

# „DreamView“ - ein kooperativer Webbrowser für eine synchrone CSCW-Umgebung

Jörg Roth  
Fernuniversität Gesamthochschule in Hagen  
Fachbereich Informatik  
Praktische Informatik II  
Joerg.Roth@Fernuni-Hagen.de

**Zusammenfassung:** Über das World Wide Web sind riesige Mengen von Informationen verfügbar, die über Webbrowser abrufbar sind. Herkömmliche Webbrowser sind für die Bedienung durch einen einzelnen Benutzer gedacht. Es ist interessant, einen Webbrowser auch im Rahmen einer synchronen Kooperation zur Verfügung zu stellen. Dieses Paper beschreibt den kooperativen Webbrowser *DreamView*. *DreamView* kann benutzt werden, um in einer Gruppensitzung Seiten aus dem World Wide Web zu laden oder um gemeinsam erstellte HTML-Dokumente zu diskutieren. Neben den üblichen Elementen eines Webbrowsers besitzt *DreamView* einige Komponenten, die speziell für die Benutzung in der Gruppe konzipiert wurden. So gibt es Fenster zur Verfolgung des Seitenausschnitts anderer Gruppenmitglieder und verteilte Mauszeiger. Ein besonderes Merkmal stellt die Möglichkeit dar, in der Gruppe Bemerkungen an den Rand einer Webseite anzubringen, die mit Elementen der Webseite verknüpft werden können.

## **1 Einleitung**

Das World Wide Web hat in den letzten Jahren zu einem Aufschwung in der weltweiten Informationsverbreitung beigetragen. Durch das World Wide Web kann auf einfache Weise auf riesige Mengen von Informationen zugegriffen werden und diese können unter einer einfach zu bedienenden Oberfläche dargestellt werden. Die Bedienung von Webbrowsern gehört mittlerweile zum Grundwissen eines Benutzers im Umgang mit Computern.

Webseiten, die in der Seitenbeschreibungssprache HTML [W3] verfaßt sind, eignen sich nicht nur zur Verbreitung von Informationen im Internet. Es bietet sich an, HTML-Seiten auch als Basis für die Bearbeitung von Dokumenten allgemein zu verwenden. So werden beispielsweise schon Dokumentationen zu Softwarewerkzeugen im HTML-Format ausgeliefert.

Interessant wäre es, die Benutzung eines Webbrowsers auch im Zusammenhang mit CSCW-Systemen (Computer Supported Cooperative Work) zu ermöglichen. Für den Einsatz eines kooperativen Browsers sind unter anderem folgende Szenarien denkbar:

- Kooperatives Browsen: Für die Sitzung relevante Webserver werden besucht und geladene Seiten für alle Teilnehmer dargestellt. Für diesen Einsatz ist ein kooperativer Webbrowser als ein Werkzeug in einer ganzen Sammlung verteilter Anwendungen denkbar. In diesem Fall wäre die Gruppe mit einer Aufgabe betraut, die nicht unbedingt etwas mit dem World Wide Web zu tun hat. Von Zeit zu Zeit wird aber der Browser als Hilfsmittel hervorgeholt, um Information zu laden.
- Diskussion von Dokumenten: Eine Gruppe kann ein Dokument oder einen ganzen Hypertext-Baum diskutieren. Hierbei ist es nicht erforderlich, daß die HTML-Seiten auf einem Webserver vorliegen, sie können auch lokal gespeichert sein. In diesem Szenario ist der Webbrowser die Hauptanwendung in der Sitzung.
- Computerunterstützter Vortrag: Ein Vortrag wurde als Sammlung von HTML-Seiten entwickelt, der nun in einer Gruppensitzung vorgeführt werden soll. Hierzu ist es notwendig, daß alle Zuschauer ihre jeweiligen Browser in einen Verfolgungsmodus bringen können, damit die Ausführungen des Vortragenden beobachtet werden können.

Ein kooperativer Webbrowser stellt als verteilte Anwendung eine Besonderheit dar. Neben der Vernetzung der Gruppenteilnehmer untereinander sind alle mit dem World Wide Web verbunden. Es finden also gewissermaßen Netzwerkaktivitäten in zwei Richtungen statt.

Bei der Einbettung synchroner Kooperation in einen Webbrowser gibt es mehrere Möglichkeiten:

#### Variante I:

Man verwendet einen handelsüblichen Webbrowser (z.B. Netscape) und bettet zusätzliche Funktionalitäten ein. Diese können z.B. in Form von Plug-Ins oder Applets realisiert werden. Zusätzlich wird das Protokoll zum Webserver erweitert, ggf. durch die Bereitstellung zusätzlicher Kommunikationskanäle. Hierüber können gruppenspezifische Kontrollfunktionen abgewickelt werden. So muß beispielsweise dem Umstand Rechnung getragen werden, daß das HTTP-Protokoll [W3b] zustandslos ist. Benutzer sind auch bezüglich des Webserver nicht dauerhaft angemeldet, d.h. die Verbindung zwischen Browser und Server wird nur zum Laden einer Seite aufgebaut und sofort wieder beendet. Auch kann ein Webserver nicht aktiv Ereignisse an die Browser melden. Solche Schwächen des HTTP-Protokolls müssen durch zusätzliche Komponenten aufgefangen werden. Zwei Ansätze dieser Kategorie sind in [WR94] und [FLF94] beschrieben.

#### Variante II:

Man nutzt die Eigenschaft bestimmter Browser, sich von außen steuern zu lassen. So bietet beispielsweise der Mosaic-Browser [NCSAc] eine CCI-Schnittstelle (*Common Client Interface*) an, mit dem man Seitenaufrufe von außen über einen Datenkanal anweisen kann. Zusätzlich zum Browser benötigt man eine Anwendung, die die Gruppenaktivitäten koordiniert und den Browser mit den entsprechenden Kommandos versorgt.

#### Variante III:

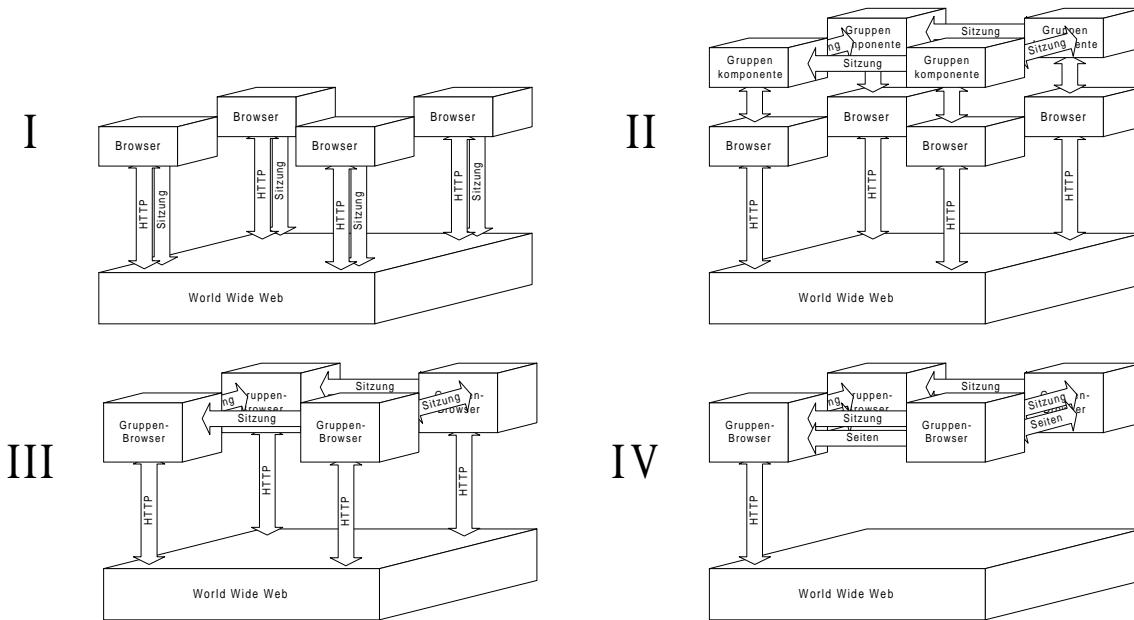
Ein Browser wird komplett als verteilte Anwendung neu entwickelt. Das würde die gesamte Browser-Funktionalität samt Abwicklung des HTTP-Protokolls und HTML-Parsing beinhalten.

