org/1999/xlink"
   xlink:type      (arc)       #FIXED "arc"
   xlink:from      (end)       #FIXED "end"
   xlink:to        (end)       #FIXED "end"
   xlink:arcrole   CDATA       #FIXED "http://www.scone.de/xlink/related">
```