Universität Regensburg
Philosophische Fakultät IV: Sprach- und Literaturwissenschaften
Lehrstuhl für Informationswissenschaft
Dozent: Prof. Dr. Rainer Hammwöhner

SS 1999

Arbeit zum Hauptseminar „Hypermedia“

Funktionen des KHS im WWW

Walter Piechulla*
Kramgasse 7
93047 Regensburg

Claus Atzenbeck†
Rohrer Straße 1
93359 Wildenberg

1. Dezember 1999

*Walter Piechulla (Dipl.-Psych.)
E-Mail: Walter.Piechulla@psychologie.uni-regensburg.de
WWW: http://pc1521.psychologie.uni-regensburg.de/lst/walter/
†Claus Atzenbeck (stud. phil.)
E-Mail: Claus.Atzenbeck@stud.uni-regensburg.de
WWW: http://homepages.uni-regensburg.de/~atc16247/

Diese Hauptseminararbeit beschreibt die Entwicklung, Implementierung und Evaluation eines Konzepts zur ergonomischen und benutzerorientierten Präsentation von Hypertexteinheiten des Konstanzer Hypertext Systems (KHS) im World Wide Web (WWW). Die vorgestellte Lösung benutzt eine leicht modifizierte, in VisualWorks (Smalltalk-Programmiersystem der Firma Cincom) realisierte KHS-Implementierung namens KHS/R zur dynamischen Generierung von HTML-Code. Ein integriertes Übersichts- und Navigationselement ist als Java-Applet implementiert. Eine lauffähige Demonstration des Konzepts steht im WWW zum Ausprobieren[1] bereit. Eine heuristische Kurzevaluation ergibt ermutigende Hinweise darauf, dass die angestrebten Design-Ziele Selbsterklärungsfähigkeit und Einfachheit durchaus erreicht wurden.

[1] http://pc5939.psychologie.uni-regensburg.de:8008/realizeKHSClient/default.htm

Inhaltsverzeichnis

Abbildungsverzeichnis

0. Bereichseingrenzung

Im Rahmen des Hauptseminars „Hypermedia“ arbeiten wir an der Realisierung der Schnittstelle von KHS[2] und dem WWW. Unser Bereich, abgegrenzt von der Inhaltsdarstellung, die sich ausschließlich mit der Darstellung der Knoteninhalte auf HTML-Browsern beschäftigt, umfasst die Funktionalität des KHS im WWW. Darunter verstehen wir in erster Linie die Navigations- und Suchkomponente[3], sowie die Möglichkeit, weitere Funktionen zu aktivieren. Den größten Teil unserer Arbeit wird jedoch die Navigation in Anspruch nehmen.

Unsere Arbeit richtet sich in erster Linie auf die von Campell und Goodman beschriebene Präsentationsebene[4], d. h. die Benutzerschnittstelle des Hypertextsystems.[5] Parallel dazu müssen wir uns jedoch auch um die Generierung der HTML-Dokumente beschäftigen. Die Schnittstelle stellt VisualWave zur Verfügung.

1. Funktionale Elemente

Als erstes stellt sich die Frage nach den nötigen funktionalen Elementen. Welche gibt es? Welche werden benötigt?[6]

1.1. Konventionelle Metainformationen

Inhaltsverzeichnis. Ein Inhaltsverzeichnis ist besonders bei hierarchisch geordneten Systemen empfehlenswert,[7] denn „[e]ine strikte Monohierarchie ist eindeutig linear abarbeitbar.“[8] Da die Hypertextmaschine des KHS Knoten grundsätzlich in einer Hierarchie zusammenfasst, bietet sich ein Inhaltsverzeichnis auch für die Schnittstelle zum WWW an.

Register. Ein Register wäre wünschenswert, ist jedoch nicht unbedingt notwendig, zumal diese Funktion auch mit einem guten Suchmechanismus ersetzt werden kann.[9]

[2] Konstanzer Hypertext-System. Ein eigenständiges, in Smalltalk (VisualWorks.) geschriebenes Hypertextsystem.
[3] Vgl. Kuhlen (1991), 12.
[4] Nielsen (1996), 134.
[5] Nielsen (1996), 131.
[6] Folgende Aufzählung nach Kuhlen (1991), 136–160.
[7] Kuhlen (1991), 137.
[8] Kuhlen (1991), 29.
[9] In diesem Seminar beschäftigt sich keine Gruppe mit der Implementierung eines Registers.

Glossare. Weiterführende Erklärungen zu Begriffen sind mit einem Glossar zu lösen. Im KHS ist dafür der Knotentyp „Glossar“ zu definieren, zu welchem dann von den jeweiligen Seiten gesprungen werden kann.

1.2. Hypertextspezifische Orientierungs- und Navigationsmittel

Grafische Übersichten, vernetzte Sichten (*web views*). Mit der Realisierung einer solchen Übersicht beschäftigt sich eine weitere Gruppe. Zur Übersicht gehört auch die Darstellung des aktuellen Kontextes.

Autorendefinierte Übersichtsmitte. Im KHS ist es möglich, einen Knotentyp „Übersichtsmitte“ zu definieren, der das Zentrum einer Übersicht bildet. Die Definition eines eigenen Knotens ist deswegen notwendig, weil Übersichten automatisch anhand der Informationen im KHS erstellt werden sollen. Die Übersichtsmitte stellt einen für den Benutzer wichtigen Orientierungspunkt dar. Der Startknoten bietet sich als Orientierungspunkt[10] und Übersichtsmitte an. Die Implementierung liegt im Bereich der Gruppe, die sich mit Übersichten beschäftigt.

Geführte Unterweisungen (*guided tours*). Eine Reihe von Problemen treten bei Guided Tours auf. Da wir uns jedoch nur mit der Funktionalität des Hypertextes beschäftigen, fallen diese nicht in unseren Bereich.[11] Das Einbinden einer fertigen Funktion „Guided Tour starten“ wäre kein großes Problem.

Pfade. Hier gilt das Gleiche wie bei den geführten Unterweisungen.

Backtrack-Funktionen, Dialoghistorien. Eine Backtrack-Funktion ermöglicht dem Benutzer den benutzten Weg zurückzugehen. Diese Funktion trägt wesentlich zur Navigation bei. Deswegen ist die Einbindung notwendig. Eine weitere Gruppe beschäftigt sich mit solchen Funktionen.

Leserdefinierte Fixpunkte (*book marks*). Sehr wünschenswert sind temporäre Bookmarks.[12] In den meisten WWW-Browsern existieren Bookmarks; diese sind jedoch nicht

[10] Vgl. Nielsen (1996), 259.
[11] Auch hier gibt es keine Gruppe, die sich damit beschäftigt.
[12] Es fand sich in diesem Seminar keine Gruppe, die sich mit dieser Thematik beschäftigt.

nur für ein geschlossenes Hypertextsystem, wie es das KHS darstellt, beschränkt. Wünschenswert wären Bookmarks, die nur für das KHS-System gelten, parallel zu den anderen WWW-Bookmarks. So könnte ein speziall an die Bedürfnisse des KHS gerichtetes Bookmarksystem erstellt werden, das benutzerfreundlicher wäre als das der üblichen Browser.

Autorendefinierte Wegweiser (*thumb tabs*). Wegweiser leiten den Benutzer zu wichtigen Punkten und ermöglichen so eine schnelle Orientierung. Diese Funktionalität kann mit Hilfe eines besonderen Verknüfungstyps erreicht werden, der eben diese Verknüpfung im WWW als Wegweiser deklariert ausgibt. Wegen der eingeschränkten Zeit des Seminars werden wir diese Funktion nicht implementieren.

Markierung gelesener Bereiche (*bread crumbs*). Die meisten WWW-Browser erkennen bereits früher aktivierte Links. Diese Funktionalität kann dazu genutzt werden, um dem Benutzer Orientierung darüber zu geben, welche Wege er bereits begangen hat. Weiter kann man einen angezeigten Knoten mit dem Hinweis darstellen, dass dieser bereits besucht worden ist; eine Möglichkeit ist das farbliche Kennzeichnen eines solchen Links. Ein Hinweis auf das Datum und die Uhrzeit des letzten Besuchs sind denkbar.

2. Entwurf der Navigation

Die wichtigste Komponente bei Hypertexten auf Rezipientenseite ist die Navigation. Sie soll möglichst benutzerfreundlich gestaltet sein. Ein System gilt nach Nielsen (1996) als benutzerfreundlich, wenn es folgende Forderungen erfüllt:[13]

1. Einfach zu erlernen
2. Effizient in der Benutzung
3. Leicht zu behalten
4. Niedrige Fehlerrate
5. Gefällig in der Benutzung

Wir versuchen, ihnen mit unserem Entwurf gerecht zu werden.

[13] Vgl. Nielsen (1996), 275. Vgl. DIN 66234 Teil 8, bei denen folgende Gestaltungsgrundsätze beschrieben werden: Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Fehlerrobustheit (zitiert nach Herczeg (1994), 105).

2.1. Direkte Manipulation

Direkte Manipulation erlaubt dem Benutzer „schnelle, reversible, inkrementelle Aktionen, deren Auswirkungen auf die Objekte sichtbar sind“[14] und trägt damit zur einfacheren Bedienung bei. Jedoch dürfen die Möglichkeiten nicht zu komplex werden, so dass der Benutzer die Kontrolle verliert[15]. Die wichtigsten Vorteile der direkten Manipulation sind:[16]

1. Funktionalität wird über die Semantik der Objekte direkt erreicht ⇒ nur wenig Grundwissen nötig
2. Benutzer fühlt sich als Handelnder ⇒ subjektive Zufriedenheit
3. Leicht zu erlernende Operationen ⇒ für alle Benutzertypen geeignet

Allgemein ist festzustellen, dass die „Designer und die Anwender interaktiver Computersysteme … durch die *Attraktivität* direkt manipulativer Systeme immer mehr auf diese Technik fixiert“[17] sind.

Jedoch ist darauf zu achten, dass die Eingaben des Benutzers „so einfach und so konsistent wie möglich“[18] sind. Das bezieht sich speziell auf das Eingabealphabet,[19] in unserem Fall die Interaktion mit der Maus.

Ein weiteres Ziel bei der Darstellung ist, es dem Benutzer zu ermöglichen, „Systemausgaben zu selektieren und als Eingaben zu verwenden (*interreferentielle Ein-/Ausgabe*)“[20]. Dies erreichen wir dadurch, dass der Navigationsbereich sowohl zur Interaktion, d. h. zur Eingabe, als auch als Ausgabe der aktuellen Position genutzt wird.

2.2. Darstellungsart

2.2.1. Allgemeines

Bildet man die Knoten in grafischer Weise räumlich zueinander ab, so kann dadurch eine parallele Verarbeitung beim Benutzer erreicht werden, im Gegensatz zur rein sprachlichen Darstellung, die einen zusätzlichen Kodieraufwand bedeutet.[21]

[14] Kuhlen (1991), 15.
[15] Vgl. Kuhlen (1991), 16.
[16] Vgl. Kuhlen (1991), 15f und Herczeg (1994), 115.
[17] Herczeg (1994), 121.
[18] Herczeg (1994), 29.
[19] Herczeg (1994), 28.
[20] Herczeg (1994), 30.
[21] Vgl. Krause (1996), 5.

Der Navigationsbereich sollte unabhängig vom Kontext immer die gleichen Möglichkeiten bieten. Dadurch wird die Bedienung auch Anfängern erleichtert. Es gibt Studien, „die zeigen, daß es für Anfänger und Neulinge sehr schwer ist, ein eigenes mentales Modell des Informationsraumes aufzubauen, wenn mehrere Strukturierungen verwendet werden, da sie nicht ständig durch dieselbe Informationsdarstellung in ihrem Verständnis bestärkt werden.“[22]

Links sollen sowohl im Knoteninhalt als auch separat dargestellt werden („internes navigieren“[23]). Untersuchungen haben ergeben, dass „Benutzer um 26 % schneller arbeiten, wenn die Ankerpunkte im Text integriert sind“[24]. Links in einem separaten Navigationsteil („externes navigieren“[25]), haben sowohl Vor- als auch Nachteile. Ein Nachteil ist sicherlich die Gefahr der Desorientierung und des chaotischen Assoziationseffekts.[26] Die Vorteile überwiegen jedoch: Schelle Orientierung und Navigation sind möglich, weil eine parallele Verarbeitung stattfindet.[27] Außerdem werden kreative Assoziationen und Serendipity-Effekte unterstützt.[28] Als Serendipity-Effekt bezeichnet man den Vorgang, bei dem man „auf der Suche nach einer bestimmten Information von einer anderen Information so ‚beschlagnahmt‘ wird, daß über deren aktueller Dominanz das ursprüngliche Ziel irrelevant oder vergessen wird“. Man findet in der Literatur breite Übereinstimmung, dass „die Orientierungsteile von der Entwurfstechnik her von den informativen Teilen im engeren Sinne getrennt werden sollten.“[29]

Ein weiterer Vorteil der Trennung von Navigation und Knoteninhalt ist die daraus entstehende Möglichkeit, im WWW Mehrfachverbindungen (*fat links*)[30] darstellen zu können. Die verschiedenen Links werden im Navigationsbereich nebeneinander dargestellt. Auch Links, die nicht im Knoteninhalt vorkommen, können so dargestellt werden.

2.2.2. Metaphern

Einer unserer ersten Gedanken war es, eine Metapher bereitzustellen, die dem Benutzer eine intuitive Navigation ermöglichen sollte. Abb. 1 zeigt dazu ein Beispiel, das im WWW Anwendung findet.

[22] Nielsen (1996), 258.
[23] Kuhlen (1991), 136.
[24] Nielsen (1996), 138; nach einer Studie von Vora et al. (1994).
[25] Kuhlen (1991), 136.
[26] Vgl. Kuhlen (1991), 110.
[27] Vgl. Krause (1996), 5.
[28] Vgl. Kuhlen (1991), 110.
[29] Kuhlen (1991), 135. Zu den „Orientierungsteilen“ zählen wir auch unsere Navigationselemente, zumal sie neben den Navigationsmöglichkeiten auch den näheren aktuellen Kontext anzeigen.
[30] Vgl. Nielsen (1996), 107.

Abbildung 1: Matapher: Crispen Café (http://hiwaay.net/~crispen/worlds/cafe/2/cafe.wrl)

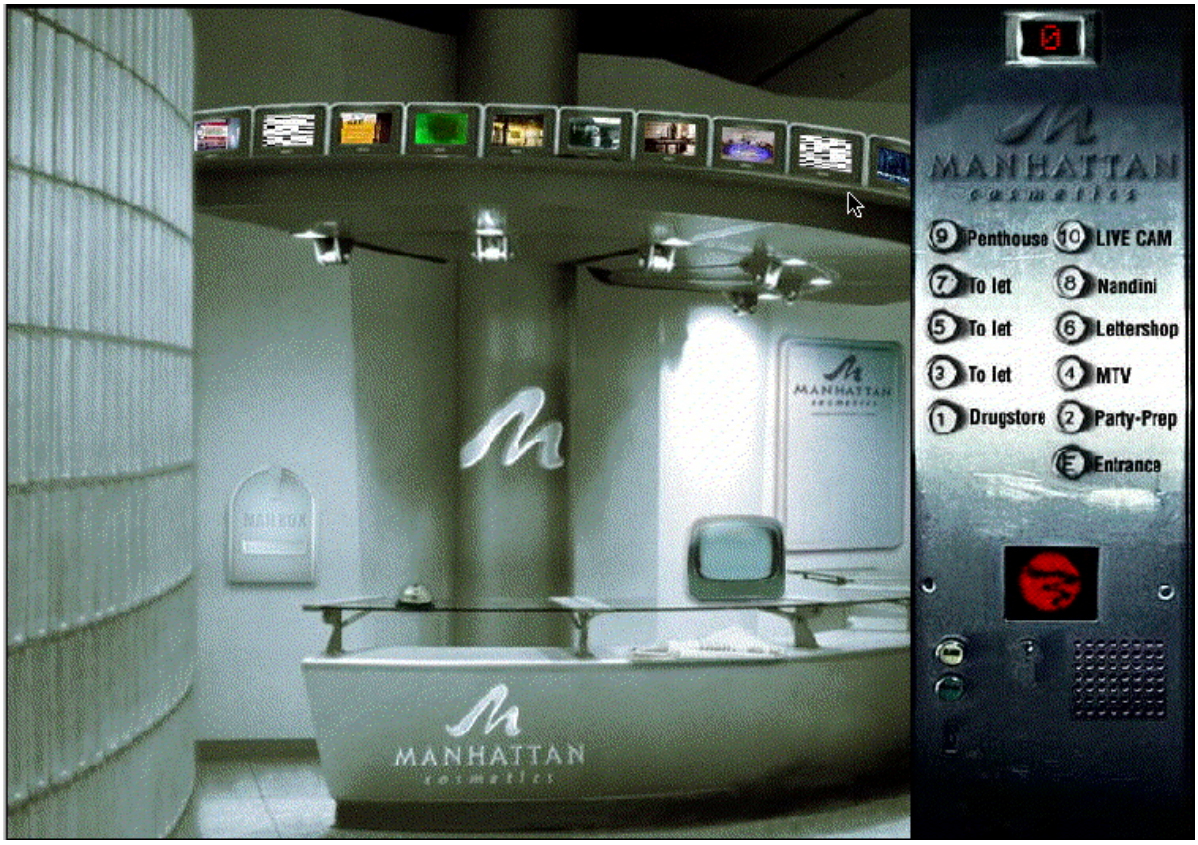

Abbildung 2: Matapher: Aufzug (http://www.manhattan.com)

Der Benutzer befindet sich in einem Café, in dem er sich frei bewegen kann. Selbst der Kellner-Roboter geht im Raum umher. Links zu anderen Seiten sind an den Wänden angebracht; die auf den Tischen befindlichen Symbole führen ebenfalls zu weiteren Seiten. Ein Klick auf einen Link öffnet ein neues Browser-Fenster. Technisch ist diese 3D-Welt mit VRML realisiert.