#### Variante IV:

Wie Variante III, nur daß das HTTP-Protokoll über einen ausgewählten Teilnehmer abgewickelt wird. Der Vorteil dieser Variante ist, daß das Problem der Konsistenzerhaltung sehr effektiv gelöst werden kann. Der gravierende Nachteil ist jedoch, daß der Teilnehmerarbeits-

platz, der die Seiteninformationen vom Webserver lädt, besonders ausfallsicher sein muß. Fehler von dieser Seite würden die komplette Sitzung gefährden. Zusätzlich stellt dieser Arbeitsplatz bzgl. der Performance einen Flaschenhals dar, da alle Seiten von ihm an die Gruppe verteilt werden. Da Webseiten oft über große Datenmengen in Form von Bildern verfügen, ist diese Vorgehensweise als kritisch zu bewerten.

Die vier Möglichkeiten sind in der folgenden Abbildung schematisch dargestellt.



**Abbildung 1: Varianten bei der Realisierung eines kooperativen Webbrowsers**

Der entscheidende Vorteil der ersten beiden Varianten ist, daß bestehende Webbrowser verwendet werden können. Gerade kommerziell entwickelte Browser haben einen hohen Entwicklungsgrad erreicht und sind weit verbreitet. Die Geschlossenheit vorhandener Browserkonzepte erweist sich dagegen als gravierender Nachteil: man ist im wesentlichen auf die vorhandenen Funktionen eines Browsers angewiesen, da Erweiterungen schwierig bis unmöglich sind.

Variante I ist sicher vom Realisierungsaufwand gesehen die einfachste, hat aber den Nachteil, daß die komplette Gruppenkommunikation über den Webserver läuft. Da die Gruppenaktivitäten im Kontext des Browsers dargestellt werden, sind hier Einschränkungen möglich. So unterliegen beispielsweise Applets gewissen Sicherheitsschranken, so daß nicht der volle Sprachumfang genutzt werden kann.

In der Variante II können zumindest die Teilnehmer direkt miteinander kommunizieren. Der Anwender wird allerdings mit zwei Anwendungen konfrontiert. Neben der eigentlichen Browseranwendung muß eine Gruppenanwendung die notwendigen Gruppenaktivitäten vermitteln. Diese Zweiteilung kann den Anwender u.U. verwirren. Aus Anwendersicht hat man es mit einer Gruppenaktivität zu tun, die Trennung in zwei Teilanwendungen erfolgt nur aus technischen Gesichtspunkten.

Variante III ist aus der Sicht des Gruppenbewußtseins optimal. Gruppenfunktionen sind direkt in den Browser integriert. Die gesamte Anwendung kann so gestaltet werden, daß ein Betrieb in einer Gruppenumgebung unterstützt wird. Der Realisierungsaufwand ist allerdings sehr hoch, da zunächst die grundlegenden Funktionen eines Browsers implementiert werden müssen.

Variante IV hat wie beschrieben einen entscheidenden Nachteil bezüglich Performance und Ausfallsicherheit. Im Rahmen interaktiver Elemente (Kapitel 2.6.3) wird sie noch einmal diskutiert.

## **2 DreamView**

### **2.1 Übersicht**

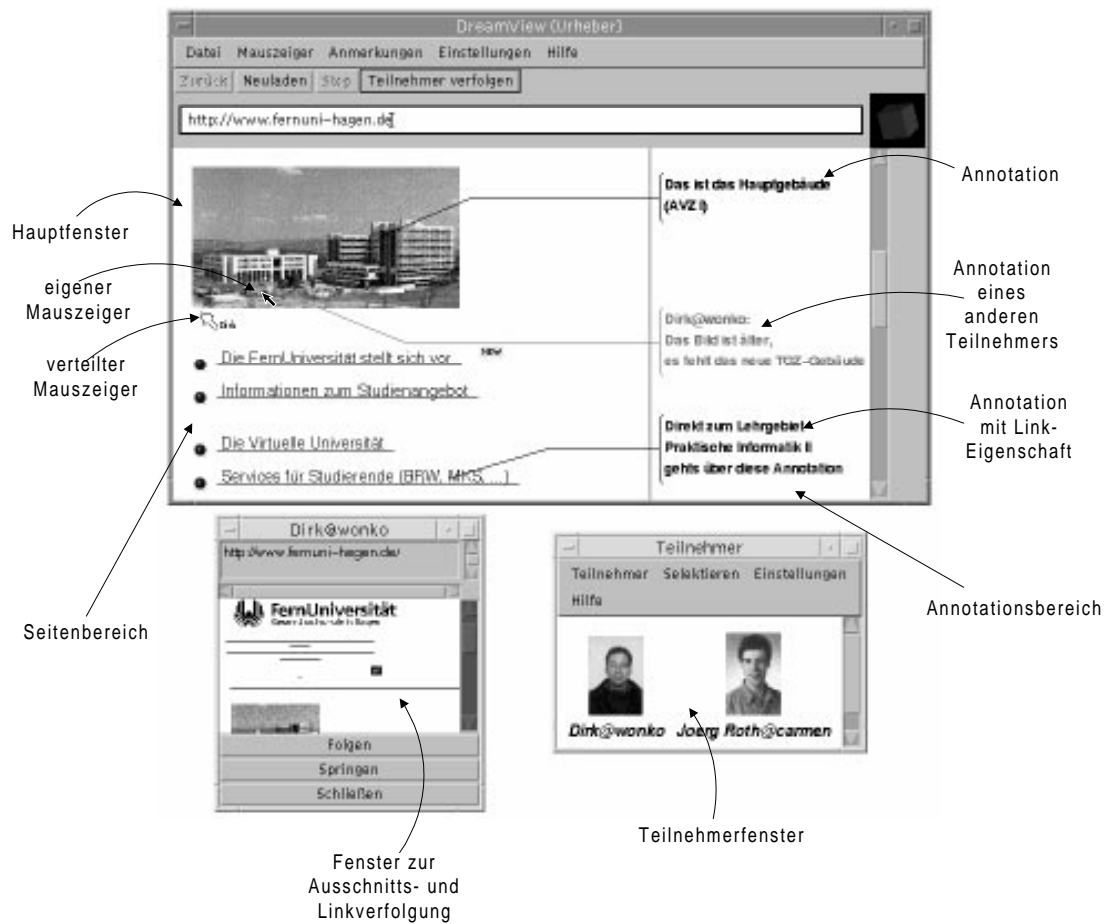
*DreamView* ist einerseits ein Webbrowser, andererseits eine Gruppenanwendung. Als Browser besitzt er die üblichen Funktionen zum Browsen von Seiten, sowohl lokal als auch im World Wide Web. Als Navigationshilfe werden die besuchten Seiten vermerkt. Eine Druckfunktion ist integriert.

Als verteilte Anwendung stehen gruppenspezifische Funktionen zur Verfügung. Da die meisten Gruppenfunktionen sich gut in eine allgemeine Bibliothek abspalten lassen und damit nicht für jede weitere verteilte Anwendung erneut realisiert werden müssen, bietet sich die Benutzung einer Plattform an. *DreamView* wurde mit Hilfe der CSCW-Plattform *DreamTeam* [Roth97] realisiert. *DreamTeam* ist eine Plattform für synchrone Gruppenarbeit. Der Aspekt der Synchronität bedeutet, daß Gruppenmitglieder sich zeitlich abstimmen müssen, um miteinander arbeiten zu können. *DreamTeam* wurde für einen dezentralen Betrieb konzipiert. Es gibt also keinen Server im System, der die Gruppenaktivitäten koordiniert. Neben der Netzinfrastruktur wird nur für jeden Teilnehmer ein Rechnerarbeitsplatz benötigt, um Gruppenarbeit zu ermöglichen. Der dezentrale Ansatz wurde gewählt, um bestimmte Nachteile zentraler Lösungen zu umgehen. So müssen beispielsweise zentrale Instanzen besonders ausfallsicher sein und stellen oft einen Flaschenhals dar.

Entwickler von verteilten Anwendungen werden durch eine Funktionsbibliothek und eine Laufzeitumgebung unterstützt. Wesentliche gruppenspezifische Elemente sind in der Umgebung vorhanden und können bei der Realisierung einer eigenen verteilten Anwendung als Building Blocks übernommen werden.

*DreamView* wurde gemäß Variante III (Abbildung 1) realisiert, d.h. es mußte ein kompletter Browser implementiert werden. Für die Browserfunktionen mußte eine Benutzungsoberfläche erstellt werden mit den entsprechenden Dialogfenstern zum Konfigurieren. Eine Komponente wickelt das HTTP-Protokoll ab. Der größte Aufwand entstand durch das Parsen der HTML-Beschreibungen sowie bei der graphischen Aufbereitung einer Webseite. All diese Bereiche würden auch bei der Entwicklung eines üblichen Webbrowsers zu berücksichtigen sein. Bei *DreamView* wurden zusätzliche Funktionen zur kooperativen Linkverfolgung, zur Synchronisation von Seitenausschnitten, zur Verwaltung verteilter Mauszeiger sowie zur Bearbeitung von Annotationen realisiert.