Unter http://www.intervista.com/vrml/gallery/site-nav.html finden sich weitere Beispiele zur Navigation mit Hilfe einer Metapher, die mit VRML realisiert sind.

Abb. 2 zeigt eine mit *Macromedia Flash!* animierte Aufzugmetapher der Firma Manhattan Cosmetics. Der Benutzer wählt via Mausklick zwischen den verschiedenen Stockwerken. Der Aufzug schließt sich, die Stockwerksanzeige rechts oben beginnt bis zum gewünschten Stockwerk zu zählen. Dort öffnet sich die Tür und der Benutzer scheint sich nach vorne zu bewegen. Außerhalb sind verschiedene Gegenstände klickbar, die die gewünschten Seiten in einem eigenen Fenster öffnen.

Man kann bei beiden Beispielen vermuten, dass Benutzer an der Metapher anfangs interessiert sind. Beide haben parallelen zu Adventure-Spielen, besonders die Aufzugmetapher. Interesse und Neugier werden geweckt, der Benutzer möchte entdecken, was in den verschiedenen Stockwerken zu sehen ist.

Diese Art der Navigation ist jedoch auf kleinere Anwendungen beschränkt. Für Sites mit vielen Knoten bietet sie nur ein mühsames Vorwärtskommen. Auch die automatische Anordnung von mehreren Knoten stellt sich als problematisch dar.

2.2.3. Visual formalisms

Obwohl eine Navigationsmetapher sehr reizvoll wäre, haben wir uns doch für die Darstellung als *Visual formalism* entschieden, weil diese ebenfalls leicht erlernbar ist, jedoch für unsere Zwecke eine klarere Darstellung und somit eine schnellere und einfachere Bedienung ermöglicht.

> „Visual formalisms sind … im Kern nichtbildhafte, nichtmetaphorische, visuelle Gestaltungsmittel, deren graphischer Charakter in Verbindung mit kognitiven Grundfähigkeiten eine effiziente, direktmanipulative Systembedienung ohne (bzw. mit nur geringem) Lernaufwand ermöglichen und die Problemlösung durch ‚external representation' unterstützen."[31]

Eine sehr gute Lösung bietet Inxight mit dem *Hyperbolic tree*, der in Abb. 3 zu sehen ist. Er ist als Java-Applet auf den meisten Browsern lauffähig und kann interaktiv bedient werden. So bewirkt ein einfacher Klick auf einen Knoten, dass sich dieser in die Mitte des Fensters bewegt; die restlichen Knoten werden während der Animation ständig automatisch angepasst. Eine andere Möglichkeit ist, mit gedrückter Maustaste einen beliebigen Punkt zu verschieben; auch hier werden alle Objekte in Laufzeit angepasst. Ein Doppelklick öffnet den Knoten in einem eigenen Fenster.

Der *Hyperbolic tree* ist als Fischauge realisiert, d. h. je näher ein Objekt in der Mitte liegt, desto genauer wird sein Umfeld angezeigt. Weit außen liegende Knoten, die im Fenster keinen Platz haben, werden nicht dargestellt. Angedeutet sind nur die dahin führenden Kanten.

Herczeg (1994) führt verschiedene Probleme bei Netzdarstellungen auf,[32] die jedoch beim *Hyperbolic tree* gelöst sind. So wird der Bildschirmplatz durch die Anordnung der Knoten gut genutzt und dadurch eine gute Übersichtlichkeit und Lesbarkeit erreicht.

[31] Krause (1996), 21; vgl. auch Nardi & Zarner (1993), 5.
[32] Vgl. Herczeg (1994), 99.

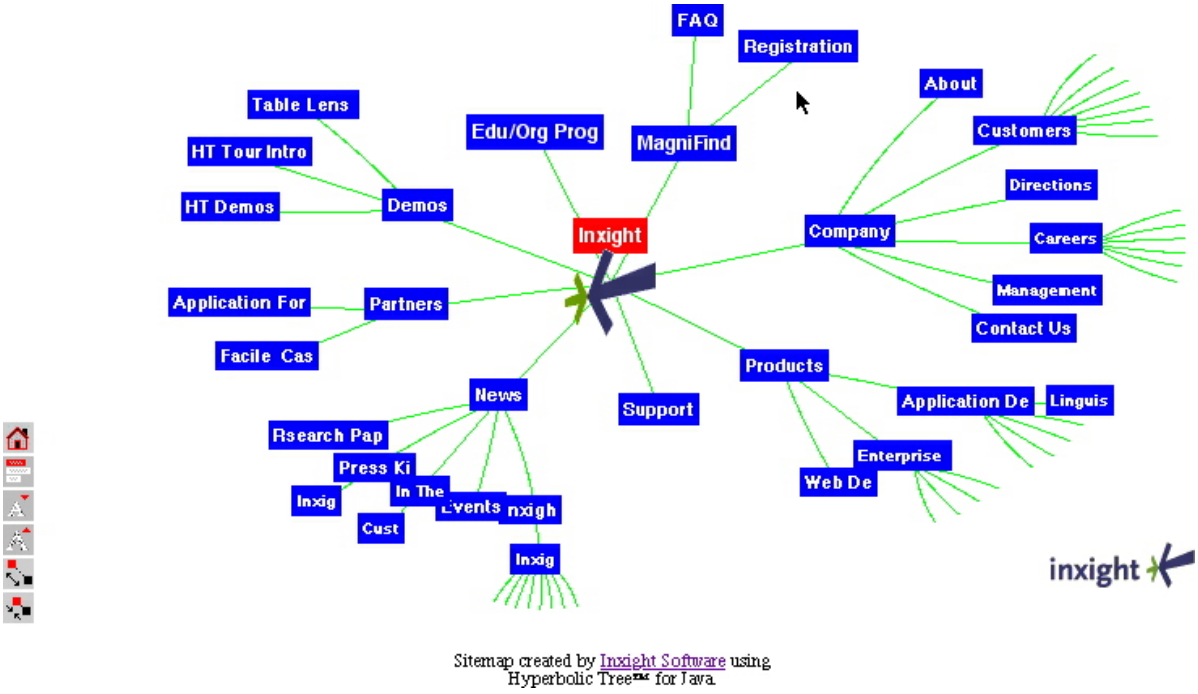

Abbildung 3: *Hyperbolic tree* (http://www.inxight.com/products/hw/IS2.html)

Der von Inxight angebotene *Hyperbolic tree* ist hauptsächlich zur Navigation in netzwerkartigen Hypertexten geschrieben. Obwohl wir grundlegend eine hierarchische Darstellung des KHS-Systems verfolgen, können wir die Funktionsweise für dieses System bei unseren Überlegungen nutzen.

2.3. Hierarchie und außerhalb liegende Knoten

Das KHS baut auf einer hierarchischen Struktur auf, d. h. jeder Knoten hat einen in der Hierarchie darüberliegenden Knoten. Wie bereits auf Seite 6 vermerkt, bietet sich bei so geordneten Hypertextbasen die Darstellung als Inhaltsverzeichnis an.[33]

[33] Vgl. Kuhlen (1991), 137. Andere Möglichkeiten, wie beispielsweise die Darstellung als alphabetisch geordnete Liste, wird in diesem Fall als weniger gut betrachtet.

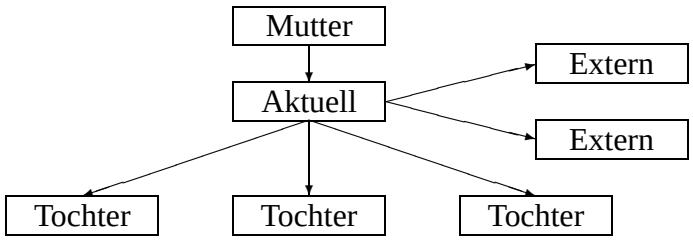

Abbildung 4: Navigationselemente unter Berücksichtigung der Hierarchiedomäne und der daraus herausführenden Links

2.3.1. Darstellung als visueller Formalismus

Die Problematik ergibt sich daraus, dass im KHS auch Links möglich sind, die außerhalb der aktuellen Hierarchiedomäne liegen; darunter fallen vor allem *assoziative Relationen*[34]. Diese gilt es, in die Navigation miteinzubinden. Ein reines Inhaltsverzeichnis scheidet damit aus. Abb. 4 zeigt den ersten Entwurf unserer Navigation, die sowohl die hierarchischen Gegebenheiten als auch die Links aus der aktuellen Hierarchiedomäne berücksichtigt. Er lehnt sich an den *Hyperbolic tree* an, weist jedoch mit der Anordnung der Objekte zusätzlich Metainformationen über die hierarchische Struktur in der jeweiligen Domäne aus.

Zu sehen ist die aktuelle Hierarchiedomäne. Der aktuelle Knoten befindet sich in der Mitte. Außerhalb liegen die aus der Hierarchie herausführenden Links. Jedes dieser Felder ist klickbar (außer der aktuelle Knoten); nach dem Klicken ändert sich entsprechend der Kontext: Der so ausgewählte Knoten wird zum aktuellen und rückt damit in die Mitte. Die Knoten in der Hierarchiedomäne und die Links nach außen passen sich dem aktuellen Knoten an.

Navigiert der Benutzer innerhalb der Hierarchie, ist folgendes zu bemerken:

1. Beim Klick auf einen Tochterknoten wird der aktuelle Knoten zum Mutterknoten, der neu gewählte Knoten wird als aktueller Knoten in die Mitte der Hierarchie gestellt.

2. Beim Klick auf den Mutterknoten rückt dieser an die Stelle des aktuellen Knotens. Der aktuelle Knoten wird dann als Tochter dargestellt.

Innerhalb der Hierarchie kann man nach „oben“ und „unten“ navigieren (*Natural mapping*[35]). Die nach außen führenden Links gestalten sich umso schwieriger: Mit dem Anwählen eines solchen Knotens ändert sich der komplette Kontext, zumal ein anderer Mutterknoten und

[34] Vgl. Kuhlen (1991), 104
[35] Vgl. Krause (1996), 22.

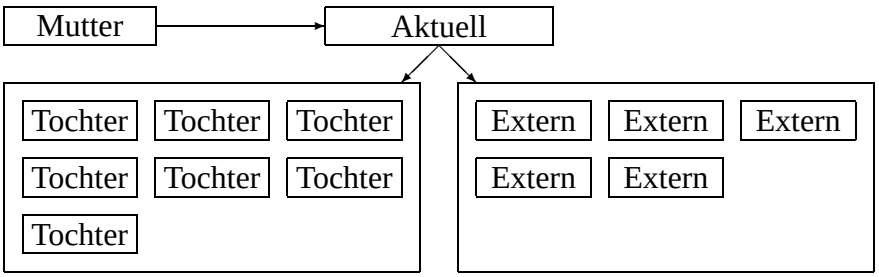

Abbildung 5: Clusterbildung der Navigationselemente

andere Tochterknoten existieren. Deswegen ist es wesentlich, dem Benutzer den Unterschied zwischen Töchter und herausführenden Links sichtbar zu machen.

Bei vielen Knoten kann die Darstellung, wie sie in Abb. 4 gezeigt wird, problematisch werden, zumal sie sehr viel Raum für die Platzierung der Knoten einnimmt. Abb. 5 zeigt eine abtrakte Darstellung des gleichen Prinzips: Der Mutterknoten befindet sich auf gleicher Ebene wie der aktuelle Knoten; die Hierarchie ist nur noch anhand der Pfeilrichtung erkennbar. Die Töchter und die außerhalb liegenden Knoten sind jeweils eingerahmt. Es führen nur noch jeweils ein Pfeil vom aktuellen Knoten zu den beiden Clustern.

Diese Darstellung wäre von der Platzausnutzung wesentlich günstiger, jedoch vermuten wir, daß dem Benutzer die Aufteilung in Hierarchie und außerhalb liegende Knoten nicht durchsichtig ist. Obwohl wir aus diesem Grund den Gedanken verworfen haben, sei er dennoch hier kurz erwähnt.

Abb. 6 zeigt einen der ersten Prototypen, der jedoch noch keine außerhalb der Hierarchie liegenden Links darstellen kann. Wir nennen diese Darstellung *Kartoffelstaude*, weil sie an eine Kartoffelpflanze erinnert, deren Kartoffelknollen ausgebreitet im Boden wachsen.

2.3.2. Listendarstellung

Die *Kartoffelstaude* verbraucht viel Platz, vor allem horizontal. Abb. 6 zeigt anschaulich, dass der darunterliegende Knoteninhalt nur unübersichtlich dargestellt werden kann. Werden die Zeilen über die gesamte Fensterbreite geschrieben, wäre nur noch ein sehr erschwertes Lesen möglich.

Um diese Problemen aus dem Weg zu räumen, haben wir alternativ eine Listendarstellung entworfen. Sie besitzt die gleichen Funktionselemente, lediglich die Anordnung ist anders. Abb. 7 zeigt einen Bildschirmauszug des Prototyps.

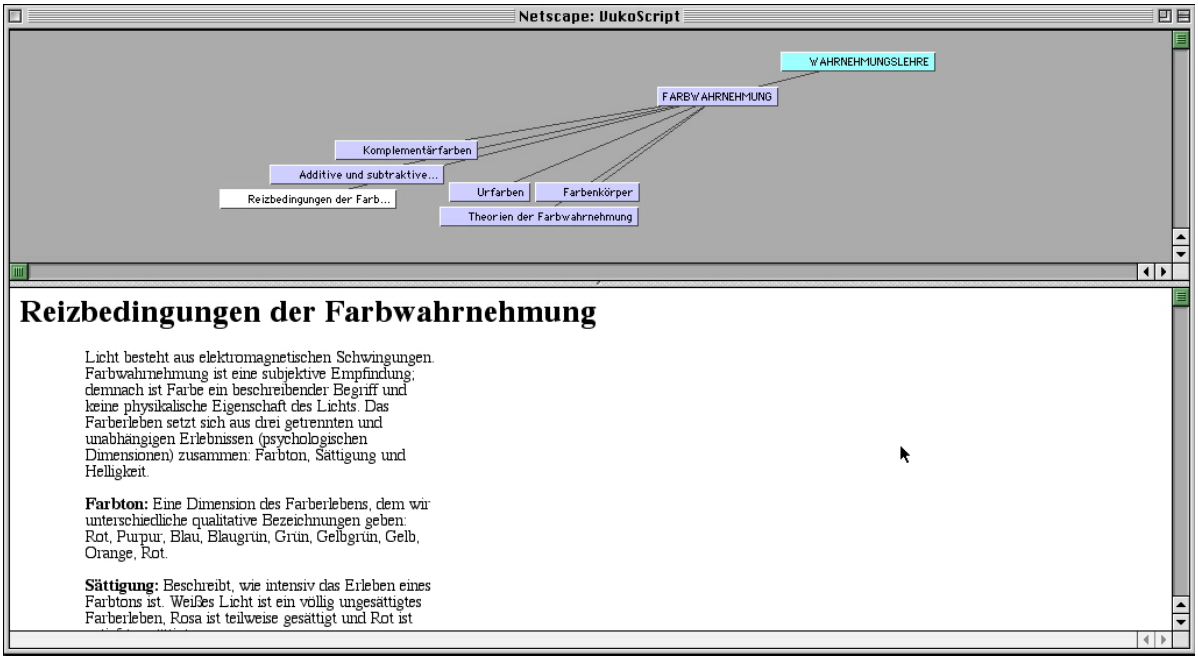

Abbildung 6: Navigationelemente als *Kartoffelstaude* (Prototyp)

Abbildung 7: Navigation als Listendarstellung (Prototyp)

Die Listendarstellung scheint ähnlich einem Inhaltsverzeichnis zu sein. Die Navigation ist am linken Rand, was dem Knoteninhalt etwas von der Breite nimmt, diesem jedoch die gesamte Fensterhöhe zugesteht. Die Aufteilung ist – wie bei der der *Kartoffelstaudendarstellung* – in verschiedene Bereiche eingeteilt:

1. Mutterknoten
2. Aktueller Knoten
3. Tochterknoten
4. Nach außen führende Links

2.4. Darstellung der Knotensymbole

2.4.1. Icons

Bei beiden Navigationsdarstellungen sind Piktogramme eingefügt. Sie sind auf das wesentliche reduziert, um einer besseren Informationsverarbeitung beim Benutzer gerecht zu werden.[36]

Die Listendarstellung besitzt vier verschiedene Piktogramme:

1. Pfeil nach oben: Zeigt den in der Hierarchie darüberliegenden Knoten an.
2. Raute: Zeigt den aktuellen Knoten an.
3. Pfeil nach unten: Zeigt die Tochterknoten an.
4. Pfeil nach links: Zeigt auf Knoten außerhalb der aktuellen Hierarchiedomäne

Das Verständnis der Piktogramme wird vom *Natural mapping* unterstützt. So ist „die bekannte Übertragung (abstrakter) hierarchischer Beziehungen … auf das räumliche Gegensatzpaar ‚oben/unten' … als kultureller Standard reklamiert"[37] worden.

Die *Kartoffelstaude* besitzt nur ein *Dokument*-Icon, das auf einen Knoten mit Inhalt hinweist. Fehlt dieses Icon, so handelt es sich um einen reinen Strukturknoten.

[36] Vgl. Krause (1996), 13.
[37] Martin/McClure (1985), 28; zitiert nach Krause (1996), 22.

2.4.2. Knotennamen

Beide Darstellungsarten zeigen neben den Icons auch die Titel der jeweiligen Knoten. Das wirkt sich positiv „für das Verstehen und Behalten des Wissens in den den Titeln zugeordneten Texten“[38] aus.

Bei der *Kartoffelstaude* bietet die Platzierung der Knotennamen und deren grafische Manipulation dem Nutzer Metainformationen über die Art des Knotens im jeweiligen Kontext. So erscheinen Mutter-, Tochter- und aktueller Knoten in einer jeweils anderen Farbe. Im Gegensatz zur Listendarstellung kann sich der aktuelle Knoten auch in der dritten Ebene befinden (sh. Abb. 6). Das ist dann der Fall, wenn der aktuelle Knoten keine Töchter mehr besitzt. Das gewährleistet ein schnelleres wechseln zu den Knoten gleicher Ebene, jedoch leidet darunter die Durchsichtigkeit für den Benutzer, warum einmal der aktuelle Knoten in zweiter, ein andermal in dritter Ebene dargestellt wird.

2.5. Funktionsobjekte

Die in Abschnitt 1 beschriebenen Funktionen können als Objekte sowohl in die *Kartoffelstaude* als auch in die Listendarstellung eingebunden werden. Es bieten sich Icons an, die am Rand des Navigationselements platziert werden. Bei der Listendarstellung, die vertikal angeordnet ist, wäre der beste Platz oberhalb der Navigationsliste.

3. Technische Realisierung

Dieser Abschnitt soll die praktische Implementierung eines Teilkonzepts vertieft behandeln, sowie Designentscheidungen erklären und rechtfertigen. Das Design ist generisch und nicht von bestimmten Programmiersprachen oder -werkzeugen abhängig. Die tatsächlich ausprogrammierten Teile des Konzepts sind lauffähig und tatsächlich in Benutzung.[39] Die technische Realisierung ist nicht Selbstzweck; sie dient der detaillierten Erklärung und Erprobung des Konzepts.

[38] Kuhlen (1991), 90.
[39] Als Bestandteil der Homepage http://pc1521.psychologie.uni-regensburg.de/lst/walter/

3.1. Ausgangslage

Die konkrete Fragestellung ist ein Gestaltungsvorschlag für die Präsentation von Hypertext-Einheiten im World Wide Web. Die Aufgabe umfaßt die Analyse der Problemsituation, die Vorstellung eines Lösungsansatzes, sowie die Demonstration einer lauffähigen Implementierung.

3.1.1. Das Basissystem

Als Ausgangsbasis steht ein modifiziertes Smalltalk Raw Image zur Verfügung. Es handelt sich dabei um die *VisualWorks*-Technologie der Firma ObjectShare (ehemals Parc Place Systems, eine Abspaltung des legendären Xerox PARC), die am 06. 09. 1999 in den Besitz der Firma Cincom Systems übergegangen ist[40]. Eine Mailingliste mit sehr aktiven Teilnehmern informiert über mit VisualWorks durchgeführte Projekte, bekannte Fehler, Probleme und anderes [41].

Dieses Smalltalk-Programmiersystem wird von Experten als einzige ernst zu nehmende Smalltalk-Implementierung angesehen. Über das beliebte Konkurrenzprodukt Smalltalk/V, das Anfang der 90er Jahre Aufmerksamkeit erregte, weil es das erste Smalltalk war, das auf dem marktbeherrschenden Windows[42] lief, fällt z. B. Rosenbeck das vernichtende Urteil „für industrielle Systeme untauglich“[43], denn es läßt in Sachen Robustheit und Performanz zu wünschen übrig.

3.1.2. Nutzung der Serverfunktionalität

Den Smalltalk-üblichen Begriff Raw-Image für die zur Verfügung stehende Programmierumgebung zu verwenden, ist eine starke Vereinfachung: Das Image enthält bereits eine am Lehrstuhl für Informationswissenschaft der Universität Regensburg durchgeführte Implementierung des Konstanzer Hypertext Systems (KHS) und einen rudimentären Web-Server. Um diesen benutzen zu können, mußte lediglich die Zeile

```
hostFileOut := 'localhost:8008'
```

in der Methode forVisualWave: accessor im Protokoll initialize der Klasse ServerDescription (die zur Klassenkategorie KHS Document Export gehört) durch die Zeile

[40] Vgl. ObjectShare (1999)
[41] vwnc@cs.uiuc.edu
[42] Präzise ausgedrückt: Betriebssystem Microsoft DOS mit Betriebssystemaufsatz Microsoft Windows.
[43] Rosenbeck (1995), 327.

hostFileOut := 'pc5939.psychologie.uni-regensburg.de:8008'

ersetzt, also die IP-Host-Adresse des eigenen Rechners eingetragen werden. Das Einrichten eines entsprechenden Servers in der Server-Konsole ist im VisualWave Application Developer's Guide[44] ausreichend besprochen.

3.1.3. Grenzen der Serverfunktionalität

Die „non-commercial"-Version von VisualWorks – also das eigentliche Raw Image – ist identisch mit dem kommerziellen Produkt VisualWorks Developer[45]. Damit enthält es die Klasse TinyHttpServer. Der TinyHttpServer ist zwar in erster Linie für Testzwecke gedacht, aber durchaus als WWW-Server verwendbar, hat allerdings zwei Einschränkungen: Erstens liefert er nicht automatisch Dokumente vom Dateisystem des Server-Rechners – dazu muß ein sogenannter FileResponder eingerichtet werden. Zweitens sind keinerlei Sicherheitsmechanismen enthalten. Für professionelle Anwendungen ist das Produkt VisualWorksServer gedacht, das mit kommerziellen Webservern zusammenarbeitet.[46]

3.1.4. Einen FileResponder einrichten

Für viele Zwecke ist es sinnvoll, in der VisualWave Server-Konsole einen FileResponder einzurichten. Auch das in dieser Arbeit vorgestellte Konzept ist auf einen FileResponder angewiesen: Da der Großteil der grafischen Darstellung auf dem Client-Rechner von einem Java-Applet erledigt wird und das *Sandbox*-Sicherheitskonzept von Java einem Applet nur erlaubt, mit dem WWW-Server Kontakt aufzunehmen, von dem es geladen wurde[47], muss die HTML-Seite, die das Java-Applet enthält (siehe Anhang A.3 auf Seite 40), von einem FileResponder (er heißt *realizeKHSClient*) geliefert werden[48]. Der VisualWave Application Developer's Guide widmet dem Einrichten eines FileResponder Resolver ein eigenes Kapitel,[49] so dass hier weitere Ausführungen überflüssig sind.

[44] ObjectShare (1998), 33–38.
[45] Vgl. Nowak (1999).
[46] Vgl. ObjectShare (1998), 33.
[47] Vgl. Culverhouse (1996). Der Mechanismus des Code-Signing erlaubt es einem Applet zwar, aus diesem „Sandkasten" auszubrechen, bei Standard-Sicherheitseinstellung des Browsers wird dann aber i. d. R. eine Rückfrage an den Benutzer ausgeführt, ob er „diesen Inhalten vertraut". Code-Signing bezieht kommerzielle Zertifizierungsagenturen mit ein, deren Qualitätszertifikat – falls vorhanden – mit angezeigt wird, ein solches Zertifikat ist aber nicht erforderlich, um die Sandbox zu umgehen.
[48] Es ist nicht auszuschließen, dass auch andere Wege gangbar sind, aber dies ist wohl die mit Abstand einfachste Lösung.
[49] ObjectShare (1998), 167–172.

3.2. Von der Analyse zur Design-Entscheidung

Die praktische Erfahrung mit dem KHS stammte zum einen aus dem Herumspielen mit dem Hypertext Browser des KHS/R, zum anderen aus der Exploration einer statisch mit dem KHS erzeugten Lehrstuhlhomepage[50]. Beides stellt den Status Quo der Inhaltsdarstellung beim KHS dar, der Mechanismus zum dynamischen, sitzungsweisen Erzeugen von HTML-Dokumenten wird unseres Wissens derzeit noch nicht eingesetzt.

3.2.1. Die KHS/R-Benutzerschnittstelle ist für Anfänger ungeeignet

Der KHS Hypertext Browser ist für den Anfänger sehr gewöhnungsbedürftig, weil unkonventionell. Ein gutes Beispiel sind die Auswahllisten im sogenannten Knoteninhaltsfenster[51], mit denen einer der Autoren dieses Textes eine sehr interessante Erfahrung machte: Erst nachdem das Prinzip des „Anwählen per Mausklick, dann Aktion Auslösen durch weiteren Mausklick“ in der KHS-Literatur[52] gefunden war, gelang es ihm, zu einer Untereinheit zu gelangen. Gängige Benutzeroberflächen kennen nur den Einfachklick und den Doppelklick. In einem Standardwerk zur Windows-Programmierung ist zu lesen: „In der Tat ist das Modell Einfach-/Mehrfachklick sehr intuitiv und findet beispielsweise nicht nur bei Windows, sondern auch beim Apple Macintosh weite Verbreitung“[53]. Jedenfalls zeigt die erwähnte Einzelfallbeobachtung, dass man als langjähriger Windows-Nutzer derart an dieses Modell gewöhnt sein kann, dass man zunächst nicht mit dem vom KHS Hypertext Browser geforderten Modell Klick-Warten-Klick zurecht kommt und sich wundert, warum keine Reaktion erfolgt. Die von Buchheit geäußerte Meinung, der Doppelklick sei intuitiv, wird allerdings von Wessel[54] als unter Windows-Kennern „verbreitetes Mißverständnis“ abgetan: „Bei GUI-Ein- und DOS-Umsteigern kann man deutlich beobachten, dass ihre Feinmotorikfähigkeit gerade hier erst einmal überfordert ist“. Das Argument überzeugt, es handelt sich wohl tatsächlich um einen reinen Gewöhnungseffekt. Eine Windows-Applikation, die sich nicht an die Windows-typischen Konventionen hält und damit nicht mit den Erwartungen des Benutzers konform geht, ist allerdings problematisch[55]. Dies ist aber ein generelles Problem von mit VisualWorks

[50] http://rsls8.sprachlit.uni-regensburg.de/KHS-Docs/IW/index.html
[51] Terminologie nach Hammwöhner (1997), 220.
[52] Hammwöhner (1997), 221–223.
[53] Buchheit (1992), 151.
[54] Wessel (1998), 93.
[55] Hier wird nur *ein* illustratives Beispiel stellvertretend für viele unkonventionelle Details der KHS-Werkzeuge besprochen, man könnte die Liste aber fast beliebig lang (und langweilig) machen.

erstellten Anwendungen. Auch ein Praktiker wie Rosenbeck[56] bemängelt, „wie weit eine ‚native VisualWorks-Anwendung' von der Windows-Konformität entfernt ist". Diese Bemerkung überrascht, weil Rosenbeck hauptsächlich anspruchsvolle Spezialanwendungen entwickelt, die nicht auf den Massenmarkt der Heimanwender abzielen (persönliche Kommunikation[57]). Trotzdem sieht auch er die Notwendigkeit zur Standardisierung der Benutzerschnittstelle.

3.2.2. Der Accidental User und sein Konservatismus

Da das KHS und seine Werkzeuge aber eindeutig als Hochleistungssystem für Spezialisten (Informationswissenschaftler) entwickelt wurden, scheint es müßig, seine Eignung für größere Benutzerkreise zu hinterfragen, das KHS *will* sich gar nicht an die üblichen Konventionen halten. Beim WWW-Nutzer dagegen hat man es mit dem Prototyp des Accidental User sensu Hollnagel[58] zu tun und muss daher großen Wert auf intuitive Bedienbarkeit und Selbsterklärungsfähigkeit legen. Außerdem ist bekannt, dass unter den WWW-Nutzern ein gewisser „Konservatismus" zu finden ist, neuartige Konzepte werden nicht geschätzt, die Erwartungen gehen eher in die Richtung: „Just give it to us plain and simple, using interaction techniques we already know from other sites"[59]. Spürbare Lern- und Eingewöhnungsphasen werden also nicht toleriert.

3.2.3. KHS-konformes Web-Design ist antiintuitiv

Die oben schon erwähnte Homepage des Lehrstuhls für Informationswissenschaft bietet zwar mit Index und Inhaltsverzeichnis leistungsfähige Suchmöglichkeiten, die dafür verwendeten Icons können aber nicht als selbsterklärend betrachtet werden. Die horizontale und vertikale Bewegungsmöglichkeit in der Hypertextstruktur ist zwar komfortabel, informelle Befragungen ergeben aber, dass so gut wie niemand, der nicht mit dem KHS vertraut ist, diese Art der Navigation versteht. Der Unterschied zwischen „zum übergeordneten Knoten" und „Zurückblättern" ist schlecht intuitiv erfassbar. Diese Aussagen beruhen allerdings nur auf eigenen Erfahrungen und informeller Befragung im Gespräch mit Kommilitonen, regelrechte Benutzertests wurden nicht durchgeführt. Trotzdem entstand schnell die Überzeugung, dass eine noch sehr viel einfachere Lösung anzustreben sei.

[56] Rosenbeck (1995), 327.
[57] Peter Rosenbeck hat fast zehn Jahre lang erst Lisp, später Smalltalk am Institut für Psychologie der Universität Regensburg unterrichtet und war der Lisp-Lehrer eines der Autoren dieser Arbeit.
[58] Hollnagel & Marsden (1996).
[59] Nielsen (1998).

3.2.4. Die Suche nach der intuitiven Lösung

Eine Übersicht neuerer Navigationskonzepte im WWW (siehe Abschnitt 1 für eine theoretische Auseinandersetzung mit dem Thema) ließ die Idee eines integrierten Konzepts der Übersicht und Navigation aufkommen: Inhalte (z. B. Texte oder Bilder) repräsentierende Schaltflächen sollten durch ihre Positionierung die Beziehungen zwischen diesen Inhalten verdeutlichen, gleichzeitig sollte beim Anklicken einer Schaltfläche der entsprechende Inhalt in einer separaten Pane angezeigt werden (siehe Abschnitt 1.1 auf Seite 6 für bekannte Vor- und Nachteile bei Trennung von Navigation und Inhaltsdarstellung). Diese früh festgesetzten Eckpunkte des Designs haben schließlich auch überdauert.

Die erste probeweise implementierte Variante allerdings ging von einer Art Fenster auf eine lokale Teilmenge der Hypertextstruktur aus, die angewählte Einheit sollte stets ins Zentrum des Fensters rücken, also den Focus des Fensters verschieben, so dass immer die gerade selektierte Einheit und ihre unmittelbare Nachbarschaft dargestellt würde. Das fertige Demonstrationsprogramm hatte erstaunliche Ähnlichkeit mit den KHS-Graphen[60], das Konzept wurde aber wieder verworfen, nachdem es von einer Testperson[61] als sehr schlecht bewertet wurde. Das entscheidende Problem dieses Konzepts war, dass durch die ständige Neu-Focussierung des Fensters auf die angewählte Einheit der Überblick völlig verloren ging. Es wurde daher aufgegeben.

3.2.5. Feinkonzept nach einigen Verbesserungen

Nach den daraufhin vorgenommenen Änderungen ist die Navigationsfläche (oberer Frame in Abbildung 8) nun nach einem einfachen visuellen Formalismus (vgl. Abschnitt 2.2.3 auf Seite 13) gestaltet, der „tiefer eintauchen“ und „auftauchen“ als metaphorische Konnotate nahelegen soll. Somit scheint ein intuitives Erfassen der Funktionsweise durch den Benutzer wahrscheinlich, da – wir wagen es kaum auszusprechen, so selbstverständlich wie es ist – die Unterscheidung von oben und unten ein wesentliches Merkmal der menschlichen Orientierung im dreidimensionalen Raum darstellt. Donald A. Norman hat im Rahmen des Konzepts vom Nutzerzentrierten Design den Begriff des *Natural Mapping* für solche intuitiv erfassbaren Zusammenhänge eingeführt[62], ein schwer zu definierendes theoretisches Konstrukt, für das

[60] Vgl. Hammwöhner (1997), 202.