Die folgende Abbildung zeigt ein typische Arbeitsumgebung bei der Arbeit mit *DreamView*:



**Abbildung 2: Der Arbeitsbereich der DreamView-Anwendung**

Im Rahmen von Gruppensitzungen können beliebig viele verteilte Anwendungen offen sein. Hier ist jetzt der Fall dargestellt, daß der Browser die einzige offene Anwendung ist.

Neben den Fenstern der verteilten Anwendungen wird immer auch ein Teilnehmerfenster geöffnet (Abbildung 2 unten rechts). Dieses wird von der Umgebung bereitgestellt und ist nicht eigentlich Bestandteil der Anwendung.

Der Browser selbst stellt sich wie ein üblicher Webbrowser dar. Der größte Bereich wird durch die dargestellte Webseite in Anspruch genommen. Im Gegensatz zu anderen Webbrowsern ist ein Bereich für Annotationen abgetrennt worden. Im Annotationsbereich können textuelle Bemerkungen hinterlegt werden, die mit der HTML-Seite verankert werden können. Auf Wunsch kann der Annotationsbereich abgeschaltet werden, und die verfügbare Fläche fällt vollständig der Seitendarstellung zu.

Zur Verfolgung der Aktivitäten eines anderen Benutzers steht ein spezielles Dialogelement zur Verfügung (Abbildung 2 unten links). Ein kleines Fenster gibt in Form einer Übersicht den Seitenausschnitt eines anderen Teilnehmers auf seinem Browser wieder. Ein Benutzer kann die Aktivitäten jedes anderen Benutzers verfolgen, indem er entsprechend viele solcher Fenster öffnet.

In den folgenden Kapiteln wird konkret auf die Gruppenfunktionen des Browsers eingegangen. So wird das Konzept verteilter Mauszeiger, die Verfolgung von Seitenausschnitten und das Verteilungskonzept der Annotationen beschrieben. In einem weiteren Kapitel wird auf Realisierungsaspekte eingegangen.

## 2.2 Verteilte Mauszeiger

### 2.2.1 Relaxiertes WYSIWIS

Teilnehmer an einer Gruppensitzung bearbeiten denselben Informationsraum. Dies hat unmittelbar Auswirkungen darauf, wie Benutzer die gemeinsamen Daten wahrnehmen. Es gibt verschiedene Modelle. Ein häufiges ist *WYSIWIS* (What You See Is What I See). Hiermit wird ausgedrückt, daß jeder Benutzer exakt dieselbe Sicht auf die gemeinsamen Daten hat. Aus der Perspektive eines einzelnen Benutzer heißt das, daß jedes Ereignis oder jedes Objekt, auf das er sich in einer bestimmten Aktion bezieht, von allen anderen Teilnehmern auch exakt so wahrgenommen wird. Eine Form, in der das Aussehen auf allen Rechnern exakt gleich aussieht, wird *striktes WYSIWIS* genannt.

In der Regel hat jedoch jeder Benutzer unterschiedliche Vorlieben, seinen Arbeitsbereich einzurichten. Hieraus resultieren geringfügige Unterschiede in der Darstellung. Werden neben den öffentlichen Bereichen auch noch private Arbeitsbereiche benutzt, kann prinzipiell nicht mehr auf allen Rechnern exakt dieselbe Sicht vorausgesetzt werden. Hierbei spricht man dann von *relaxiertem WYSIWIS*. Bei dieser Form ist zwar der grobe Inhalt der verteilten Arbeitsbereiche auf allen Rechnern gleich, es können aber Unterschiede bzgl. der Fensterpositionen, der Skalierungen und der eingestellten Sichtbarkeitsbereiche vorliegen.

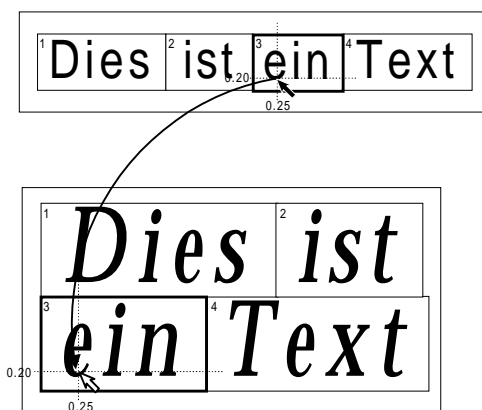
In der Regel wird diese Form von *WYSIWIS* bevorzugt, da eine exakt gleiche Darstellung auf allen Systemen einerseits nur schwer zu erreichen ist, andererseits nicht die Wünsche nach einer individuellen Arbeitsplatzgestaltung beachtet.

Im Falle eines kooperativen Webbrowsers führen unterschiedliche Fenstergrößen oder Schriften auch zu unterschiedlicher Formatierung. So wird beispielsweise Fließtext am Ende einer Zeile umgebrochen, auch wenn kein expliziter Zeilenvorschub vorliegt. Deshalb benötigt ein Text in einem schmalen Browserfenster mehr Zeilen als in einem breiten.

### 2.2.2 Verfahren zur Abbildung von Mauspositionen

Bei verteilten Mauszeigern ist man normalerweise nicht daran interessiert, welche x-y-Position die Maus eines anderen Teilnehmers einnimmt, sondern über welchem Objekt die Maus positioniert ist. Einfache Verfahren zur Abbildung von Mauspositionen, die auf der Übertragung der reinen x-y-Koordinaten basieren, führen nicht zu hilfreichen Ergebnissen. Deshalb wurde in *DreamView* ein Verfahren verwendet, das auf den Inhalten der Seiten operiert.

Dazu wird zuerst die gesamte Seite in rechteckige Einheiten unterteilt. Dies geschieht, indem um jedes Wort und jede Graphik eine Bounding Box gelegt wird. Diese Rechtecke werden



**Abbildung 3: Abbildung von Mauspositionen**

nun anhand ihrer Position innerhalb der HTML-Beschreibung durchnummeriert. Man erhält dadurch eine eindeutige Indizierung, die unabhängig von der resultierenden Anordnung im Seitenlayout ist. Möchte man eine Mausposition abbilden, so wird zuerst das Element unter der Maus ermittelt. Innerhalb des entsprechenden Rechtecks werden die relativen Koordinaten ermittelt. Die Elementnummer und die relativen Koordinaten werden nun an die anderen Teilnehmer übermittelt. Diese ermitteln den Ursprung des entsprechenden Elements und addieren die Anteile innerhalb des Elements hinzu. In unserem Beispiel schwebt der Mauszeiger unter dem „e“ des Wortes „ein“. Die Elementnummer

3 und die relativen Koordinaten werden übertragen. Auf dem Zielsystem wird der Ursprung des entsprechenden Elements ermittelt und anhand der relativen Koordinaten innerhalb des Elements die absolute Mausposition berechnet. In unserem Beispiel wird der Mauszeiger auch wieder unter dem „e“ positioniert.

Das Verfahren funktioniert für Bilder und Text gleichermaßen. Bei Text ist das Verfahren, unabhängig von der jeweiligen Schriftart hinreichend genau, solange nicht zwischen Proportional- und Monospaced-Schriften abgebildet wird. Da aber in einer HTML-Beschreibung eindeutig festgelegt wird, welche Passagen in Proportional- und welche in Monospaced-Schrift dargestellt werden, besteht hier kein Problem. Auf eine Unterteilung der Seite bis auf die Ebene einzelner Buchstaben wurde verzichtet, da dann ein beträchtlicher Verwaltungs- und Suchaufwand innerhalb einer Seite bei der Abbildung der Mauspositionen die Folge wäre. Da aber kontinuierliche Mausbewegungen in Echtzeit übertragen werden müssen, ist dieser Aufwand nicht vertretbar.

### **2.2.3 Einschränkung der Mauszeigerverteilung**

Verteilte Mauszeiger müssen nicht immer ein Nutzen sein. Gerade bei größeren Gruppen kann bei zu vielen Mauszeigern die Übersicht verlorengehen. Aus diesem Grund kann jeder Benutzer die Verteilung einschränken. Dazu stehen zwei Möglichkeiten zur Verfügung:

- Er kann festlegen, ob der eigene Mauszeiger an die anderen Teilnehmer verteilt werden soll oder nicht.
- Er kann festlegen, ob er andere Mauszeiger sehen möchte oder nicht.

Die erste Festlegung beeinflusst die eigene Privatsphäre. Es kann sein, daß man sich auch in einem öffentlichen Bereich mit der Maus bewegen möchte, ohne daß alle anderen Teilnehmer dies sehen sollen.

Die zweite Festlegung beeinflusst die Übersicht. Bei Detailarbeiten möchte ein Teilnehmer eventuell nicht von anderen Mauszeigern gestört werden und schaltet alle ab.

Durch diese Einstellungen wird jedem Teilnehmer die notwendige Kontrolle bei der Arbeit mit gemeinsamen Dokumenten gegeben.

## **2.3 Verfolgung von Seitenausschnitten**

### **2.3.1 Allgemeine Problematik**

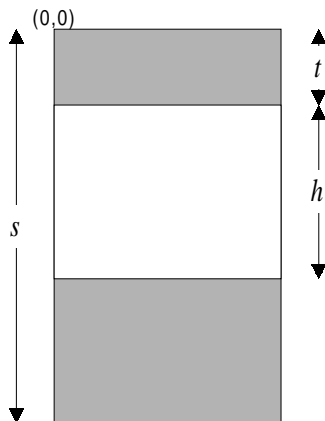
In der Regel sind Webseiten größer als die Bildschirmseite, die zur Darstellung zur Verfügung steht. Aus diesem Grund kann der Benutzer über Scrollbar-Funktionen den entsprechenden Seitenausschnitt einstellen.

In Gruppensitzungen kann es vorkommen, daß die Teilnehmer unterschiedliche Bildausschnitte auf der gleichen Seite eingestellt haben. Bei Diskussionen müssen sich alle Teilnehmer über die jeweils anderen Seitenausschnitte informieren können. Bezieht sich nämlich ein Teilnehmer auf ein bestimmtes Element aus seinem individuellen Seitenausschnitt, so müssen andere Teilnehmer diese Referenz auflösen können und ggf. bei sich den entsprechenden Ausschnitt einstellen können.

Analog zu den verteilten Mauszeigern ergibt sich jedoch durch das relaxierte WYSIWIS-Paradigma ein Problem. Durch unterschiedliche Fenstergrößen oder Schriftarten ist es in der Regel so, daß ein exakt gleicher Seitenausschnitt gar nicht eingestellt werden kann. Möchte ein Teilnehmer den Seitenausschnitt eines anderen Teilnehmers bei sich einstellen, so ist man darauf angewiesen, eine Abbildung vorzunehmen. Dieser Sachverhalt soll im folgenden formalisiert werden.

### 2.3.2 Ein Formalismus zur Beschreibung von Seitenausschnitten

Ein Seitenausschnitt  $S$  ist ein Tripel  $D = (s, t, h)$ , wobei  $s \in \mathbb{N}$ ,  $h \in \mathbb{N}$ ,  $t \in \mathbb{N}_0$ ,  $h \leq s$ ,  $t \leq s - h$ . Sei  $\mathbf{D}$  die Menge aller Tripel  $D$ .



**Abbildung 4: Variablen zur Beschreibung eines Ausschnitts**

Abbildung 4 zeigt die Bedeutung der Variablen:  $s$  ist die Länge des Gesamtdokuments (gemessen z.B. in Pixel des Ausgabegeräts).  $t$  ist der Abstand der oberen Position des Sichtbereichs vom Dokumentenanfang,  $h$  ist die Größe des Sichtbereichs.  $s$ ,  $t$  und  $h$  werden in denselben Einheiten gemessen. Verwendet man Pixel als Einheit, so gibt  $h$  die genaue Größe des Ausgabefensters an. Es wird davon ausgegangen, daß der Dokumentenausschnitt nur in vertikaler Richtung variabel ist und in horizontaler Richtung die komplette Breite des Dokuments umfaßt.

Da das Dokument auf unterschiedlichen System u.U. unterschiedlich dargestellt wird, resultieren daraus unterschiedliche Seitengrößen. Um aber eine einheitliche Bezugsgröße für die Positionierung im Dokument zu erhalten, wird der Dokumentenbereich auf das Intervall  $[0,1]$  abgebildet. Wir teilen dazu eine beliebige Position  $y$  im Dokument durch  $s$ .

Das Intervall  $v_D = \left[ \frac{t}{s}, \frac{t+h}{s} \right]$  gibt den sichtbaren Bereich des Dokuments an.

Die partiell definierte Funktion  $a_D(y): [0,1] \rightarrow [0,h]$ ,  $a_D(y) = \begin{cases} y \cdot s - t & \text{für } y \in v_D \\ \text{undef} & \text{sonst} \end{cases}$

beschreibt die Abbildung aus dem Dokumentensystem auf den Darstellungsbereich.

Gesucht wird nun eine Abbildung  $f: \mathbf{D} \rightarrow \mathbf{D}, (s_{remote}, t_{remote}, h_{remote}) \rightarrow (s_{local}, t_{local}, h_{local})$ , die in „sinnvoller“ Weise einen Seitenausschnitt eines fernen Benutzers lokal abbildet. Hierbei soll unter „sinnvoll“ folgendes verstanden werden:

- i)  $f(s_{remote}, 0, h_{remote}) = (s_{local}, 0, h_{local})$  d.h., wenn der ferne Benutzer an den Dokumentenanfang gesprungen ist, soll der lokale Benutzer auch den Anfang sehen.
- ii)  $f(s_{remote}, s_{remote} - h_{remote}, h_{remote}) = (s_{local}, s_{local} - h_{local}, h_{local})$  für  $s_{remote} - h_{remote} > 0$  d.h., wenn der ferne Benutzer an das Dokumentenende gesprungen ist, soll der lokale Benutzer auch das Ende sehen.
- iii) wenn  $\frac{h_{remote}}{s_{remote}} \leq \frac{h_{local}}{s_{local}}$ , dann soll gelten  $v_{remote} \subseteq v_{local}$ , sonst  $v_{local} \subseteq v_{remote}$ , d.h. alles, was der Benutzer mit dem kleineren relativen Seitenausschnitt sieht, wird auch vom anderen gesehen.

Seien  $\tilde{s}_{local}$  und  $\tilde{h}_{local}$  von der Umgebung des lokalen Benutzers fest vorgegebene Größen, d.h.  $\tilde{s}_{local}$  ist die Dokumentengröße, die durch lokal vorliegende Randbedingungen (z.B. Schriftgröße) resultiert.  $\tilde{h}_{local}$  ist die Fenstergröße, die der lokale Benutzer eingestellt hat.

Sei  $f$  die Funktion  $f: \mathbf{D} \rightarrow \mathbf{D}, (s_{remote}, t_{remote}, h_{remote}) \rightarrow (s_{local}, t_{local}, h_{local})$  mit

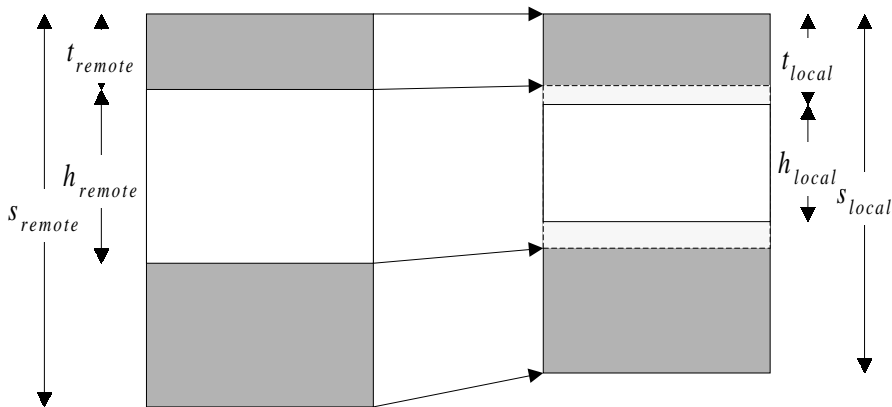
$$s_{local} = \tilde{s}_{local},$$

$$h_{local} = \tilde{h}_{local} \text{ und}$$

$$t_{local} = \begin{cases} \frac{s_{local} - h_{local}}{s_{remote} - h_{remote}} \cdot t_{remote} & \text{für } s_{remote} > h_{remote} \\ t_{remote} & \text{sonst} \end{cases}$$

Dann erfüllt  $f$  die Bedingungen *i)* bis *iii)*, ist also eine sinnvolle Abbildung von Seitenausschnitten.

Die folgende Abbildung stellt die Zusammenhänge dar:



**Abbildung 5: Zusammenhänge der Abbildung  $f$**

Angenommen der ferne Benutzer hat den größeren relativen Seitenausschnitt eingestellt. Eine lineare Umrechnung würde zu dem gestrichelten Bereich führen. Hiermit würde der lokale Bereich exakt den fernen Bereich wiedergeben. Der lokale Ausschnitt ist jedoch durch die individuell eingestellte Größe  $h_{local}$  vorgegeben.  $t_{local}$  wird nun durch die Abbildung so berechnet, daß der sichtbare Bereich innerhalb des gestrichelten Bereichs liegt.