[61] Ein professioneller Programmierer mit langjähriger Erfahrung in der Applikationsprogrammierung mit grafischen Benutzerschnittstellen, man kann somit von einer Expertenbeurteilung sprechen.

[62] Norman (1986).

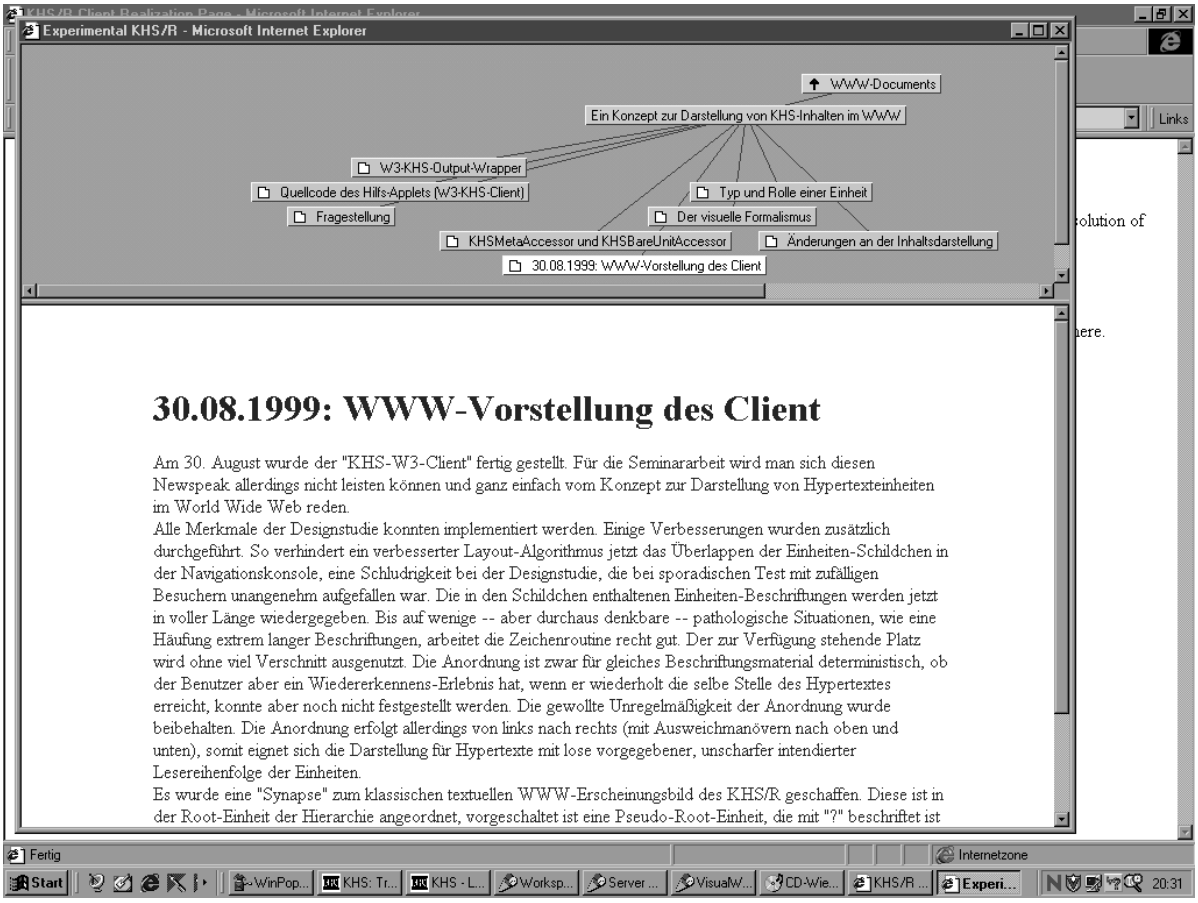

Abbildung 8: Bildschirmauszug des Konzepts auf Windows 95

man am ehesten durch die Lektüre seiner vergnüglich zu lesenden Polemik gegen schlechtes Design von Alltagsgegenständen[63] ein Gespür bekommt. Zum Verständnis der folgenden Ausführungen ist es sinnvoll, die Online-Demonstration[64] des Konzepts parallel zum Lesen dieses Textes auszuprobieren, Abbildung 8 zeigt einen beispielhaften Bildschirmauszug der hier vorzustellenden Lösung.

Der Formalismus unterscheidet drei übereinander liegende Schichten, die einen lokalen Ausschnitt eines hierarchisch gegliederten Hypertextes darstellen. Die oben liegende Schicht bewirkt beim Anklicken mit der „Maus“ ein Verschieben des dargestellten Übersichts-Ausschnitts einer Hypertext-Monohierarchie[65] um eins höher, die darunter liegende Schicht da-

[63] Norman (1988).
[64] http://pc5939.psychologie.uni-regensburg.de:8008/realizeKHSClient/default.htm
[65] Das hier entwickelte und ausprogrammierte Feinkonzept eignet sich nicht für Polyhierarchien.

gegen bewirkt ein entsprechendes Absteigen in der Hierarchie. Ausnahmen von dieser Regel bilden einerseits der Root-Knoten („?“) in der oberen Schicht, andererseits „Blätter“ (terminale Knoten) in der unteren Schicht. In diesen beiden Fällen wird beim Anklicken lediglich der Focus innerhalb der Navigationsfläche gewechselt. Dieser Focus wird durch das Zeichnen des betroffenen Schildchens mit weißer Hintergrundfarbe angedeutet. Nicht angewählte Einheiten werden in einer von der Distanz zur Hypertext-Wurzel abhängigen Farbe dargestellt, die nahe der Wurzel eher bläulich ist und mit zunehmender Distanz zur Wurzel in ein Violett übergeht; damit soll die aktuelle „Tauchtiefe“ intuitiv erfassbar gemacht werden. Die obere Schicht und die mittlere Schicht können jeweils nur von einer einzigen Einheit besetzt sein, Verzweigungen und damit Navigationsentscheidungen finden nur in der unteren Schicht statt.

3.2.6. Abschaltung von Menü- und Iconleiste des Browsers

Es ergibt sich ein integriertes Instrument zur lokalen Übersicht und Navigation, dessen wesentliches Merkmal seine extreme Einfachheit ist. Im Idealfall soll das Gefühl, mit einem technischen Artefakt zu interagieren ganz zurücktreten gegenüber der Auseinandersetzung mit der inhaltlichen Gliederung des Hypertextes. Deshalb wird auch ein „nacktes“ Fenster ohne irgendwelche Browser-spezifischen Navigationsmöglichkeiten eröffnet, die Navigation soll komplett mit der integrierten Übersichts- und Navigationsfäche erfolgen. Damit der Nutzer aber nicht erschrickt, wenn er sich der üblichen Browser-Funktionalität beraubt sieht, wird dieses Fenster nicht im Vollbildmodus eröffnet, sondern erst einmal relativ klein (900×500 Bildpunkte) dargestellt.

3.2.7. Typisierung und Rollenverteilung

Jede Einheit – in der Übersichts- und Navigationsfläche als beschriftetes Schildchen repräsentiert, das gleichzeitig als Schaltfläche arbeitet – besitzt eine dichotome Typisierung. Eine Schaltfläche *kann* außerdem (muss aber nicht) eine Rolle in der Navigation besitzen.

Die Typisierung besteht schlicht darin, dass eine Einheit entweder Inhalt enthält oder nicht. Enthält sie Inhalt, dann erscheint ein kleines Seitensymbol (Piktogramm) in der zugeordneten Schaltfläche. Wegen der Trivialität der dichotomen Typisierung, die dem Benutzer nur sagt, dass er Schaltflächen ohne das Inhalts-Piktogramm oft gar nicht anzuklicken braucht, weil es sich lediglich um ein Strukturelement handelt, treten bekannte Probleme der Eindeutigkeit und Verständlichkeit von Piktogrammen[66] erst gar nicht auf.

[66] Vgl. Hammwöhner (1997), 197.

Die Rolle ist nicht Eigenschaft einer Hypertext-Einheit, sondern Merkmal seines symbolischen Stellvertreters (Schaltfläche) in der Navigationsfläche: Alle nicht-terminalen Schildchen in der oberen, sowie der unteren Ebene bewirken beim Anklicken eine Ausschnittsänderung, sozusagen eine Änderung der „Eintauchtiefe“ in den Hypertext und zeigen dies mit entsprechenden Pfeilsymbolen an. Um dem Benutzer Rückmeldung über das Anwählen einer Einheit zu geben, sind die Schildchen in der üblichen Art von Schaltflächen („Buttons“) implementiert, d.h. sie werden beim Anklicken „gedrückt“: Verschiebung um ein Pixel nach unten und nach rechts und Vertauschen der dunklen und hellen Umrandungen in der nach der Schattenwurf-Metapher pseudo-dreidimensional gestalteten Umrandung. Um die Rückmeldung über die Funktionalität der Buttons noch zusätzlich anzureichern, werden beim Anklicken zusätzlich vergrößerte Varianten der Typ- und Rollensymbole gezeichnet. Im Falle der Pfeile (Rollensymbole) entsteht so ein minimaler Bewegungseindruck, der einen Hinweis auf die ausgelöste Ausschnittsverschiebung beim Blick auf die Hypertext-Struktur liefert. Beim Inhaltspiktogramm dagegen soll beim Benutzer die Assoziation „Einheit tritt in den Vordergrund“ ausgelöst werden.

3.2.8. Verbesserungen an der Inhaltsdarstellung

Die inhaltliche Darstellung der KHS-Units wurde weitestgehend beibehalten, da sie ohnehin sehr ausgereift erscheint und den Gepflogenheiten des WWW entspricht. Lediglich beim Dilemma Lesbarkeit am Bildschirm versus Ökonomie beim Ausdruck wurde der Bildschirmlesbarkeit etwas mehr Gewicht beigemessen, indem die Zeilen auf eine erträgliche Länge reduziert wurden. Großzügiger Rand oben, sowie links und rechts verhindert außerdem, dass die Seite zu sehr „zugestrickt“ und damit etwas billig wirkt. Hier kann auf Gepflogenheiten und ästhetische Prinzipien aus dem Buchdruck verwiesen werden[67]. Diese Änderungen entsprechen auch den Empfehlungen, die die zweite Arbeitsgruppe „Inhaltsdarstellung“ im Hypermedia-Seminar erarbeitet hat (Seminarvortrag im SS 1999) . Um den Kontrast für die Lesbarkeit zu optimieren wurde die Hintergrundfarbe außerdem auf Weiß eingestellt[68] und die Vordergrundfarbe auf ein dunkles Grau. Vorbild war hierbei das in PDF-Dokumenten[69] oft anzutreffende Grau der Schrift, das von einem der Autoren dieser Zeilen als besonders

[67] Loviscach (1996). Ein „Extrakt“ daraus ist online verfügbar: http://pc1521.psychologie.uni-regensburg.de/student2001/Documents/Textsatz.html

[68] Standardmässig übernimmt der Browser die Hintergrundfarbe vom Betriebssystem, der Hintergrund wird somit je nach den vom Benutzer vorgenommenen Einstellungen durchaus nicht immer weiß dargestellt, es sei denn, das HTML-Dokument macht explizite Vorgaben.

[69] Portable Document Format der Firma Adobe.

angenehm empfunden wird. Es kann sich hierbei allerdings um eine Idiosynkrasie handeln, physiologische oder wahrnehmungspsychologische Erkenntnisse lagen dieser Entscheidung nicht zugrunde. Ingenieurpsychologische Richtlinien verlangen einen Kontrast[70] von mindestens 3:1, je nach Anwendung kann ein Kontrast von bis zu 7:1 sinnvoll sein[71], die von uns gewählte Farbgebung bewegt sich auf jeden Fall im Rahmen dieser Richtlinien.

3.2.9. Grenzen des Konzepts

Das vorgestellte Detailkonzept benötigt einen monohierarchischen Hypertext, die im KHS enthalteten komplexen Mechanismen zum Navigieren in Polyhierarchien wurden bewußt ausgespart. Dies einerseits, um die Aufgabe überschaubar zu halten. Andererseits, weil fraglich, ist, ob dafür ein brauchbarer visueller Formalismus überhaupt gefunden werden könnte. Für einfache Spezialfälle wie „Hierarchiewälder“, die nur aus zwei oder drei Hierarchiebäumen bestehen, wäre dies sicherlich möglich, eine generische Lösung, die ohne begrenzende Zusatzannahmen auskommt, dürfte aber nur sehr schwierig zu visualisieren sein (siehe Abschnitt 2.3.1 auf Seite 15 für eine vertiefte Auseinandersetzung mit dem Problem).

3.3. Implementierung

Die Demonstration des Konzepts ist online einsehbar[72], die gesamte WWW-Demonstration befindet sich auf pc5939.psychologie.uni-regensburg.de. Beim Aufruf passiert folgendes:

1. Wird der FileResponder-Resolver *realizeKHSClient* mit dem Parameter *default.htm* aufgerufen, liefert er die entsprechende HTML-Seite an den Browser (Anhang A.1 auf Seite 38).

2. In dieser Seite ist eine Java Script-Funktion *openWindow* definiert, die beim Aufruf (im Code gleich nach der Definition) ein Fenster mit den gewünschten Eigenschaften eröff-

[70] Kontrast ist für visuelle Reize definiert als das Verhältnis der Leuchtdichte eines Symbols zur Leuchtdichte des Hintergrundes. Leuchtdichte wird in der Psychologie und Arbeitswissenschaft in Candela pro Quadratmeter angegeben. Die nötigen Präzisionsmessgeräte stehen zwar am Institut für Psychologie zur Verfügung, die Messung erfordert aber größere homogene Farbflächen, für die extra ein Wegwerfprogramm hätte geschrieben werden müssen. Ausserdem sind die Ergebnisse von selbstleuchtenden Medien wie Monitore ohnehin nicht auf hinterleuchtete Flüssigkristalldisplays übertragbar. Schließlich kommt noch hinzu, dass der Benutzer den Kontrast i. d. R. an seinem Monitor / Display nachregeln kann, was ohnehin jede Messung hinfällig macht. In der Praxis ist also die Beurteilung des Kontrastes per Augenschein oft die bessere, weil effizientere Lösung.

[71] Vgl. Zimmer (1999).

[72] http://pc5939.psychologie.uni-regensburg.de:8008/realizeKHSClient/default.htm

net, in das die HTML-Seite *FramesDefinition.htm* (Anhang A.2 auf Seite 39) geladen wird.

3. Diese definiert zwei Frames, einen schmalen oben liegenden und einen größeren unten liegenden. In den unteren Frame lädt sie durch Aufruf der *access*-Methode des RegisteredSessionLauncher *khs* die Einstiegsseite (*KHSServer* ist der Name des Hypertextes, *unit* ist die Signatur des Parameters der Methode *access* und *WWW* ist der tatsächliche Parameter). Als Einstieg dient somit eine „Synapse“ zum bei Aufgabenstellung erhaltenen KHS-Wave[73] mit reiner Textdarstellung. In den oberen Frame lädt *FramesDefinition.htm* die HTML-Seite upper.htm (Anhang A.3 auf Seite 40).

4. Die Seite *upper.htm* enthält nicht viel mehr als einen Aufruf des Applets KHSClient (Anhang A.4 auf Seite 41). Dieses Java-Applet implementiert das integrierte Übersichts- und Navigationsfeld. Es kommuniziert mit dem RegisteredSessionLauncher *walterskhs*. Dieser löst mit der Signatur *meta* den *KHSMetaAccessor* und mit der Signatur *access* den *KHSBareUnitAccessor* aus.

Die Klassen KHSBareUnitAccessor und KHSMetaAccessor wurden in der Kategorie KHS-Wave neu definiert und sind triviale Varianten der Klasse KHSUnitAccessor. Deren Methode setHTMLTexts ist folgendermaßen definiert:

```
1   | unit |
2   unit := context selectedUnit.
3   mapper printHierarchy.
4   hier value: mapper htmlText.
5   mapper printNavigationForm: unit.
6   links value: mapper htmlText.
7   mapper printUnitToHTML: unit.
8   text value: mapper htmlText
```

Beim KHSMetaAccessor, den das Applet aufruft, um Metainformationen über die Hypertextstruktur zu bekommen, sind nun im wesentlichen die Zeilen 5 und 6 weggelassen, weil sie für das Applet irrelevante Informationen in den erzeugten HTML-Code einfügen. Beim BareUnitAccessor dagegen sind die Zeilen 3, 4, 5 und 6 weggelassen, damit das Applet eine

[73] Einige triviale Änderungen, die auch dort schon vorgenommen wurden, sind hier nicht der Rede wert.