### **2.3.3 Ausschnitts- und Linkverfolgung mit dem Document Tracker**

Dieses Ergebnis ist in die Konzeption eines neuen Dialogelements eingeflossen. In *Dream-View* wurde für die Ausschnittsverfolgung ein Dialogelement namens *Document Tracker* entwickelt.



**Abbildung 6: Der Document Tracker**

Abbildung 6 zeigt einen geöffneten *Document Tracker*. Dieses Fenster kann gleichermaßen zur Orientierung und zur Synchronisation von Ausschnitten verwendet werden. Es wird immer der Seitenausschnitt des anderen Teilnehmers im Verhältnis 1:3 dargestellt. Die dargestellte Scrollbarposition entspricht genau der des fernen Teilnehmers. Ein Teilnehmer kann zu jedem anderen Teilnehmer ein solches Fenster öffnen und sich so einen Überblick über den jeweils eingestellten Seitenausschnitt verschaffen. Hierzu muß nur der andere Teilnehmer aus der Teilnehmerliste ausgewählt werden. Das Fenster kann beliebig auf der Arbeitsoberfläche positioniert werden.

Über zwei Knöpfe kann man sich mit anderen Teil-

nehmern synchronisieren. Mit „Folgen“ gibt man an, daß sowohl jede Linkverfolgung als auch jedes Scrolling des anderen Benutzers auch lokal durchgeführt werden soll. Diese Verfolgung kann zu beliebigen Zeiten aufgegeben werden, um individuell weiterzuarbeiten. Mit „Springen“ wird einmalig der Ausschnitt synchronisiert. Sowohl die einmalige, als auch die ständige Synchronisation von Seitenausschnitten wirkt sich unmittelbar auf das Hauptfenster der Anwendung aus. Einmal aktiviert, wird ständig der Scrollbar des Hauptfensters und damit der gesamte Seitenausschnitt mit jeder Seitenbewegung des fernen Teilnehmers aktualisiert. Hierbei wird die Abbildung des Ausschnitts mit der im letzten Kapitel vorgestellten Funktion  $f$  durchgeführt.

## **2.4 Annotationen**

### **2.4.1 Annotationen in DreamView**

Zur Zeit bestehen Annotationen in DreamView aus Texten, die mit Seitenelementen verknüpft werden können. Über eine abknickende Linie wird der Text mit der Position auf der Seite verbunden. Diese Linie kann auf Wunsch auch weggelassen werden. Annotationen können selbst als Links verwendet werden. Neben dem Annotationstext wird dann eine URL hinterlegt. Über das Browserfenster können die Links hinter den Annotationen bequem verfolgt werden. Jeder Benutzer kann zu jeder beliebigen Seite Bemerkungen eintragen. Da Webseiten von Browsern in der Regel nicht zurückgeschrieben werden können, müssen die Bemerkungen getrennt von der eigentlichen Seite gespeichert werden. Eine Möglichkeit zur Speicherung wäre eine zentrale Datenbank, auf die alle Teilnehmer zugreifen. Zu den Realisierungskonzepten der *DreamTeam*-Umgebung gehört aber eine dezentrale Organisation, so daß sich der Einsatz einer zentralen Instanz zur Datenhaltung verbietet. Deshalb wurde ein eigener Ansatz verfolgt, der im folgenden beschrieben wird.

### **2.4.2 Data Bags als Konzept lose gekoppelter benutzerspezifischer Daten**

Im wesentlichen ist ein *Data Bag* eine verteilte, dezentral organisierte Datenbank mit bestimmten Eigenschaften. Ein *Data Bag* ist die Vereinigung aller lokalen Datenbestände der an der Sitzung teilnehmenden Benutzer. Die Menge der Datenbestände kann dabei variieren, da jeder Benutzer seine Daten nur dann zum Gesamtbestand zufügt, wenn er der Sitzung beigetreten ist. Ein *Data Bag* hat folgende Eigenschaften:

- Ein *Data Bag* besteht aus Datensätzen, die alle dieselbe Struktur haben.
- Jedem Datensatz ist genau ein Teilnehmer zugeordnet, nämlich derjenige, der ihn generiert hat.
- Ein Datensatz kann nur von dem Teilnehmer gelöscht oder verändert werden, von dem er generiert wurde.
- Tritt ein Teilnehmer aus einer Sitzung aus, so werden auch alle ihm zugeordneten Datensätze aus dem *Data Bag* entfernt.
- Tritt ein Teilnehmer einer Sitzung bei, so fügt er alle ihm zugeordneten Datensätze hinzu.

Wie man leicht sieht, kann der strenge Konsistenzgedanke von Datenbanken hier nicht aufrechterhalten werden. Gerade die Haupteigenschaften von Datenbanktransaktionen, abgekürzt mit *ACID* (atomicity, consistency, isolation, durability), treffen nicht zu. Es gibt allerdings Anwendungsfälle, in denen diese Einschränkungen akzeptiert werden können. Die Verwaltung von gemeinsamen Annotationen zu Seiten ist ein solcher Fall. Ein *Data Bag* eignet sich dann besonders, wenn die Abfrage von Daten nach bestimmten Suchkriterien die am häufigsten benutzte Operation ist und Datenänderungen eher selten sind. Genau diese Eigenschaften gelten für die Annotationen. Die häufigste Funktion ist der Aufbau einer Seite. Hierbei wer-

den die Annotationen aller Teilnehmer erfragt. Das Suchkriterium ist dabei, daß eine Annotation zu der aktuellen Webseite erstellt wurde. Annotationen zu anderen Seiten sind nicht von Interesse und sollen herausgefiltert werden. Solange eine Seite aufgebaut ist, bleibt der Suchfilter gültig. Erst wenn die Seite gewechselt wird, muß der Datenbestand erneut durchsucht werden.

Die Abfrageoperation soll anhand der oben gemachten Bedingungen effizient durchgeführt werden. Die Vorgehensweise soll nun für den allgemeinen Fall dargelegt werden. Dazu werden folgende Definitionen getroffen:

Sei  $n \in \mathbb{N}$  die Anzahl der zur Zeit teilnehmenden Benutzer,  $D_i$  die Daten eines Benutzers  $i$  ( $0 < i \leq n$ ) und  $\mathbf{D} = \bigcup_{0 < i \leq n} D_i$  die Menge aller zur Zeit zugreifbaren Daten. Ein Benutzer  $i$  kann

folgende Operationen auf seinen Daten durchführen:

- Hinzufügen eines Datenobjektes  $d$ :  $Add(D_i, d)$   
 Vorbedingung:  $d \notin D_i$   
 Nachbedingung:  $d \in D_i$
- Löschen eines Datenobjektes  $d$ :  $Delete(D_i, d)$   
 Vorbedingung:  $d \in D_i$   
 Nachbedingung:  $d \notin D_i$
- Ändern eines Datenobjektes  $d_1$  in  $d_2$ :  $Change(D_i, d_1, d_2)$   
 Vorbedingung:  $d_1 \in D_i, d_2 \notin D_i$   
 Nachbedingung:  $d_1 \notin D_i, d_2 \in D_i$

Zur Abfrage von Datenbeständen kann ein beliebiges Prädikat  $q_i$  verwendet werden mit:

$q_i(d)$  gdw. das Datenelement  $d$  soll in dem Ergebnis der Abfrage des Benutzers  $i$  enthalten sein.

Ein Abfrageergebnis ist somit definiert durch  $Q(q_i, \mathbf{D}) = \{d \in \mathbf{D} | q_i(d)\}$ . Das Verfahren zur Bestimmung dieser Menge soll nun bezüglich der Kosten optimiert werden. Dazu definieren wir  $R_{i,j} := \{d \in D_j | q_i(d)\}$  und zerlegen  $Q$  folgendermaßen:

$$Q(q_i, \mathbf{D}) = R_{i,i} \cup \bigcup_{0 < j \leq n, j \neq i} R_{i,j},$$

d.h. in einen „lokalen“ und einen „fernen“ Anteil. Der erste Teil wird durch die Menge der Daten bestimmt, die dem Benutzer zugeordnet sind, der auch die Abfrage gemacht hat. Da diese Daten lokal vorliegen, ist die Bestimmung dieser Menge nicht kostenintensiv. Der ferne Anteil besteht aus den Daten der restlichen Benutzer. Änderungen der Datenbestände dieser Benutzer wirken sich unmittelbar auf das Abfrageergebnis aus. Man möchte sich allerdings ersparen, bei jeder Datenänderung durch einen Benutzers  $j$  die komplette Menge  $R_{i,j}$  neu zu berechnen und zum Benutzer  $i$  zu übertragen.

Durch folgende Methode kann dies vermieden werden:

Das Prädikat  $q_i$  wird an alle Benutzer  $j \neq i$  übertragen. Zusätzlich werden die datenverändernden Operationen eines Benutzer  $j$  erweitert:

- Hinzufügen eines Datenobjekt:  $Add(D_j, d)$   
 Vorbedingung:  $d \notin D_j$   
 Nachbedingung:  $d \in D_j$ , wenn  $q_i(d)$  dann  $d \in R_{i,j}$
- Löschen eines Datenobjekts:  $Delete(D_j, d)$   
 Vorbedingung:  $d \in D_j$   
 Nachbedingung:  $d \notin D_j$ , wenn  $q_i(d)$  dann  $d \notin R_{i,j}$
- Ändern eines Datenobjekts:  $Change(D_j, d_1, d_2)$   
 Vorbedingung:  $d_1 \in D_j, d_2 \notin D_j$

Nachbedingung:  $d_1 \notin D_j, d_2 \in D_j$ , wenn  $q_i(d_1)$  dann  $d_1 \notin R_{i,j}$ , wenn  $q_i(d_2)$  dann  $d_2 \in R_{i,j}$

Damit die Menge  $R_{i,j}$  nicht jedesmal zum Benutzer  $i$  erneut übertragen werden muß, wird sie beim Benutzer  $i$  gehalten und es werden nur die Änderungen übertragen. Das führt dazu, daß bei den Operationen  $Add(D_j, d)$  und  $Delete(D_j, d)$  maximal der Datensatz  $d$  übertragen werden muß, und bei der Operation  $Change(D_j, d_1, d_2)$  die Sätze  $d_1$  und  $d_2$ . Im günstigen Fall, wenn nämlich das Prädikat  $q$  auf die entsprechenden Datensätze nicht zutrifft, wird nichts übertragen. Das ist gegenüber der Übertragung der Menge  $R_{i,j}$  nach jeder Operation eine beträchtliche Reduktion des Datenaufkommens.

Wenden wir diese Ergebnisse nun wieder auf die Annotationen in *DreamView* an. Ein Benutzer  $i$  öffnet eine beliebige Seite  $p$  und möchte dazu alle in der Gruppe definierten Annotationen sichten. Das entsprechende Suchkriterium  $q_i$  lautet:

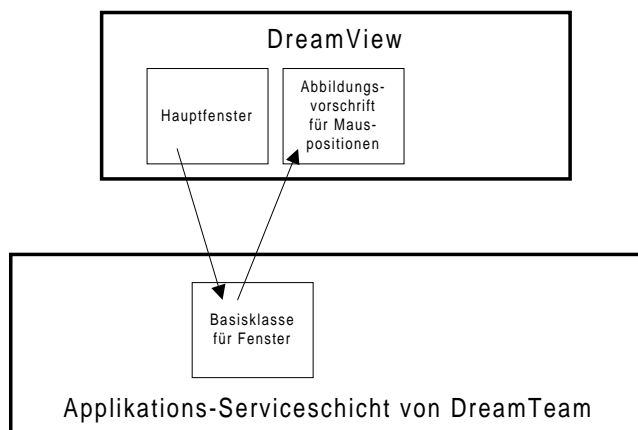
$q_i(d)$  gdw. die Annotation  $d$  wurde für die Seite  $p$  eingetragen

$q_i$  wird nun an alle Teilnehmer  $j$  ( $j \neq i$ ) übertragen, worauf diese jeweils die Menge  $R_{i,j}$  zurückliefern. Hierauf baut der Benutzer  $i$  die Seite mit allen Annotationen auf. Angenommen diese Seite bleibt nun für einen bestimmten Zeitraum offen. Der Benutzer  $i$  möchte aber immer den aktuellen Stand sehen, ungeachtet der Änderungen, die andere Benutzer an den Annotationen vornehmen. Angenommen, ein anderer Benutzer  $k$  hat dieselbe Seite offen und ändert eine Annotation  $a_1$  zu  $a_2$ . Er hat immer noch das Suchkriterium des Benutzers  $i$  geladen. Durch Auswertung von  $q_i(a_2)$  stellt er fest, daß diese Annotation eine Seite betrifft, die ein anderer Benutzer gerade offen hat. Daraufhin verschickt er  $a_1$  und  $a_2$  an den Benutzer  $i$ , der seine lokale Kopie von  $R_{i,j}$  aktualisiert und die entsprechenden Annotationen neu anzeigt.

## 2.5 Realisierungsaspekte

### 2.5.1 Verwendung der DreamTeam-Plattform

Sowohl *DreamView* als auch die gesamte *DreamTeam*-Umgebung wurden auf der Basis von Java [Sun] realisiert. Durch konsequente Ausnutzung der objektorientierten Programmierweise wurde der Aufwand für die Entwicklung einer verteilten Anwendung minimiert. Die wesentlichen Funktionen zur Gruppenarbeit liegen in der *DreamTeam*-Plattform schon vor, so daß nur jeweils der anwendungsspezifische Teil realisiert werden muß. Zugreifbar werden die Dienstleistungen der Umgebung über die Applikations-Serviceschicht [Roth97b]. Es werden beispielsweise verteilte Mauszeiger und das *Document Tracker*-Element zur Verfügung gestellt, sie müssen nur von der verteilten Anwendung integriert werden.



**Abbildung 7: Aufrufe für die Mauszeigerverteilung**

Werden sehr komplexe Elemente in eine allgemeine Plattform verlagert, ergibt sich allerdings ein Problem: die Konfigurierbarkeit. So ist es beispielsweise nicht möglich, einen verteilten Mauszeiger zu konstruieren, der für *alle* denkbaren Anwendungen benutzt werden kann. Spätestens bei der Abbildung der Mauspositionen über verschiedene Plattformen (siehe Kapitel 2.2.2) versagt ein allgemeines Konzept.

Für die *DreamTeam*-Plattform wurde daher folgendermaßen vorgegangen: die Plattform stellt allgemeine Funktionen zu Verfügung, die ihrerseits wieder über

Klassen der Anwendung konfiguriert werden. Die Abbildung 7 zeigt das Konzept am Beispiel verteilter Mauszeiger.

Die Anwendung möchte im Hauptfenster verteilte Mauszeiger benutzen. Sie ruft dazu eine Klasse der Plattform auf, die diese Dienstleistung erbringt. Die Umrechnung der Mauspositionen ist jedoch wieder anwendungsabhängig, so daß die Plattform ihrerseits wieder eine Klasse der Anwendung aufruft, um die Umrechnung durchzuführen.

Aufrufe aus der Plattform heraus in die Anwendung werfen ein Stabilitätsproblem auf. Anwendungen unterliegen der Programmierung durch den jeweiligen Anwendungsentwickler, müssen somit nicht von vorne herein fehlerfrei sein. Die Abarbeitung läuft aber im Kontext der Umgebung, potentiell könnten damit interne Prozesse unkontrolliert terminiert werden. Deshalb werden solche Aufrufe so durchgeführt, daß potentielle Fehler nicht die Umgebung beeinflussen können. Fehler, die während des Aufrufs auftreten werden für Auswertungszwecke protokolliert.

## **2.5.2 Bewertung des Realisierungsaufwandes**

Bei der Realisierung von *DreamView* wurden folgende Hauptbereiche ausgemacht:

- das Hauptprogramm mit dem Hauptfenster und allen Dialogen zur Konfiguration
- die Klassen zur Eingabe und Modifizierung von Annotationen sowie die entsprechenden Datenstrukturen
- Klassen für die Verwaltung von HTTP-Verbindungen, zum Parsen der HTML-Seiten und zum Laden der Bilder
- Graphische Aufbereitung und Darstellung einer Seite inklusive Abbildung der Mauskoordinaten

Die Zeilenzahl ist natürlich kein standardisiertes Maß für den Implementierungsaufwand. Sie soll aber zumindest einen Eindruck vermitteln.

<b>Bereich</b>	<b>Anzahl Klassen</b>	<b>Anzahl Zeilen</b>
<b>Hauptprogramm, Benutzungsoberfläche</b>	5	1568
<b>Annotationen</b>	3	685
<b>HTTP und HTML Parsing</b>	5	2075
<b>Graphische Aufbereitung einer Seite</b>	5	2914

Die synchronen Gruppenfunktionen sind nun in den oben vorliegenden Klassen eingebettet. Hierbei wurde folgende Anteile an der Implementierung erzielt.