HTML-Seite anfordern kann, die nur den „nackten“ Inhalt für die Darstellung im unteren Frame enthält. Der HTML-Code, den der KHSMetaAccessor liefert, ist im Applet genauso leicht zu interpretieren wie jede andere denkbare Codierung der Metainformation. Daher wurden die Eingriffe in das KHS/R minimal gehalten. Dadurch ergibt sich die interessante Möglichkeit, das KHS/R-System ohne viel Arbeitsaufwand durch eine neuere, weiterentwickelte Version zu ersetzen, sobald diese existiert.

Das heutige KHS/R krankt aber unseres Erachtens vor allem daran, dass es keine lehrbuchartige Dokumentation der (hochkomplexen) Systemarchitektur gibt, die es einem als Student möglich machen sollte, sich im Selbststudium schnell soweit einzuarbeiten, dass man ohne monatelange Systemanalyse nichttriviale Änderungen vornehmen kann.

4. Evaluation

Um gut begründete Aussagen darüber machen zu können, ob das vorgestellte grafische Konzept tatsächlich leichter bedienbar oder auch nur für das subjektive Empfinden des Durchschnittsbenutzers angenehmer ist als der mit der Aufgabenstellung ausgelieferte textorientierte Ansatz, wäre eine empirische Studie erforderlich.

4.1. Ein Versuchsdesign

Es wäre ein leichtes, ein überzeugendes Leistungskriterium festzulegen, z. B. die kumulierte Zeit in Sekunden, die ein Benutzer braucht, um eine Liste von Fragen abzuarbeiten, die er nur bearbeiten kann, indem er sich Informationen aus einer geeigneten Dokumentenbasis, die ihm bereitgestellt wird, heraussucht. So könnte man das klassische textbasierte WWW-KHS gegen die hier vorgestellte grafische Lösung hinsichtlich ihrer *Usability* vergleichen.

Für die zufallskritische Absicherung der Ergebnisse würde sich ein zweifaktorielles varianzanalytisches Design mit den jeweils zweistufigen Faktoren (unabhängigen Variablen) Erfahrung (Anfänger versus Erfahrene) $\times$ KHS-Variante (klassisch versus grafisch) anbieten. Die KHS-Variante ist ein Within-Subjects-Faktor, also eine Wiederholungsmessung auf dem selben Individuum, zum Vermeiden von Reihenfolgeeffekten wird gegenbalanciert, die Hälfte der Versuchspersonen fängt also mit dem klassischen WWW-KHS an, die andere Hälfte mit der grafischen Variante. Der Faktor Erfahrung ist ein zweistufiger Between-Subjects-Faktor, hier werden zwei Extremgruppen verwendet, WWW-Unerfahrene und WWW-Erfahrene. Das varianzanalytische Design geht sehr schön auf z. B. mit 8 Versuchspersonen (zwei Erfahre-

ne/Unerfahrene fangen jeweils mit der einen oder anderen Variante an). Das größte Problem wäre die Konstruktion zweier gleich schwerer Parallelformen an Aufgaben – denn jede Vp wird ja *zwei mal* benutzt. Ein praktisch bedeutsamer Unterschied sollte sich schon bei wenigen Vpn zeigen (bei nur 8 Vpn muss der Effekt wirklich groß sein, um statistisch signifikant zu werden), man kann also mit der geringen Anzahl an Vpn auskommen.

So könnte man objektiv testen, ob es Unterschiede gibt und wenn ja, wie diese Unterschiede von der Erfahrung des Nutzers moderiert werden[74]. Sogar für die Aquisition der Versuchspersonen hätten wir schon ein Konzept: Wir würden Erstsemester als Versuchspersonen anwerben, in dieser Gruppe finden sich erfahrungsgemäß genug WWW-Desinteressierte[75], die hier als „Unerfahren“ eingestuft werden können. Außerdem hätten wir damit wohl kaum Probleme, andere potenzielle Einflussfaktoren wie Geschlecht und Alter in den beiden Erfahrungs-Extremgruppen zu parallelisieren.

Wir demonstrieren mit dem vorgestellten Versuchsdesign lediglich, dass wir die Methodik für derartige Evaluationsstudien beherrschen. Zwei wichtige Gründe sprechen nämlich dagegen, diesen Weg tatsächlich zu geben.

Der erste ist ein ganz pragmatischer, nämlich schlicht das Gefühl für die Verhältnismäßigkeit der Mittel: Die Benutzerstudie ist zu aufwendig, den Autoren dieser Arbeit entsteht beträchtlicher Zeitaufwand für die Versuchsdurchführung; außerdem sind Versuchspersonen rare Ressourcen und es ist somit äußerst unfair gegenüber Kommilitonen, die händeringend Vpn für ihre Diplomarbeitsexperimente suchen, Testpersonen für eine „nur“-Hauptseminararbeit zu aquirieren.

Interessanter ist der zweite Grund: Wie Eichinger (1999) ausführt, befindet man sich beim Usability-Testing generell sehr oft in der Situation, dass Benutzertests aus Kostengründen nicht in Frage kommen:

> „In den meisten Fällen wird nicht gefragt, welche Usability Engineering Maßnahmen korrekt sind sondern was mit gegebenen Ressourcen geleistet werden kann. Verglichen mit der am weitesten verbreiteten Usability Politik, nämlich diesen Bereich ganz zu ignorieren, sind Usability Inspection Methoden als Vertreter des sog. Discount-Usability Ansatzes mit Sicherheit vorzuziehen. Unter diesen Umständen müssen und können diese Methoden Ersatz für Usability Tests sein.“

[74] Wir würden vermuten, dass sowohl erfahrene, als auch Anfänger mit der grafischen Lösung besser zurecht kommen.

[75] Zur Zeit verfügen lediglich 10 % der Bevölkerung über einen Internet-Zugang (Focus Online, 1999), aber faktisch 100 % der Studierenden.

4.2. Beurteilung per Usability Inspection

Usability Inspection ist ein Sammelbegriff für eine Reihe von Methoden, bei denen Gutachter Usability-relevante Aspekte eines Produktes überprüfen. Die Gutachter können Endanwender, Domänenexperten, Softwareentwickler oder Usability Ingenieure sein. Der Ansatz baut auf den Glauben an die Fähigkeit der Gutachter, Probleme der Endanweder vorherzusagen.

Um das Verfahren zu strukturieren, wurde der an der Universität Osnabrück entwickelte standardisierte Usability-Fragebogen *Isometer*[76] verwendet. Es handelt sich dabei um ein Verfahren zur Evaluation von Software Nach ISO 9241/10 (Grundsätze der Dialoggestaltung). Zu den beispielsweise von Wessel[77] aufgelisteten Normen ISO 9241/11 bis ISO 9241/17, die stärker ins Datail gehen, wurde kein Material gefunden. Auch die Website der International Organization for Standardization ISO[78] hilft hier nicht weiter, unter der Rubrik FAQ findet sich lediglich der Hinweis: „For the moment, there is no electronic access to the content of ISO standards. However, we are working towards provision of that service“[79]. Daher wurde das recht allgemein gehaltene *Isometer*-Verfahren in der Kurzform IsoMetrics^{S}[80] verwendet. Sowohl die hier entwickelte grafische Schnittstelle, als auch die klassische Form des KHS-Wave wurden von einer einzigen Gutachterin, die Erfahrung im Umgang mit Büro-Anwendungen und Web-Browsern hatte, aber weder das KHS, noch das grafische Frontend vor der Beurteilung kannte, mit Isometrics^{S}-Fragebögen bewertet (Anhang B). Für eine wirklich aussagekräftige Bewertung verlangt Gediga (1999) mindestens 20 Bewerter, doch auch eine Einzelbegutachtung sollte schon Hinweise auf besondere Probleme geben.[81]

Weil von vorne herein klar war, dass viele der Fragen nicht anwendbar sind und deswegen unbeantwortet bleiben würden, wird per Antwortkategorie der Mittelwert aus den beantworteten Fragen verwendet. Man erhält so eine grobe Abschätzung folgender Eigenschaften:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität

[76] Download: http://www.psycho.uni-osnabrueck.de/isometer/
[77] Wessel (1998), 24.
[78] http://www.iso.ch
[79] http://www.iso.ch/infoe/faq.htm#Standards
[80] Vgl. Willumeit, Hamborg & Gediga (1996).
[81] Vgl. Eichinger (1999).

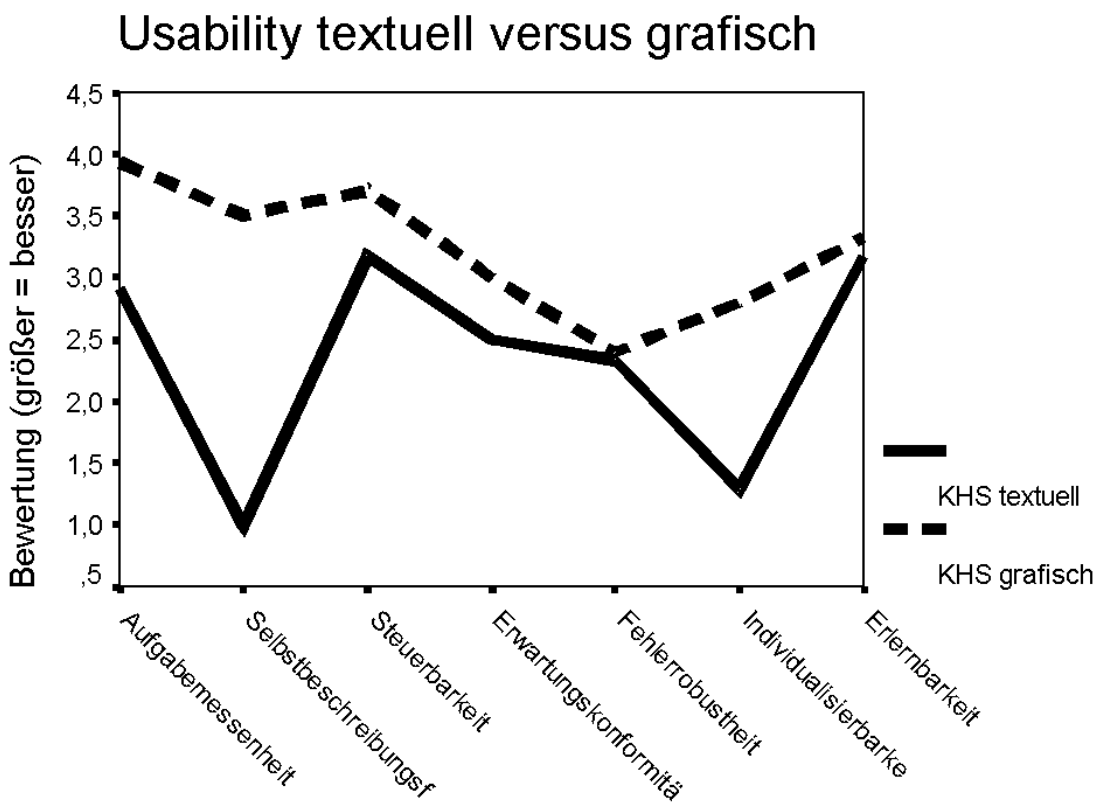

Abbildung 9: Usability Rating einer Beurteilerin

- Fehlerrobustheit
- Individualisierbarkeit
- Erlernbarkeit

Abbildung 9 zeigt die Ergebnisse. Bei der Interpretation ist Vorsicht angebracht, das Isometer-Verfahren wird hier nicht in seiner Eigenschaft als geeichtes Fragebogeninstrument verwendet, sondern lediglich als heuristisches Verfahren „missbraucht“.

Trotzdem zeichnet sich ab, dass zumindest diese Beurteilerin der grafischen Variante vor allem höhere Selbstbeschreibungsfähigkeit und Individualisierbarkeit (objektiv gesehen kann man allerdings nur die Größe der Frames anpassen) zuschreibt. Ansonsten erhält die grafische Variante generell eine etwas bessere Bewertung, nur bei Fehlerrobustheit und Erlernbarkeit

zeigen sich keine oder nur winzige Unterschiede. Dies ist plausibel, denn Fehler treten nicht auf und das Erlernen der Bedienung sollte angesichts der Primitivität der Gestaltung eigentlich in keiner der beiden Varianten besonders schwierig sein.

Insgesamt gesehen kann die Tatsache, dass bei umfassender, strukturierter Begutachtung im Einzelfall die grafische Lösung bevorzugt wird, als Hinweis darauf interpretiert werden, dass das Konzept durchaus gelungen ist. Der hier entwickelte visuelle Formalismus erweist sich somit als vielversprechender Ansatz zur lokalen Übersicht und Navigation in Hypertext-Einheiten.

Glossar

GUI Graphical User Interface.

ISO International Institute for Standardization. Einflussreiche europäische Normierungsinstitution. ISO kann nur Vorschläge machen, diese Richtlinien gehen aber oft in nationale Gesetzgebungen oder andere Regularien ein und werden damit verbindlich.

Pane Urprünglich engl. Wort für Fensterscheibe in einem Sprossenfenster, bei der GUI-Programmierung oft benutzte Bezeichnung für ein „Unterfenster“, in etwa synonym zu dem beim Web-Design bevorzugten Begriff *Frame*.

PARC Palo Alto Research Center der Firma Xerox, renommierte Forschungsstätte, auf die bahnbrechende Technologien wie Ethernet, Smalltalk und die grafische Benutzeroberfläche (GUI) zurückgehen.

Raw-Image Smalltalk-Programmierumgebung, wie vom Hersteller ausgeliefert, noch ohne eigene Änderungen an der Klassenhierarchie.

Usability „Usability is the measure of the quality of the user experience when interacting with something – whether a website, a traditional software application, or any other device the user can operate in some way or another.“[82]

Vp Versuchsperson.

Vpn Versuchspersonen.

[82] Nielsen (1998); zitiert nach Eichinger (1999).

Literatur

Buchheit, M. (1992). Windows Programmierbuch. Düsseldorf; San Francisco; Paris; Soest (NL): Synergy bei Sybex.

Culverhouse, M. (1996). Code Signing Java with Authenticode. In: M. Culverhouse, C. Walnut, N. Howell & G. Perry, Using Visual J++ (S. 519-564). O. O.: Que.

Eichinger. A. (1999). Usability. Skript auf dem WWW-Server des Lehrstuhls für Psychologie II der Universität Regensburg. Verfügbar: http://pc1521.psychologie.uni-regensburg.de/student2001/Skripten/Zimmer/usability.html

Focus Online (1999). WWW-Statistik. Das Internet in Zahlen. Online verfügbar: http://www.focus.de/D/DD/DD36/dd36.htm

Gediga, G. (1999). IsoMetrics. Development of a software usability instrument. Online verfügbar: http://www.psycho.uni-osnabrueck.de/isometer/

Hammwöhner, R. (1997). Offene Hypertextsysteme: das Konstanzer Hypertextsystem (KHS) im wissenschaftlichen und technischen Kontext. Konstanz: Univ.-Verl. Konstanz.

Herczeg, M. (1994). Software-Ergonomie. Grundlagen der Mensch-Computer-Kommunikation. Bonn et al.: Addison-Wesley.

Hollnagel, E.& Marsden, P. (1996) Human interaction with technology: The accidental user. Acta Psychologica 91, 345-358.

Krause, J. (1996). Visualisierung und graphische Benutzeroberflächen. Techn. Ber. 3, Informationszentrum Sozialwissenschaften.

Kuhlen, R. (1991). Hypertext. Ein nicht-lineares Medium zwischen Buch und Wissensbank. Berlin et al.: Springer.

Loviscach, J. (1996). Die Form zerbrechen. Layout zwischen Klassik und Moderne. c't 4/96, 242-245.

Nardi, B. A. & Zarmer, C. L. (1993). Beyond Models and Metaphors: Visual Formalisms in User Interface Design. Journal of Visual Languages and Computing, 4, 5-33.

Nielsen, J. (1996). Multimedia, Hypertext und Internet. Grundlagen und Praxis des elektronischen Publizierens (K. Lagrange, M. Linster übers.). Braunschweig; Wiesbaden: Vieweg.

Nielsen, J. (1998). The Increasing Conservatism of Web Users. Online verfügbar: http://www.useit.com/alertbox/980322.html

Nielsen, J. (1993). Usability Engineering. Boston: Academic Press.

Norman, D. A. (1986). User Centered System Design. New Perspectives On Human-Computer Interaction. Hillsdale, NJ: Erlbaum.

Norman, D. A. (1988). The Psychology of Everyday Things. New York: Basic Books.

Nowak, H. (1999). hnowak@objectshare.de "Re: ObjectShare infOO <20.05.1999> VisualWorks 3.1 für Linux now available". Online verfügbar: http://pc5939.psychologie.uni-regensburg.de:8008/realizeKHSClient/default.htm, Rubrik WWW Documents – VisualWorks news.

ObjectShare (1999). Pressemitteilung über den Verkauf von VisualWorks. München, 06. September 1999. Online verfügbar: http://pc5939.psychologie.uni-regensburg.de:8008/realizeKHSClient/default.htm, Rubrik WWW Documents – VisualWorks news.