<b>Verteilungsaspekt</b>	<b>Anzahl Zeilen</b>	<b>in welchem Bereich realisiert</b>
<b>Grundfunktionen der Verteilung, Initialisierung der Umgebung</b>	297	Hauptprogramm
<b>Verteilte Mauszeiger</b>	148	Graphische Aufbereitung einer Seite
<b>Ausschnitts- und Linkverfolgung</b>	71	Graphische Aufbereitung einer Seite, Benutzungsoberfläche
<b>Datenverwaltung der Annotationen</b>	31	Hauptprogramm

Setzt man die Aufwände zueinander ins Verhältnis, so erhält man:

7,6% des Sourcecodes wurden für Verteilungsaspekte verwendet

92,4% wurden für Funktionen aufgebracht, die auch für einen Ein-Benutzer-Browser notwendig gewesen wären

Es ergeben sich somit durch konsequente Ausnutzung der *DreamTeam*-Plattform effiziente Realisierungen verteilter Anwendungen.

## **2.6 Weitere Aspekte**

### **2.6.1 Privater Modus**

Anwendungen unter *DreamTeam* können auch in einem privaten Modus gestartet werden. Im privaten Modus wird die Anwendung so gestartet, als ob eine Sitzung nur aus einem einzigen Teilnehmer bestünde. Alle Funktionen, die sich auf Gruppenaktivitäten beziehen (verteilte Mauszeiger, *Document Tracker* etc.) sind gesperrt.

Im privaten Modus stellt sich *DreamView* wie ein üblicher Webbrowser dar. Die Annotationsfunktionen stehen zur Verfügung und operieren auf derselben Datenbasis wie im verteilten Modus.

### **2.6.2 Verweise auf file-Links**

Bei der kooperativen Benutzung von Browsern stellen die file-Links ein Problem dar. Im Gegensatz zu http-Links werden diese auf jedem System anders ausgewertet, da sie sich auf das lokale Dateisystem beziehen.

File-Links haben jedoch einen großen Vorteil. Große HTML-Verzeichnisse lassen sich auf CD-ROM verteilen oder können asynchron vor einer Sitzung heruntergeladen werden. Bei einer langsamen Anbindung ans Internet könnten somit in erträglicher Zeit kooperativ HTML-Dokumente betrachtet werden, da nur noch die Gruppenfunktionen über das Netzwerk laufen. Aus diesem Grund wurden in *DreamView* die kooperative Benutzung von Seiten mit file-Links ermöglicht. Um die Unterschiede der jeweiligen Dateisysteme zu überbrücken, gibt jeder Teilnehmer ein Stammverzeichnis vor, in dem die lokalen HTML-Seiten abgelegt werden. Hierbei können auch Unterverzeichnisse gebildet werden. Voraussetzung für das kooperative Arbeiten auf lokalen Seiten ist, daß jeder Teilnehmer in dem ausgewiesenen Unterverzeichnis dieselben Seiten- und Verzeichnisstruktur vorliegen hat.

Wird nun ein file-Link aufgelöst, so wird das für jeden Teilnehmer gültige Stammverzeichnis lokal ersetzt. Als Beispiel:

<b>Teilnehmer</b>	<b>Stammverzeichnis</b>	<b>file-Link</b>
<b>Teilnehmer1 (PC)</b>	D:\pages	file://D:\pages\page1.html
<b>Teilnehmer2 (UNIX)</b>	/home/roth/html	file:///home/roth/html/page1.html

Hier arbeitet der Teilnehmer1 auf einem PC und hat alle HTML-Dateien in ein Verzeichnis „D:\pages“ gelegt. Er referenziert jetzt den Link „file://D:\pages\page1.html“. Teilnehmer2 möchte denselben Link verfolgen. Bei der Übertragung wird nun das Stammverzeichnis von Teilnehmer1 in dem Link ersetzt durch das Stammverzeichnis des Teilnehmers2. Eventuelle Konventionen im Pfad (z.B. „\“ unter DOS entspricht „/“ unter UNIX) werden plattformspezifisch geändert.

### **2.6.3 Interaktive Elemente**

Die Unterstützung interaktiver Elemente (z.B. Formulare und Applets) fehlt in *DreamView* vollständig. Dies ist zum einen technisch begründet, da für diese Elemente ganz spezielle Mechanismen eingeführt werden müßten.

Der eigentliche Grund liegt aber darin, daß Formulare und Applets zu einer Interaktivität im Browser führen, die von der Umgebung nicht mehr behandelt werden kann. Dies soll an einem Beispiel erläutert werden:

Die Gruppe öffnet kooperativ eine Seite mit einem Formular. Dieses wird gemeinsam ausgefüllt und an den Webserver verschickt. Solange jeder Teilnehmer eine eigene HTTP-Verbindung zum Webserver verwendet, wird der Webserver von jedem Teilnehmer einzeln mit der Bearbeitung des Formulars beauftragt. Es ist aber in keiner Weise festgelegt, daß ein Webserver auf gleiche Aufträge exakt gleich antworten muß. So kann er z.B. je nach Identität des Absenders unterschiedliche Resultate zurückliefern, oder es werden Zufallselemente in die Bearbeitung eingebaut.

Es kann somit nicht mehr davon ausgegangen werden, daß alle Teilnehmer dasselbe Resultat erhalten. Hierdurch wird eine weitere kooperative Betrachtung der Seiten unmöglich, da jeder Teilnehmer sich auf seine lokale Sicht beziehen würde.

Die Einbindung von Applets hätte einen ähnlichen Effekt. Applets sind kleine Programme, die nicht für den Betrieb in einer Gruppenumgebung ausgelegt sind. Sie sind deshalb denkbar ungeeignet, im Rahmen einer Gruppensitzung gestartet zu werden.

Aus den dargelegten Gründen besitzt DreamView nur eine Unterstützung für die Darstellung passiver Seitenelemente.

Betrachtet man die Realisierungsmöglichkeiten für die Einbettung synchroner Kooperation in Webbrowser aus Kapitel 1, so ermöglicht nur Variante IV eingeschränkt die Verwendung interaktiver Elemente. Da hier zumindest die jeweiligen Seiten grundsätzlich für alle Teilnehmer gleich aussehen, können Formulare mit dieser Variante bearbeitet werden. Aber auch hier würde die Verwendung von Applets Probleme bereiten, da diese wieder als lokalen Kopien laufen müßten.

## **3 Verwandte Arbeiten**

### **3.1 GroupWeb**

Der verteilte Webbrowser *GroupWeb* [GR96] basiert auf der Groupkit-Plattform [RG96]. Er wurde also wie *DreamView* nicht von Grund auf neu realisiert, sondern stützt sich auf eine CSCW-Plattform ab.

*GroupWeb* besitzt verteilte Mauszeiger, kooperative Linkverfolgung sowie Ausschnittsverfolgung. Zusätzlich kann über einen kooperativen Texteditor zu jeder Seite ein Kommentar verfaßt werden.

Auch *GroupWeb* benutzt das relaxierte WYSIWIS-Paradigma. Es wird also nicht vorausgesetzt, daß alle Teilnehmer exakt dieselbe Sicht auf eine Seite haben.

Die wesentlichen Unterschiede zu *DreamView* können auf folgende Bereiche aufgeteilt werden:

#### Ausschnittsverfolgung

Die Verfolgung von Ausschnitten wird über sogenannte Multiuser-Scrollbars bewerkstelligt. Neben der Scrollbar für den Benutzer werden für jeden weiteren Teilnehmer Scrollbars dargestellt, die auf den jeweiligen Seitenausschnitt eingestellt sind. Hierdurch kann ein Teilnehmer den Ausschnitt eines anderen Teilnehmers abschätzen. Durch Synchronisationsfunktionen ist es möglich, Ausschnitte gleichzuschalten.

Durch eine ungünstige Abbildungsfunktion der Seitenausschnitte ist es u.U. nicht möglich, einen Ausschnitt am Ende einer Seite zu verfolgen. *GroupWeb* verwendet zur Abbildung die Funktion  $f_{gw}$  mit

$$t_{local} = \begin{cases} \frac{S_{local}}{S_{remote}} \cdot t_{remote} & \text{für } \frac{S_{local}}{S_{remote}} \cdot t_{remote} + h_{local} \leq S_{local} \\ S_{local} - h_{local} & \text{sonst} \end{cases} .$$

Diese Funktion hat den Nachteil, daß u.U. die Bedingung *ii*) (vgl. 2.3.2) verletzt wird.