ObjectShare (1998). VisualWave Application Developer's Guide. ObjectShare, Inc., 16811 Hale Avenue, California 92606-5089.

Rosenbeck, P. (1995). Es lohnt sich. Erfahrungen mit Smalltalk. c't 11/95, 324-330.

Wessel, I. (1998). GUI-Design. Richtlinien zur Gestaltung ergonomischer Windows-Applikationen. München; Wien: Hanser.

Willumeit, H., Hamborg, K. C. & Gediga, G. (1996). IsoMetrics^{S}. Fragebogen zur Evaluation von graphischen Benutzungsschnittstellen. Universität Osnabrück, Fachbereich Psychologie, Seminarstr. 20, D–49069 Osnabrück, Germany.

Zimmer, A. C. (1999). Kurzkompendium für kognitiv-ergonomische Gestaltungsrichtlinien für computergestützte Informationsgestaltung im Fahrzeug. Unveröffentlichtes internes SANTOS-Dokument (http://www.santosweb.de). Einsehbar nach Rücksprache mit dem Verfasser, Kontaktadressen online verfügbar: http://pc1521.psychologie.uni-regensburg.de/trafficresearch/AlfZimmer.htm

A. Quellcode

A.1. KHS-Client Aufrufseite (. . . /default.htm)

```
1  <html>
2  <head>
3     <META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <META NAME="Robots"  CONTENT="NOINDEX,NOFOLLOW">
5     <title>KHS/R Client Realization Page</title>
6  </head>
7
8  <body bgcolor="#FFFFFF">
9  <script language="javascript">
10
11 function openWindow() {
12    winStats='resizable=yes,toolbar=no,location=no,directories=no,menubar=no,'
13    winStats+='scrollbars=no,width=900,height=500'
14    if (navigator.appName.indexOf("Microsoft")>=0) {
15       winStats+=',left=10,top=10'
16     }else{
17       winStats+=',screenX=10,screenY=10'
18     }
19    floater=window.open("http://pc5939.psychologie.uni-regensburg.de:8008/
          realizeKHSClient/FramesDefinition.htm","ScriptView",winStats)
20 }
21
22 openWindow();
23
24 </script>
25
26 <h2>KHS/R Client Realization Page
27 </h2>
28
29 <p>If your browser did not open a new window, something went
30 wrong. You must have JavaScript and Java turned on to view these
31 pages. A minimum resolution of 1024 * 768 pixels is recommended.
32 </p>
33
34 <p>Use your browser's back button or
35 <a href="http://pc5939.psychologie.uni-regensburg.de:8008/realizeKHSClient/default.htm"
       >restart
36 KHS/R connection now
37 </a>
38 to go on.
39 </p>
40
41 <p>This is the practical part of a seminar paper. Those interested in the technical
       aspects can
42 <a href="http://pc1521.psychologie.uni-regensburg.de/lst/walter/Downloads/KHSClient.
       java">download
43 the source code file KHSClient.java (final version)
44 </a>
45 here.
46 </p>
47
48 </body>
49 </html>
```

A.2. KHS-Client-Fenster (FramesDefinition.htm)

```
1  <HTML>
2  <HEAD>
3  <META NAME="GENERATOR" Content="Microsoft␣Developer␣Studio">
4  <META HTTP-EQUIV="Content-Type" content="text/html;␣charset=iso-8859-1">
5  <META NAME="Robots" CONTENT="NOINDEX,NOFOLLOW">
6  <TITLE>Experimental KHS/R</TITLE>
7  </HEAD>
8
9  <frameset rows="220,*" marginheight=4 marginwidth=4 noresize>
10 <frame src="upper.htm" name="upperFrame" scrolling="yes">
11 <frame src="http://pc5939.psychologie.uni-regensburg.de:8008/khs/access/KHSServer/unit/
       WWW" name="vhlk" scrolling="yes">
12 </frameset>
13
14
15 <noframe>
16 <BODY>
17 Sorry, your browser does not support frames!
18 </BODY>
19 </noframe>
20 </HTML>
```

A.3. Übersichts-/Navigations-Frame (upper.htm)

```
1  <html>
2
3  <head>
4  <meta http-equiv="Content-Type"
5  content="text/html; charset=iso-8859-1">
6  <META name="GENERATOR" content="Microsoft FrontPage Express 2.0">
7  <META NAME="Robots" CONTENT="NOINDEX,NOFOLLOW">
8  <title>Applet Host Frame</title>
9  </head>
10
11 <body bgcolor="#ACA099" text="#FFFFFF">
12 <p align="center"><applet code=KHSClient codebase="JavaCode/KHSClient" align="baseline"
13 width="1200" height="200">
14 <param name="basiccolor" value="182,254,169">
15 <param name="serverandport" value="http://pc5939.psychologie.uni-regensburg.de:8008">
16 <param name="hypertext" value="KHSServer">
17 <param name="initialnode" value="WWW">
18 <param name="initiallabel" value="WWW">
19 <param name="structuremsg" value="KHSRLogo.htm">
20 </applet></p>
21 </body>
22 </html>
```

A.4. Quellcode des Client (KHSClient.java)

```
1  // KHSClient.java
2  // Written by Walter Piechulla in August 1999 for the seminar "Hypermedia"
3
4  // To read this code, set tab position of your editor to 2 (tab represented by 2 spaces
       ) !!!
5
6  // This applet was written with Microsoft Visual J++ 1.1 (oh, what a scandal!).
7  // Let me explain: I'm not a fan of Billyboy, I did it for practical reasons:
8  // The Developer Studio is a very convenient programming workplace, and the
9  // bytecode produced by J++ 1.1 runs on virtually EVERY browser's JVM. That's all.
10
11 import java.util.*;
12 import java.awt.*;
13 import java.net.*;
14 import java.io.*;
15 import java.applet.Applet;
16
17 class Node
18 {
19   String label;
20   String theURL;
21   Rectangle rc;
22   Point center;
23   boolean pushed;
24   boolean selected;
25   boolean hasContent;
26   boolean isTerminal;
27   boolean isUpper;
28   boolean isDowner;
29
30   public Node()
31   {
32     label = new String();
33     theURL = new String();
34     rc = new Rectangle();
35     center = new Point(0,0);
36     pushed = false;
37     hasContent = true;
38     isTerminal = false;
39     isUpper = false;
40     isDowner = false;
41     selected = false;
42   }
43 }
44
45 class GrandDaughter extends Node
46 {
47   public GrandDaughter()
48   {
49     // Nothing to do
50   }
51 }
52
53 class Daughter extends Node
54 {
55   Vector GrandDaughters;
56   int distance2root;
57
58   public Daughter()
59   {
60     GrandDaughters = new Vector(10,5);
```

```
61      }
62    }
63
64    class Mother extends Node
65    {
66      Vector Daughters;
67
68      public Mother()
69      {
70        Daughters = new Vector(10,5);
71      }
72    }
73
74    class LabelsManager
75    {
76      Vector deadRectangles;
77      boolean justPastDaughter;
78
79      public LabelsManager()
80      {
81        deadRectangles = new Vector(10,5);
82        justPastDaughter = false;
83      }
84
85      private Rectangle gloved(Rectangle naked,int gloveSize)
86      {
87        Rectangle withGlove = new Rectangle();
88        withGlove.reshape(naked.x,naked.y,naked.width,naked.height);
89        withGlove.grow(gloveSize,gloveSize);
90        return withGlove;
91      }
92
93      private boolean anyCollision(Rectangle rc,int gloveSize)
94      {
95        for (Enumeration enum = deadRectangles.elements();enum.hasMoreElements();)
96        {
97          Rectangle avoidMe = (Rectangle) enum.nextElement();
98
99          if (avoidMe.intersects(gloved(rc,gloveSize)))
100           return true;
101       }
102
103       return false;
104     }
105
106     // To understand this you must know that a Rectangle comes in with its true
          dimensions,
107     // but centered on its predecessor (to be improved)
108     public Rectangle getAnotherLabelRc(int direction,Rectangle preparedRc,Rectangle
          avoidThisRc,int gloveSize)
109     {
110       Rectangle resultRc = new Rectangle(preparedRc.x,preparedRc.y,preparedRc.width,
            preparedRc.height);
111
112       switch (direction)
113       {
114         case 3:  // == going left down == drawing daughter == refresh
115               justPastDaughter = true;
116               deadRectangles.removeAllElements();
117               deadRectangles.addElement(gloved(avoidThisRc,gloveSize));
118
119               while (anyCollision(resultRc,gloveSize))
120               {
```

```
121               resultRc.translate(-4,1);  // Go to the left, go downwards
122             }
123             break;
124
125         case 1: // Go to the right, go upwards
126             if (justPastDaughter)
127             {
128               deadRectangles.removeAllElements();
129               justPastDaughter = false;
130             }
131
132             deadRectangles.addElement(gloved(preparedRc,gloveSize));
133
134             while (anyCollision(resultRc,gloveSize))
135             {
136               resultRc.translate(2,-1);
137             }
138             break;
139
140         case 2: // Go to the right, go downwards
141             if (justPastDaughter)
142             {
143               deadRectangles.removeAllElements();
144               justPastDaughter = false;
145             }
146
147             deadRectangles.addElement(gloved(preparedRc,gloveSize));
148
149             while (anyCollision(resultRc,gloveSize))
150             {
151               resultRc.translate(2,1);
152             }
153             break;
154       }
155
156       return resultRc;
157     }
158
159   }
160
161   class GraphPanel extends Panel implements Runnable
162   {
163     public Mother theMother = new Mother();
164     StringBuffer emergencyStringBuf = new StringBuffer();
165     boolean fatalError = false;
166     int antiCollisionMemorizer = 0;
167     boolean newLayout = true;
168     Image contentSymbol;
169     Image upperSymbol;
170     Image downerSymbol;
171     Image offscreen;
172     Dimension offscreensize;
173     Graphics offgraphics;
174     final Color backgroundColor = new Color(172,160,153);
175     final Color standardAbgetoentesWeiss = new Color(255,251,240); // Marcellus Buchheit
          : Windows Programmierbuch p.83
176     Color leveledColors[] = new Color[9];
177     Frame BrowserFrame_;
178     LabelsManager labelsManager = new LabelsManager();
179
180     KHSClient graphObj;
181     Thread relaxer;
182
```

```
183     GraphPanel(KHSClient graphObj)
184     {
185       this.graphObj = graphObj;
186     }
187
188     public void run()
189     {
190       while (true)
191       {
192         relax();
193
194         try
195         {
196           Thread.sleep(100);
197         }
198         catch (InterruptedException e)
199         {
200           break;
201         }
202       }
203     }
204
205     synchronized void relax()
206     {
207       repaint();
208     }
209
210     public synchronized void update(Graphics g)
211     {
212       Dimension d = size();
213       antiCollisionMemorizer = 0; // Rewind avoid-predecessor-logic of
              getNiceNonIntersectingRc-method
214       int focusLevel;             // How far away ist the daughter
215
216       // Offscreen graphics context
217       if ((offscreen == null) || (d.width != offscreensize.width) || (d.height !=
              offscreensize.height))
218       {
219         offscreen = createImage(d.width,d.height);
220         offscreensize = d;
221         offgraphics = offscreen.getGraphics();
222         offgraphics.setFont(getFont());
223       }
224
225       offgraphics.setColor(getBackground());
226       offgraphics.fillRect(0,0,d.width,d.height);
227
228       FontMetrics fm = offgraphics.getFontMetrics();
229
230       if (!fatalError)
231       {
232         if (newLayout)
233         {
234           theMother.center.x = d.width/10*6;
235           theMother.center.y = d.height/11;
236           sizeAndPlaceTheLabel(theMother,fm,1);
237         }
238
239         // Knowing that there is actually only one daughter:
240         Daughter soleDaughter = (Daughter) theMother.Daughters.elementAt(0);
241         focusLevel = soleDaughter.distance2root;
242
243         if (newLayout)
```

```
244         {
245           passCenter(theMother,soleDaughter);
246           sizeAndPlaceTheLabel(soleDaughter,fm,2);
247           getNiceNonIntersectingRc(soleDaughter.rc,theMother.rc,5,true);
248           updateCenter(soleDaughter);
249         }
250
251         offgraphics.setColor(Color.darkGray);
252         offgraphics.drawLine(theMother.center.x,theMother.center.y,soleDaughter.center.x,
                soleDaughter.center.y);
253
254         drawNode(offgraphics,leveledColors[focusLevel],fm,theMother,theMother.pushed);
255         drawNode(offgraphics,leveledColors[focusLevel+1],fm,soleDaughter,soleDaughter.
                pushed);
256
257         boolean firstGrandDaughter = true;
258
259         Node lastNode = new Node();
260         Node lastLastNode = new Node();
261         lastLastNode = theMother;
262         lastNode = soleDaughter;
263
264         // Conection lines, some computations
265         for (Enumeration enum = soleDaughter.GrandDaughters.elements();enum.
                hasMoreElements();)
266         {
267           GrandDaughter grandDaughter = (GrandDaughter) enum.nextElement();
268
269           if (newLayout)
270           {
271             passCenter(lastNode,grandDaughter);
272             sizeAndPlaceTheLabel(grandDaughter,fm,3);
273
274             if (firstGrandDaughter)
275             {
276               getNiceNonIntersectingRc(grandDaughter.rc,soleDaughter.rc,80,true);
277               firstGrandDaughter = false;
278             }
279             else
280             {
281               getNiceNonIntersectingRc(grandDaughter.rc,soleDaughter.rc,2,false);
282             }
283
284             updateCenter(grandDaughter);
285           }
286
287           offgraphics.setColor(Color.darkGray);
288           offgraphics.drawLine(soleDaughter.center.x,soleDaughter.center.y,grandDaughter.
                center.x,grandDaughter.center.y);
289
290           lastLastNode = lastNode;
291           lastNode = grandDaughter;
292         }
293
294         // Draw labels
295         for (Enumeration enum = soleDaughter.GrandDaughters.elements();enum.
                hasMoreElements();)
296         {
297           GrandDaughter grandDaughter = (GrandDaughter) enum.nextElement();
298           drawNode(offgraphics,leveledColors[focusLevel+2],fm,grandDaughter,grandDaughter
                .pushed);
299         }
300
```

```
301         // Redraw daughter, may be crossed by lines drawn
302         drawNode(offgraphics,leveledColors[focusLevel+1],fm,soleDaughter,soleDaughter.
                pushed);
303       }
304       else
305       {
306         // Handling of fatal errors
307         offgraphics.setColor(Color.black);
308         offgraphics.drawString("Please␣notify␣walter.piechulla@psychologie.uni-regensburg
                .de␣of␣this␣problem:",2,10);
309         offgraphics.drawString(emergencyStringBuf.toString(),2,30);
310       }
311
312       g.drawImage(offscreen,0,0,null);
313       newLayout = false;
314     }
315
316     // Position: 1=up, 2=middle, 3=down
317     private void sizeAndPlaceTheLabel(Node node,FontMetrics fm,int position)
318     {
319       node.rc.width = fm.stringWidth(node.label) + 10;
320
321       if (node.hasContent)
322       {
323         node.rc.width += 20;
324       }
325
326       // Detect navigation role of label (if any): "upper" or "downer"
327       switch(position)
328       {
329         case 1: // Top: is an upper, if not root
330                 if (node.label.equals("?"))
331                 {
332                   node.isUpper = false;
333                   node.isDowner = false;
334                 }
335                 else
336                 {
337                   node.isUpper = true;
338                   node.isDowner = false;
339                 }
340
341                 break;
342
343         case 3: // Bottom: is a downer, if not a terminal unit
344                 if (node.isTerminal)
345                 {
346                   node.isDowner = false;
347                   node.isUpper = false;
348                 }
349                 else
350                 {
351                   node.isDowner = true;
352                   node.isUpper = false;
353                 }
354
355                 break;
356
357         default: break;
358       }
359
360       if (node.isUpper || node.isDowner)
361       {
```

```
362         node.rc.width += 20;
363       }
364
365       node.rc.height = fm.getHeight() + 4;
366       node.rc.x = node.center.x - node.rc.width/2;
367       node.rc.y = node.center.y - node.rc.height/2;
368     }
369
370     private void updateCenter(Node node)
371     {
372       node.center.x = node.rc.x + node.rc.width/2;
373       node.center.y = node.rc.y + node.rc.height/2;
374     }
375
376     private void passCenter(Node n1,Node n2)
377     {
378       n2.center.x = n1.center.x;
379       n2.center.y = n1.center.y;
380     }
381
382     private void drawNode(Graphics g,Color color,FontMetrics fm,Node node,boolean pushed)
383     {
384       if (node.selected)
385       {
386         g.setColor(Color.white);
387       }
388       else
389       {
390         g.setColor(color);
391       }
392
393       g.fillRect(node.rc.x,node.rc.y,node.rc.width,node.rc.height);
394       Point leftTop = new Point(node.rc.x,node.rc.y);
395       Point rightBottom = new Point(node.rc.x+(node.rc.width-1),node.rc.y+(node.rc.height
              -1));
396
397       if (pushed)
398       {
399         g.setColor(Color.darkGray);
400       }
401       else
402       {
403         g.setColor(standardAbgetoentesWeiss);
404       }
405
406       g.drawLine(leftTop.x,rightBottom.y,leftTop.x,leftTop.y);
407       g.drawLine(leftTop.x,leftTop.y,rightBottom.x,leftTop.y);
408
409       if (pushed)
410       {
411         g.setColor(standardAbgetoentesWeiss);
412       }
413       else
414       {
415         g.setColor(Color.darkGray);
416       }
417
418       g.drawLine(leftTop.x+1,rightBottom.y,rightBottom.x,rightBottom.y);
419       g.drawLine(rightBottom.x,rightBottom.y,rightBottom.x,leftTop.y+1);
420       g.setColor(Color.black);
421
422       if (pushed) // Draw shifted one pixel to the right and one pixel down
423       {
```

```
424         if (node.hasContent)
425         {
426           if (node.isUpper)
427           {
428             g.drawImage(upperSymbol,1 + node.center.x - (node.rc.width-10)/2,1 + (node.
                  center.y - (node.rc.height-4)/2 - 7),fm.getHeight()+4,fm.getHeight()+8,
                  this);
429             g.drawImage(contentSymbol,1 + 20 + node.center.x - (node.rc.width-10)/2,1 + (
                  node.center.y - (node.rc.height-4)/2),fm.getHeight()+4,fm.getHeight()+4,
                  this);
430             g.drawString(node.label,1 + 40 + node.center.x - (node.rc.width-10)/2,(node.
                  center.y - (node.rc.height-4)/2) + fm.getAscent());
431           }
432           else if (node.isDowner)
433           {
434             g.drawImage(downerSymbol,1 + node.center.x - (node.rc.width-10)/2,1 + (node.
                  center.y - (node.rc.height-4)/2),fm.getHeight()+4,fm.getHeight()+4,this
                  );
435             g.drawImage(contentSymbol,1 + 20 + node.center.x - (node.rc.width-10)/2,1 + (
                  node.center.y - (node.rc.height-4)/2),fm.getHeight()+4,fm.getHeight()+4,
                  this);
436             g.drawString(node.label,1 + 40 + node.center.x - (node.rc.width-10)/2,(node.
                  center.y - (node.rc.height-4)/2) + fm.getAscent());
437           }
438           else
439           {
440             g.drawImage(contentSymbol,1 + node.center.x - (node.rc.width-10)/2,1 + (node.
                  center.y - (node.rc.height-4)/2),fm.getHeight()+4,fm.getHeight()+4,this
                  );
441             g.drawString(node.label,20 + 1 + node.center.x - (node.rc.width-10)/2,1 + (
                  node.center.y - (node.rc.height-4)/2) + fm.getAscent());
442           }
443         }
444         else
445         {
446           if (node.isUpper)
447           {
448             g.drawImage(upperSymbol,1 + node.center.x - (node.rc.width-10)/2,1 + (node.
                  center.y - (node.rc.height-4)/2 - 7),fm.getHeight()+4,fm.getHeight()+8,
                  this);
449             g.drawString(node.label,1 + 20 + node.center.x - (node.rc.width-10)/2,(node.
                  center.y - (node.rc.height-4)/2) + fm.getAscent());
450           }
451           else if (node.isDowner)
452           {
453             g.drawImage(downerSymbol,1 + node.center.x - (node.rc.width-10)/2,1 + (node.
                  center.y - (node.rc.height-4)/2),fm.getHeight()+4,fm.getHeight()+4,this
                  );
454             g.drawString(node.label,1 + 20 + node.center.x - (node.rc.width-10)/2,(node.
                  center.y - (node.rc.height-4)/2) + fm.getAscent());
455           }
456           else
457           {
458             g.drawString(node.label,1 + node.center.x - (node.rc.width-10)/2,1 + (node.
                  center.y - (node.rc.height-4)/2) + fm.getAscent());
459           }
460         }
461       }
462       else
463       {
464         if (node.hasContent)
465         {
466           if (node.isUpper)
```

```
467           {
468             g.drawImage(upperSymbol,node.center.x - (node.rc.width-10)/2,1 + (node.center
                    .y - (node.rc.height-4)/2),fm.getHeight()-1,fm.getHeight()-1,this);
469             g.drawImage(contentSymbol,20 + node.center.x - (node.rc.width-10)/2,1 + (node
                    .center.y - (node.rc.height-4)/2),fm.getHeight()-1,fm.getHeight()-1,this
                    );
470             g.drawString(node.label,40 + node.center.x - (node.rc.width-10)/2,(node.
                    center.y - (node.rc.height-4)/2) + fm.getAscent());
471           }
472           else if (node.isDowner)
473           {
474             g.drawImage(downerSymbol,node.center.x - (node.rc.width-10)/2,1 + (node.
                    center.y - (node.rc.height-4)/2),fm.getHeight()-1,fm.getHeight()-1,this
                    );
475             g.drawImage(contentSymbol,20 + node.center.x - (node.rc.width-10)/2,1 + (node
                    .center.y - (node.rc.height-4)/2),fm.getHeight()-1,fm.getHeight()-1,this
                    );
476             g.drawString(node.label,40 + node.center.x - (node.rc.width-10)/2,(node.
                    center.y - (node.rc.height-4)/2) + fm.getAscent());
477           }
478           else
479           {
480             g.drawImage(contentSymbol,node.center.x - (node.rc.width-10)/2,1 + (node.
                    center.y - (node.rc.height-4)/2),fm.getHeight()-1,fm.getHeight()-1,this
                    );
481             g.drawString(node.label,20 + node.center.x - (node.rc.width-10)/2,(node.
                    center.y - (node.rc.height-4)/2) + fm.getAscent());
482           }
483         }
484         else
485         {
486           if (node.isUpper)
487           {
488             g.drawImage(upperSymbol,node.center.x - (node.rc.width-10)/2,1 + (node.center
                    .y - (node.rc.height-4)/2),fm.getHeight()-1,fm.getHeight()-1,this);
489             g.drawString(node.label,20 + node.center.x - (node.rc.width-10)/2,(node.
                    center.y - (node.rc.height-4)/2) + fm.getAscent());
490           }
491           else if (node.isDowner)
492           {
493             g.drawImage(downerSymbol,node.center.x - (node.rc.width-10)/2,1 + (node.
                    center.y - (node.rc.height-4)/2),fm.getHeight()-1,fm.getHeight()-1,this
                    );
494             g.drawString(node.label,20 + node.center.x - (node.rc.width-10)/2,(node.
                    center.y - (node.rc.height-4)/2) + fm.getAscent());
495           }
496           else
497           {
498             g.drawString(node.label,node.center.x - (node.rc.width-10)/2,(node.center.y
                     - (node.rc.height-4)/2) + fm.getAscent());
499           }
500         }
501       }
502     }
503
504     private void getNiceNonIntersectingRc(Rectangle dstRc,Rectangle avoidRc,int push,
          boolean drawDaughter)
505     {
506       int decision;
507
508       if (drawDaughter)
509       {
510         decision = 3;
```

```
511       }
512       else
513       {
514         if (antiCollisionMemorizer < 2)
515         {
516           decision = 1;
517           antiCollisionMemorizer++;
518         }
519         else
520         {
521           decision = 2;
522           antiCollisionMemorizer++;
523
524           if (antiCollisionMemorizer == 4)
525             antiCollisionMemorizer = 0;
526         }
527       }
528
529       Rectangle templateRc = labelsManager.getAnotherLabelRc(decision,dstRc,avoidRc,push
              );
530       dstRc.reshape(templateRc.x,templateRc.y,templateRc.width,templateRc.height);
531     }
532
533     public synchronized boolean mouseDown(Event evt,int x,int y)
534     {
535       for (Enumeration enum = theMother.Daughters.elements();enum.hasMoreElements();)
536       {
537         Daughter daughter = (Daughter) enum.nextElement();
538
539         for (Enumeration enum2 = daughter.GrandDaughters.elements();enum2.hasMoreElements
                ();)
540         {
541           GrandDaughter grandDaughter = (GrandDaughter) enum2.nextElement();
542
543           if (grandDaughter.rc.inside(x,y))
544           {
545             grandDaughter.pushed = true;
546             grandDaughter.selected = true;
547           }
548           else
549           {
550             grandDaughter.selected = false;
551           }
552         }
553       }
554
555       if (theMother.rc.inside(x,y))
556       {
557         theMother.pushed = true;
558         theMother.selected = true;
559       }
560       else
561       {
562         theMother.selected = false;
563       }
564
565       // One single daughter only
566       Daughter soleDaughter = (Daughter) theMother.Daughters.elementAt(0);
567
568       if (soleDaughter.rc.inside(x,y))
569       {
570         soleDaughter.pushed = true;
571         soleDaughter.selected = true;
```

```
572       }
573       else
574       {
575         soleDaughter.selected = false;
576       }
577
578       repaint(); // Immediately
579       return true;
580     }
581
582     // Shape cursor to hand when it is located on a label
583     public synchronized boolean mouseMove(Event evt,int x,int y)
584     {
585       for (Enumeration enum = theMother.Daughters.elements();enum.hasMoreElements();)
586       {
587         Daughter daughter = (Daughter) enum.nextElement();
588
589         for (Enumeration enum2 = daughter.GrandDaughters.elements();enum2.hasMoreElements
                ();)
590         {
591           GrandDaughter grandDaughter = (GrandDaughter) enum2.nextElement();
592
593           if (grandDaughter.rc.inside(x,y))
594           {
595             BrowserFrame_.setCursor(BrowserFrame_.HAND_CURSOR);
596             return true;
597           }
598         }
599       }
600
601       if (theMother.rc.inside(x,y))
602       {
603         BrowserFrame_.setCursor(BrowserFrame_.HAND_CURSOR);
604         return true;
605       }
606
607       // One single daughter only
608       Daughter soleDaughter = (Daughter) theMother.Daughters.elementAt(0);
609
610       if (soleDaughter.rc.inside(x,y))
611       {
612         BrowserFrame_.setCursor(BrowserFrame_.HAND_CURSOR);
613         return true;
614       }
615
616       BrowserFrame_.setCursor(BrowserFrame_.DEFAULT_CURSOR);
617       return true;
618     }
619
620     public synchronized boolean mouseUp(Event evt, int x, int y)
621     {
622       for (Enumeration enum = theMother.Daughters.elements();enum.hasMoreElements();)
623       {
624         Daughter daughter = (Daughter) enum.nextElement();
625
626         for (Enumeration enum2 = daughter.GrandDaughters.elements();enum2.hasMoreElements
                ();)
627         {
628           GrandDaughter grandDaughter = (GrandDaughter) enum2.nextElement();
629
630           if (grandDaughter.pushed)
631           {
632             grandDaughter.pushed = false;
```

```
633
634             try
635             {
636               URL metaURL = new URL(grandDaughter.theURL);
637               URL bareUnitURL = new URL(graphObj.meta2bare(grandDaughter.theURL));
638
639               if (grandDaughter.hasContent)
640               {
641                 graphObj.getAppletContext().showDocument(bareUnitURL,"vhlk");
642               }
643               else
644               {
645                 URL logoURL = new URL(graphObj.KHSRLogopage);
646                 graphObj.getAppletContext().showDocument(logoURL,"vhlk");
647               }
648
649               if (!grandDaughter.isTerminal)
650               {
651                 graphObj.CurrentDeepness++;