Scrollbars als einziges Element zur Orientierung eignen sich nur bedingt, um einen Überblick über den Ausschnitt eines anderen Benutzers abzuschätzen. Hierzu wäre die Kenntnis des Aufbaus der gesamten Seite notwendig, was nicht unbedingt zutrifft. Eine Übersichtsfunktion ist da sinnvoller. In [GRG96] werden die Nachteile von Multiuser-Scrollbars detailliert beschrieben. Als Alternative wird ein sogenannter Radar-View angegeben. Dieser stellte das gesamte Dokument im Verhältnis 1:64 dar, wobei die Seitenausschnitte der anderen Teilnehmer als Rechteck angedeutet werden. Dieses Element hätte jedoch den Nachteil, daß bei sehr großen Seiten auch sehr große Radardarstellungen entstehen würden. Solange die Länge einer Webseite aber praktisch unbegrenzt sein kann, ist das Element des *Document Trackers* sinnvoller.

### Verteilte Mauszeiger

Wie in *DreamView* operieren die verteilten Mauszeiger nicht auf kartesischen Koordinaten, sondern bilden die Positionen anhand der Inhalte ab. Es gibt aber keine Möglichkeiten, die Verteilung der Mauszeiger einzuschränken. D.h. jeder Mauszeiger wird an jeden anderen Teilnehmer verteilt. Unglücklicherweise unterscheiden sich die Mauszeiger verschiedener Teilnehmer nicht im Aussehen. Bei mehr als drei Teilnehmern ist die Verwendung der Mauszeiger deshalb verwirrend.

### Kooperative Linkverfolgung

In *GroupWeb* wird bei allen Teilnehmern grundsätzlich dieselbe Webseite angezeigt. Wählt einer der Teilnehmer einen Link an, so wird bei allen Teilnehmern auf diese Seite gewechselt. Es ist für einen Teilnehmer nicht möglich, diesen Wechsel zu verzögern, um beispielsweise noch kurz in der aktuellen Seite zu lesen.

Im Gegensatz dazu bietet *DreamView* die Synchronisation der Linkverfolgung nur auf Verlangen an. Es ist damit für die Teilnehmer möglich, zwischen der vollständigen Synchronisation und der individuellen Linkverfolgung zu wählen.

### Annotationen

In *GroupWeb* kann zu jeder Seite ein Textdokument hinterlegt werden. Dieses Dokument ist allen Teilnehmer zugänglich und kann dazu benutzt werden, Bemerkungen zu der Seite zu hinterlegen. Im Gegensatz zu *DreamView* ist es nicht möglich, Texte mit bestimmten Elementen der Seite zu verknüpfen. Es ist auch nicht möglich, in die Bemerkung wieder Links einzubetten.

## **3.2 WWW Shared Session**

Für die CSCW-Umgebung *Habenero* [NCSA] wurde der kooperativer Webbrowser *WWW Shared Session* [NCSAb] entwickelt. Im Gegensatz zu *GroupWeb* und *DreamView* wurde kein komplett neuer Browser realisiert, sondern der Mosaic-Browser [NCSAc] in die *Habenero*-Umgebung integriert. Dieser Browser bietet die Möglichkeit, sich über das Common Client Interface von außen steuern zu lassen. Für die *Habenero*-Umgebung wurde eine kleine

Anwendung geschrieben, die die Kommandos zu dem Browser absetzt und die Gruppenaktivitäten koordiniert. Im Sinne der Einteilung von Kapitel 1 handelt es sich hierbei also um Variante II.

Als wesentliche Funktion wird die gemeinsame Linkverfolgung angeboten. Gruppenspezifische Elemente fehlen dagegen völlig. So gibt es keine Möglichkeit, gemeinsame Mauszeiger darzustellen. Die Seitenausschnitte können nicht verfolgt oder synchronisiert werden.

Bemerkungen können über ein Merkmal des Mosaic-Browsers verwaltet werden. Zu einer Seite können beliebig viele, namentlich gekennzeichnete textuelle Kommentare eingegeben werden, die selbst wieder Links enthalten dürfen. Beim Lesen einer Seite werden sie unten angehängt.

## **4 Zukünftige Entwicklungen**

Neben dem Verteilungsaspekt haben sich die Annotationen als wesentlicher Vorteil herausgestellt. Zur Zeit werden diese getrennt von der eigentlichen Webseite gespeichert und erst zur Darstellung zusammengebracht.

Zukünftig soll es möglich sein, Annotationen in der Webseite selbst speichern zu können. Die HTML-Syntax bietet beliebige Erweiterungsmöglichkeiten, da Tags neu definiert werden können. Denkbar wäre es also, Annotationen in HTML mit abzuspeichern. Diese wären für andere Browser unsichtbar, in *DreamView* könnten sie aber wieder eingelesen und dargestellt werden.

Mit einer solchen Exportschnittstelle ergäben sich neue Möglichkeiten z.B. für die Fernlehre. Übungsaufgaben könnten als HTML-Seite eingeschickt und mit Hilfe von Annotationen korrigiert zurückgeschickt werden.

Hierfür wären allerdings rein textuelle Annotationen nicht aussagekräftig genug. In Zukunft sollten auch graphische Elemente (z.B. Freihandzeichnungen) als Annotation möglich sein.

## **5 Zusammenfassung**

Mit *DreamView* wurde ein verteilter Webbrowser vorgestellt, der sowohl in einer Ein-Benutzer-Umgebung als auch in einer synchronen CSCW-Umgebung eingesetzt werden kann, wobei der Schwerpunkt sicher auf letzterem liegt. Als Merkmale besitzt er verteilte Mauszeiger und Dialogelemente zur Verfolgung von Seitenausschnitten sowie zur kooperativen Linkverfolgung. Als wesentliche Eigenschaft ist es möglich, an Webseiten Annotationen anzubringen und diese graphisch zu verankern.

Als Gruppenwerkzeug kann *DreamView* für viele denkbare Szenarien eingesetzt werden. So kann beispielsweise anhand von HTML-Dokumenten ein Vortrag durchgeführt werden, bei dem die Zuschauer als passive Teilnehmer dem Vortragenden folgen. *DreamView* kann aber auch als Hilfsmittel verwendet werden, welches im wesentlichen bei Gruppensitzungen im Hintergrund bleibt und nur gelegentlich benutzt wird.

*DreamView* wurde mit Hilfe der *DreamTeam*-Plattform realisiert. Der Hauptaufwand bei der Realisierung entstand durch Browser-spezifische Funktionen wie das HTML-Parsing und die graphischen Aufbereitung einer Seite. Die gruppenspezifischen Funktionen konnten mit verhältnismäßig geringem Aufwand realisiert werden, da die wesentlichen Bereiche schon durch die Plattform abgedeckt wurden.

# Literatur

- [FLF94] Frivold T. J., Lang R. E., Fong M. W.,  
*Extending WWW for synchronous collaboration*,  
Proceedings of the Second International World Wide Web Conference '94, Chicago,  
Illinois, 1994
- [GR96] Greenberg S., Roseman M.,  
*GroupWeb: A WWW Browser as Real Time Groupware*,  
Proceedings of the ACM Conference on Human Factors in Computing Systems, ACM Press,  
Apr. 1996, 271-272 [GRG96] Gutwin C., Roseman M., Greenberg S.,  
*A Usability Study of Awareness Widgets in a Shared Workspace Groupware System*,  
Proceedings of the ACM Conference on Computer Supported Cooperative Work, ACM Press,  
Nov. 1996, 258-267
- [NCSA] *NCSA Habanero Homepage*  
<http://www.ncsa.uiuc.edu/SDG/Software/Habanero/HabaneroHome.html>
- [NCSAb] *WWW Shared Session*  
<http://www.ncsa.uiuc.edu/SDG/Software/Habanero/Tools/WWWShare/index.html>
- [NCSAc] *NCSA Mosaic Homepage*  
<http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/>
- [RG96] Roseman M., Greenberg S.  
*Building Real-Time Groupware with GroupKit, A Groupware Toolkit*,  
ACM Transactions on Computer-Human Interaction, Vol. 3, No. 1, Mar. 1996, 66-106
- [Roth97] Roth J.,  
*„DreamTeam“ - eine synchrone CSCW-Plattform für heterogene Umgebungen*,  
Informatik Bericht 218, Fernuniversität Hagen, Mai 1997
- [Roth97b] Roth J.,  
*Wie schreibt man verteilte Applikationen mit „DreamTeam“*,  
Technische Referenz, Fernuniversität Hagen, Juli 1997
- [Sun] *JavaSoft Home Page*  
<http://java.sun.com>
- [W3] *Hypertext Markup Language*  
<http://www.w3.org/MarkUp/>
- [W3b] *Hypertext Transfer Protocol Overview*  
<http://www.w3.org/Protocols/>
- [WR94] Woo T. K., Rees M. J.,  
*A synchronous collaboration tool for World-Wide Web*,  
Proceedings of the Second International World Wide Web Conference '94, Chicago,  
Illinois,