652                 graphObj.reInit(metaURL,grandDaughter.label,grandDaughter.theURL,
                        grandDaughter.hasContent);
653               }
654             }
655             catch (MalformedURLException e)
656             {
657               troubleShooting("MalformedURLException␣beim␣öffnen␣von␣" + grandDaughter.
                    theURL);
658             }
659           }
660         }
661       }
662
663       if (theMother.pushed)
664       {
665         theMother.pushed = false;
666
667         try
668         {
669           URL metaURL = new URL(theMother.theURL);
670           URL bareUnitURL = new URL(graphObj.meta2bare(theMother.theURL));
671
672           if (0 == theMother.label.compareTo("?")) // Special case
673           {
674             graphObj.getAppletContext().showDocument(metaURL,"vhlk");
675           }
676           else if (0 == theMother.label.compareTo(graphObj.getParameter("initiallabel")))
677           {
678             graphObj.CurrentDeepness--;
679             URL accessURL = new URL(graphObj.TraditionalAccessToEntryPage.toString());
680             graphObj.getAppletContext().showDocument(accessURL,"vhlk");
681             graphObj.reInit(metaURL,theMother.label,theMother.theURL,true);
682           }
683           else
684           {
685             graphObj.CurrentDeepness--;
686             URL logoURL = new URL(graphObj.KHSRLogopage);
687             graphObj.getAppletContext().showDocument(logoURL,"vhlk");
688             graphObj.reInit(metaURL,theMother.label,theMother.theURL,false);
689           }
690         }
691         catch (MalformedURLException e)
692         {
693           troubleShooting("MalformedURLException␣beim␣öffnen␣von␣" + theMother.theURL);
```

```
694         }
695       }
696
697       // Again using my knowledge, that there is only one daughter...
698       Daughter soleDaughter = (Daughter) theMother.Daughters.elementAt(0);
699
700       if (soleDaughter.pushed)
701       {
702         soleDaughter.pushed = false;
703
704         // Display alraedy ok, just load daughter
705         try
706         {
707           if (soleDaughter.hasContent)
708           {
709             if (0 == soleDaughter.label.compareTo(graphObj.getParameter("initiallabel")))
710             {
711               URL accessURL = new URL(graphObj.TraditionalAccessToEntryPage.toString());
712               graphObj.getAppletContext().showDocument(accessURL,"vhlk");
713             }
714             else
715             {
716               URL bareUnitURL = new URL(graphObj.meta2bare(soleDaughter.theURL));
717               graphObj.getAppletContext().showDocument(bareUnitURL,"vhlk");
718             }
719           }
720           else
721           {
722             URL logoURL = new URL(graphObj.KHSRLogopage);
723             graphObj.getAppletContext().showDocument(logoURL,"vhlk");
724           }
725         }
726         catch (MalformedURLException e)
727         {
728           troubleShooting("MalformedURLException␣beim␣öffnen␣von␣" + soleDaughter.theURL
                );
729         }
730       }
731
732       return true;
733     }
734
735     public void start()
736     {
737       relaxer = new Thread(this);
738       relaxer.start();
739
740       // Get handle to browser frame; this is a little tricky, I must admit; explanation
            at http://smapp.gis.de
741       Component ParentComponent;
742       ParentComponent = getParent();
743
744       while (ParentComponent != null && !(ParentComponent instanceof Frame))
745       {
746         ParentComponent = ParentComponent.getParent();
747       }
748
749       BrowserFrame_ = (Frame) ParentComponent;
750     }
751
752     public void stop()
753     {
754       relaxer.stop();
```

```
755     }
756
757     public void troubleShooting(String whatsWrong)
758     {
759       emergencyStringBuf.setLength(0);
760       emergencyStringBuf.append(whatsWrong);
761       fatalError = true;
762       repaint();
763     }
764   }
765
766   public class KHSClient extends Applet
767   {
768     GraphPanel panel;
769     StringBuffer TraditionalAccessToEntryPage = new StringBuffer();
770     StringBuffer StructureOnlyUnitsCallThisURL = new StringBuffer();
771     String KHSRLogopage = new String("http://pc5939.psychologie.uni-regensburg.de:8008/
          realizeKHSClient/KHSRLogo.htm");
772     int CurrentDeepness = 0;
773
774     public void init()
775     {
776       Vector daughtersNames = new Vector(10,5);
777       setLayout(new BorderLayout());
778
779       panel = new GraphPanel(this);
780       add("Center", panel);
781       //Panel p = new Panel(); This is how you would add panels
782       //add("South", p);
783
784       MediaTracker mediaTracker = new MediaTracker(this);
785       URL codeBase = getCodeBase();
786       panel.contentSymbol = getImage(codeBase,"ContentSymbol.gif");
787       mediaTracker.addImage(panel.contentSymbol,0);
788       panel.upperSymbol = getImage(codeBase,"UpSymbol.gif");
789       mediaTracker.addImage(panel.upperSymbol,0);
790       panel.downerSymbol = getImage(codeBase,"DownSymbol.gif");
791       mediaTracker.addImage(panel.downerSymbol,0);
792
793       try
794       {
795         mediaTracker.waitForAll();
796       }
797       catch (InterruptedException e)
798       {
799         panel.troubleShooting("Media␣tracker␣exception␣trying␣to␣load␣symbols");
800       }
801
802       panel.leveledColors[0] = new Color(135,255,255);
803       panel.leveledColors[1] = new Color(150,240,255);
804       panel.leveledColors[2] = new Color(165,225,255);
805       panel.leveledColors[3] = new Color(180,210,255);
806       panel.leveledColors[4] = new Color(195,195,255);
807       panel.leveledColors[5] = new Color(210,180,255);
808       panel.leveledColors[6] = new Color(225,165,255);
809       panel.leveledColors[7] = new Color(240,150,255);
810       panel.leveledColors[8] = new Color(255,135,255);
811
812       String serverandport = getParameter("serverandport");
813       String hypertext = getParameter("hypertext");
814       String initialnode = getParameter("initialnode");
815       String initiallabel = getParameter("initiallabel");
816       String structuremsg = getParameter("structuremsg");
```

```
817
818       try
819       {
820         TraditionalAccessToEntryPage.append(serverandport + "/khs/access/" + hypertext +
              "/unit/" + initialnode);
821         StructureOnlyUnitsCallThisURL.append(serverandport + "/realizeKHSClient/" +
              structuremsg);
822         URL iniURL = new URL(serverandport + "/walterskhs/meta/" + hypertext + "/unit/"
              + initialnode);
823         BufferedInputStream in = new BufferedInputStream(iniURL.openStream());
824
825         eatUninterestingStuff(in); // Consume stuff before hierarchy info
826         getMotherOfMine(in); // Consume hierarchy info
827
828         panel.theMother.label = "?";  // Because we are doing the initial request
829         panel.theMother.theURL = "http://pc5939.psychologie.uni-regensburg.de:8008/
              realizeKHSClient/KHSQuestionmark.htm"; // Via FileResponder
830         panel.theMother.hasContent = true; // The text of KHSQuestionmark.htm
831
832         // This would work for multiple daughters, too
833         if (daughtersNames.contains(initiallabel))
834         {
835           // Initially impossible
836         }
837         else // Make new daughter
838         {
839           Daughter daughter = new Daughter();
840           daughter.label = initiallabel;
841           daughter.theURL = TraditionalAccessToEntryPage.toString();
842           daughter.hasContent = true; // Open door to traditinal KHS
843           daughter.distance2root = CurrentDeepness;
844           daughter.selected = true;
845           panel.theMother.Daughters.addElement(daughter);
846           daughtersNames.addElement(daughter.label);
847         }
848
849         // This is outlined to work with multiple daugthers (not supported at the moment)
850         for (Enumeration enum = panel.theMother.Daughters.elements();enum.hasMoreElements
              ();)
851         {
852           Daughter daughterTmp = (Daughter) enum.nextElement();
853
854           if (0 == daughterTmp.label.compareTo(initiallabel)) // We got her
855           {
856             // Actually, there is only one daughter, so I simlpy append all
                  granddaughters here
857             while (true)
858             {
859               String aChildOfMine = new String(getNextChildOfMine(in));
860               int cmp = aChildOfMine.compareTo("UNDEFINED");
861
862               if (cmp == 0) // No more granddaughters
863               {
864                 break;
865               }
866               else
867               {
868                 GrandDaughter grandDaughter = new GrandDaughter();
869
870                 grandDaughter.label = labelSubString(aChildOfMine);
871                 grandDaughter.theURL = access2meta(urlSubString(aChildOfMine));
872                 grandDaughter.hasContent = infoSubStringIsTerminal(aChildOfMine);
873                 grandDaughter.isTerminal = infoSubStringIsTerminal(aChildOfMine);
```

```
874                 daughterTmp.GrandDaughters.addElement(grandDaughter);
875               }
876             }
877           }
878         }
879
880         in.close();
881       }
882       catch (MalformedURLException e)
883       {
884         panel.troubleShooting("MalformedURLException in method init()");
885       }
886       catch (IOException e)
887       {
888         panel.troubleShooting("IOException in method init()");
889       }
890       catch (SecurityException e)
891       {
892         panel.troubleShooting("SecurityException in method init()");
893       }
894
895       setBackground(panel.backgroundColor);
896       panel.repaint();
897     }
898
899     public void reInit(URL newDaughterURL, String dauLabel, String dauURL, boolean
          dauHasContent)
900     {
901       Vector daughtersNames = new Vector(10,5);
902
903       // Flush old context information
904       for (Enumeration enum = panel.theMother.Daughters.elements();enum.hasMoreElements
            ();)
905       {
906         Daughter daughter = (Daughter) enum.nextElement();
907         daughter.GrandDaughters.removeAllElements();
908       }
909
910       panel.theMother.Daughters.removeAllElements();
911
912       // Parse context information from URL (new daughter node)
913       try
914       {
915         BufferedInputStream in = new BufferedInputStream(newDaughterURL.openStream());
916         eatUninterestingStuff(in);
917
918         // Read mother
919         String motherOfMine = getMotherOfMine(in);;
920
921         panel.theMother.label = labelSubString(motherOfMine);
922         panel.theMother.theURL = access2meta(urlSubString(motherOfMine));
923         panel.theMother.hasContent = false;
924         panel.theMother.isTerminal = infoSubStringIsTerminal(motherOfMine);
925         panel.theMother.selected = false;
926
927         if (panel.theMother.label.compareTo(getParameter("initiallabel")) == 0) //
              Special case
928         {
929           panel.theMother.hasContent = true;
930         }
931
932         if (dauLabel.compareTo(getParameter("initiallabel")) == 0) // It's the root doc,
              so...
```

```
933         {
934           panel.theMother.label = "?";
935           panel.theMother.theURL = "http://pc5939.psychologie.uni-regensburg.de:8008/
                  realizeKHSClient/KHSQuestionmark.htm";
936           panel.theMother.hasContent = true;
937           panel.theMother.isTerminal = false;
938           panel.theMother.selected = false;
939         }
940
941         // Configure daughter; this is outlined to work with multiple daugthers (not
                supported at the moment)
942         if (daughtersNames.contains(dauLabel))
943         {
944           // Do nothing, daughters must not have identical names
945         }
946         else // Make new daughter
947         {
948           Daughter daughter = new Daughter();
949           daughter.label = new String(dauLabel);
950           daughter.theURL = new String(dauURL);
951           daughter.hasContent = dauHasContent;
952           daughter.isTerminal = false;
953           daughter.distance2root = CurrentDeepness;
954           daughter.selected = true;
955           panel.theMother.Daughters.addElement(daughter);
956           // Remember her name
957           daughtersNames.addElement(daughter.label);
958         }
959
960         // This is outlined to work with multiple daugthers (not supported at the moment)
961         for (Enumeration enum = panel.theMother.Daughters.elements();enum.hasMoreElements
                ();)
962         {
963           Daughter daughterTmp = (Daughter) enum.nextElement();
964
965           if (0 == daughterTmp.label.compareTo(dauLabel)) // We got her
966           {
967             // Actually, there is only one daughter, so I simlpy append all
                      granddaughters here
968             while (true)
969             {
970               String aChildOfMine = new String(getNextChildOfMine(in));
971               int cmp = aChildOfMine.compareTo("UNDEFINED");
972
973               if (cmp == 0) // No more granddaughters
974               {
975                 break;
976               }
977               else
978               {
979                 GrandDaughter grandDaughter = new GrandDaughter();
980
981                 grandDaughter.label = labelSubString(aChildOfMine);
982                 grandDaughter.theURL = access2meta(urlSubString(aChildOfMine));
983                 grandDaughter.hasContent = infoSubStringIsTerminal(aChildOfMine);
984                 grandDaughter.isTerminal = infoSubStringIsTerminal(aChildOfMine);
985                 daughterTmp.GrandDaughters.addElement(grandDaughter);
986               }
987             }
988           }
989         }
990
991         in.close();
```

```
992         panel.newLayout = true;
993         panel.repaint();
994       }
995       catch (MalformedURLException e)
996       {
997         panel.troubleShooting("MalformedURLException␣in␣method␣reInit()");
998       }
999       catch (IOException e)
1000      {
1001        panel.troubleShooting("IOException␣in␣method␣reInit()");
1002      }
1003      catch (SecurityException e)
1004      {
1005        panel.troubleShooting("SecurityException␣in␣method␣reInit()");
1006      }
1007    }
1008
1009
1010    private void eatUninterestingStuff(InputStream in)
1011    {
1012      int b;
1013      char sniff4it[] = {'I','D','=','"','H','I','E','R','A','R','C','H','Y','"','>'};
1014      int cntr = -1;
1015      boolean harkening = false;
1016
1017      try
1018      {
1019        while ((b = in.read()) != -1)
1020        {
1021          if (!harkening && (char)b == sniff4it[0])
1022          {
1023            cntr = 0;
1024            harkening = true;
1025            continue;
1026          }
1027
1028          if (harkening)
1029          {
1030            cntr++;
1031
1032            if ((char)b == sniff4it[cntr])
1033            {
1034              if ((char)b == '>')
1035              {
1036                return;
1037              }
1038              else
1039              {
1040                continue;
1041              }
1042            }
1043            else
1044            {
1045              cntr = -1;
1046              harkening = false;
1047            }
1048          }
1049        }
1050      }
1051      catch (Exception e)
1052      {
1053        panel.troubleShooting("Error:␣eatUninterestingStuff␣read␣failure");
1054      }
```

```
1055    }
1056
1057    private String getMotherOfMine(InputStream in)
1058    {
1059      int b;
1060      StringBuffer hierBuf = new StringBuffer(512);
1061      char sniff4it[] = {'/','D','I','V','>'};
1062      int cntr = -1;
1063      boolean harkening = false;
1064
1065      try
1066      {
1067        while ((b = in.read()) != -1)
1068        {
1069          hierBuf.append((char)b);
1070
1071          if (!harkening && (char)b == sniff4it[0])
1072          {
1073            cntr = 0;
1074            harkening = true;
1075            continue;
1076          }
1077
1078          if (harkening)
1079          {
1080            cntr++;
1081
1082            if ((char)b == sniff4it[cntr])
1083            {
1084              if ((char)b == '>')
1085              {
1086                break;
1087              }
1088              else
1089              {
1090                continue;
1091              }
1092            }
1093            else
1094            {
1095              cntr = -1;
1096              harkening = false;
1097            }
1098          }
1099        }
1100      }
1101      catch (IOException e)
1102      {
1103        panel.troubleShooting("Error: getMotherUnitLabel read failure");
1104      }
1105
1106      // hierBuf now contains the hierarchy information, address and name of parent
1107      String debug = new String(hierBuf.toString());
1108      StringTokenizer st = new StringTokenizer(hierBuf.toString());
1109
1110      StringBuffer garbage = new StringBuffer();
1111      StringBuffer address = new StringBuffer();
1112      StringBuffer name = new StringBuffer();
1113
1114      while (true)
1115      {
1116        garbage.setLength(0);
1117        garbage.append(st.nextToken("\""));
```

```
1118        address.setLength(0);
1119        address.append(st.nextToken("\""));
1120        garbage.setLength(0);
1121        garbage.append(st.nextToken("\""));
1122        garbage.setLength(0);
1123        garbage.append(st.nextToken("\""));
1124        name.setLength(0);
1125        name.append(st.nextToken("<"));
1126
1127        if (st.countTokens() <= 2)
1128          break;
1129      }
1130
1131      String pureName = (name.toString()).substring(2);
1132      return address.toString() + "|" + pureName +"$" + "STRUCTUREONLY";
1133    }
1134
1135    private String getNextChildOfMine(BufferedInputStream in)
1136    {
1137      StringBuffer candidate = new StringBuffer();
1138      int b;
1139      boolean harkening = false;
1140      boolean somethingFound = false;
1141      char sniff4it[] = {'<','/','A','>'};
1142      int cntr = -1;
1143
1144      try
1145      {
1146        while ((b = in.read()) != -1)
1147        {
1148          candidate.append((char)b);
1149
1150          if (!harkening && (char)b == sniff4it[0])
1151          {
1152            cntr = 0;
1153            harkening = true;
1154            continue;
1155          }
1156
1157          if (harkening)
1158          {
1159            cntr++;
1160
1161            if ((char)b == sniff4it[cntr])
1162            {
1163              if ((char)b == '>')
1164              {
1165                somethingFound = true;
1166                break;
1167              }
1168              else
1169              {
1170                continue;
1171              }
1172            }
1173            else
1174            {
1175              cntr = -1;
1176              harkening = false;
1177            }
1178          }
1179        }
1180      }
```

```
1181      catch (Exception e)
1182      {
1183        panel.troubleShooting("Error: getNextChildOfMine read failure");
1184      }
1185
1186      if (somethingFound == false) // Past last node
1187      {
1188        return "UNDEFINED";
1189      }
1190      else // Figure out child description from candidate
1191      {
1192        // Cut away leading stuff before http://.....
1193        String strStep1 = candidate.toString();
1194        int idx = strStep1.indexOf("http://");
1195        String strStep2 = strStep1.substring(idx);
1196        int idxHTTPend = strStep2.indexOf("\"");
1197        StringBuffer resultURL = new StringBuffer(strStep2.substring(0,idxHTTPend)); //
              Got URL
1198        int idxNameBegin = strStep2.indexOf("return true\">") + 13;
1199        int idxNameEnd = strStep2.indexOf("</A>");
1200        String resultName = strStep2.substring(idxNameBegin,idxNameEnd); // Got Name
1201
1202        StringBuffer resultInfo = new StringBuffer();
1203        StringBuffer pureName = new StringBuffer();
1204        int idxUglyTail;
1205
1206        if ((idxUglyTail = (resultName.toString()).indexOf("...")) == -1)
1207        {
1208          pureName.append(resultName);
1209          resultInfo.append("TERMINALTEXT");
1210        }
1211        else
1212        {
1213          pureName.append(resultName.substring(0,idxUglyTail));
1214          resultInfo.append("STRUCTUREONLY");
1215        } // Got Unit Type
1216
1217        return resultURL + "|" + pureName + "$" + resultInfo;
1218      }
1219    }
1220
1221    private String labelSubString(String combinedUnitDescription)
1222    {
1223      int idxBegin = combinedUnitDescription.indexOf("|");
1224      int idxEnd = combinedUnitDescription.indexOf("$");
1225
1226      // Special characters of the german language must be deHTMLed
1227      String replacePatterns[] = {"&Auml;","&auml;","&Ouml;","&ouml;","&Uuml;","&uuml;"};
1228      char replaceChars[] = {'Ä','ä','Ö','ö','Ü','ü'};
1229      String raw = new String(combinedUnitDescription.substring(idxBegin+1,idxEnd));
1230      StringBuffer parseMe = new StringBuffer(raw);
1231      StringBuffer cooked = new StringBuffer();
1232      int idxb;
1233      int idxe;
1234
1235      for (int i = 0;i < 6;i++)
1236      {
1237        idxb = idxe = 0;
1238        cooked.setLength(0);
1239
1240        while ((idxe = (parseMe.toString()).indexOf(replacePatterns[i])) != -1)
1241        {
1242          cooked.append((parseMe.toString()).substring(idxb,idxe));
```

```
1243            cooked.append(replaceChars[i]);
1244            idxb = idxe + 6;
1245            String nextPart = new String((parseMe.toString()).substring(idxb));
1246            parseMe.setLength(0);
1247            parseMe.append(nextPart);
1248          }
1249
1250          String parseMeTail = new String(parseMe.toString());
1251          parseMe.setLength(0);
1252          parseMe.append(cooked.toString());
1253          parseMe.append(parseMeTail);
1254        }
1255
1256        return parseMe.toString();
1257      }
1258
1259      private String urlSubString(String combinedUnitDescription)
1260      {
1261        int idxEnd = combinedUnitDescription.indexOf("|");
1262        return combinedUnitDescription.substring(0,idxEnd);
1263      }
1264
1265      private boolean infoSubStringIsTerminal(String combinedUnitDescription)
1266      {
1267        if (-1 != combinedUnitDescription.indexOf("$TERMINALTEXT"))
1268          return true;
1269        else
1270          return false;
1271      }
1272
1273      private String access2meta(String access)
1274      {
1275        int cut = access.indexOf("?sessionKey");
1276        String prepared = access.substring(0,cut);
1277        int idxBeginChangePart = prepared.indexOf("/khs/access/");
1278        int idxEndChangePart = idxBeginChangePart + 12;
1279        String serverandport = prepared.substring(0,idxBeginChangePart);
1280        String unchangedTail = prepared.substring(idxEndChangePart);
1281
1282        return serverandport + "/walterskhs/meta/" + unchangedTail;
1283      }
1284
1285      public String meta2bare(String meta)
1286      {
1287        int idxBeginChangePart = meta.indexOf("/walterskhs/meta/");
1288
1289        if (idxBeginChangePart == -1)
1290        {
1291          return meta; // Leave alone special URLs
1292        }
1293
1294        int idxEndChangePart = idxBeginChangePart + 16;
1295        String serverandport = meta.substring(0,idxBeginChangePart);
1296        String unchangedTail = meta.substring(idxEndChangePart);
1297
1298        return serverandport + "/walterskhs/access/" + unchangedTail;
1299      }
1300
1301      public void start()
1302      {
1303        panel.start();
1304      }
1305
```

```
1306     public void stop()
1307     {
1308       panel.stop();
1309     }
1310   }
```

B. Isometrics^S-Fragebögen

Bitte weiterblättern, es wurden die Originale mit abgeheftet.