Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Angelika Schäfer

Ein browserbasiertes kollaboratives Mind Map System

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Angelika Schäfer

Ein browserbasiertes kollaboratives Mind Map System

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Kai von Luck
Zweitgutachter : Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 14. Mai 2009

Angelika Schäfer

Thema der Bachelorarbeit

Ein browserbasiertes kollaboratives Mind Map System

Stichworte

Web 2.0, AJAX, Collaborative Workspace, GWT - Google Web Toolkit, Powerwall, Mindmapping

Kurzzusammenfassung

Der Begriff Web 2.0 hat inzwischen einen sehr hohen Bekanntheitsgrad erreicht. Währenddessen hat sich in den letzten Jahren der Bereich der kollaborativen Zusammenarbeit an gemeinsamen Dokumenten immer weiter entwickelt. Es gibt schon verschiedene Anwendungen, die kollaboratives Bearbeiten eines Dokumentes von mehreren Personen orts- und zeitunabhängig erlauben.

Diese Bachelorarbeit setzt sich zum Ziel, eine browserbasierte Mind Map Anwendung in einen kollaborativen Kontext zu bringen. Mit Hilfe der AJAX-Technologie soll die RIA-Anwendung auf Eignung in einem Collaborative Workspace geprüft werden. Schwerpunkt ist hierbei das kooperierende Arbeiten von mehreren Personen am gleichen Ort zur gleichen Zeit. Die Powerwall wird als allgemeines großflächiges Ausgabemedium verwendet und für die Clients der einzelnen Teilnehmer die Sicht im Lese- bzw. im Schreib-Modus bereitgestellt.

Angelika Schäfer

Title of the paper

A browserbased collaborative Mind Map System

Keywords

web 2.0, AJAX, collaborative workspace, GWT - Google Web Toolkit, powerwall, mindmapping

Abstract

The importance of the term Web 2.0 has increased in the last years. Since then, the domain of collaborative work on shared documents has improved. There are many different applications available, where several people - independent of location and time - can edit one single document in collaboration.

The bachelor thesis objectives are to develop a browser-based mind map application in a collaborative context. It will be analysed, if an ajax based ria application fits in a collaborative workspace. The focus is the collaborative work by several people at the same time and place. The powerwall will be used as large-area output device and the clients will be provided with a read and write mode.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Das Thema Computer Supported Cooperative Work (CSCW) hat in den letzten Jahren immer mehr an Aufmerksamkeit gewonnen. Dieser Bereich ist seit fast drei Jahrzehnten in der Entwicklung. In diesem Kontext tauchen u.a. Namen wie Terry Winograd, Fernando Flores, Irene Greif und Paul Cashman auf.

Heutzutage gibt es unzählige Möglichkeiten, gemeinsames Wissen von mehreren Personen zu vereinen. Wikipedia, Openspace und Social Software sind nur einige Beispiele. Das CSCW kann nach Raum und Zeit klassifiziert werden, d.h. es können Personen ort- und zeitunabhängig an gemeinsamen Wissen beitragen. In Collaborative Workspaces kommunizieren alle beteiligten Personen in einem Raum zur gleichen Zeit in einer computerunterstützenden Umgebung miteinander. Die Computer rücken mehr in den Hintergrund - quasi „unsichtbar“, werden nach Terry Winograd und Fernando Flores als Unterstützung der zwischenmenschlichen Kommunikation eingesetzt.

Die HCI Gruppe in Stanford oder die IPSI Gruppe in Darmstadt haben im Bereich des CSCW einige Forschungen betrieben. Projekte in diesem Zusammenhang sind iRoom und Roomware.

Für eine Software eines Collaborative Workspace gibt es nach Köckritz (2006) einige interessante Kriterien in Bezug auf das gemeinsame Wissen, die hier zitiert werden:

- „Gegenseitiger Wissenstransfer“ erlaubt das Lernen von Anderen und ermöglicht damit auch das Angleichen des Wissensstandes aller beteiligten Personen.
- „Interaktive Korrektur“ verbessert die Qualität des Ergebnisses und unterstützt reflektiertes Arbeiten bei gleichzeitiger bidirektionaler Korrektur.

- „Parallele Aufgabenbearbeitung“ erlaubt kleine einfache atomare Teilaufgaben parallel und damit schneller zu bearbeiten.
- „Gemeinsame Aufgabenbearbeitung“ unterstützt den Wissenstransfer und die Korrektur und ermöglicht ein dialogisches Lösen von Problemen.

Es existieren verschiedene Anwendungen, die Informationen innerhalb eines Dokuments im kollaborativen Zusammenhang verwalten. Ein Beispiel für ortsunabhängige Bearbeitung ist das *Google Docs*. Dies ist eine internetbasierte Anwendung zur Textverarbeitung, Tabellenkalkulation und Präsentation. Hier ist durch einen eingebauten Chat eine Zusammenarbeit in Echtzeit möglich. Die beteiligten Personen können entweder zeitgleich oder zu verschiedenen Zeiten auf das gemeinsame Dokument zugreifen und bearbeiten.

Bei der Variante des raum- und zeitgleichen Bearbeitens gibt es u.a. das sogenannte Mind Map. Dort können Begriffe und deren Beziehungen zueinander grafisch dargestellt werden. Diese Methode kann als unterstützendes Werkzeug einer Gruppendiskussion beim Brainstorming (Ideenfindung) verwendet werden. Dabei können sich einige der oben aufgeführten Kriterien wiederspiegeln. Das Konzept des Mind Maps wurde früher oft auf der Pinnwand oder auf dem Whiteboard angewendet. Inzwischen gibt es einige softwaregestützte Mind Map Lösungen.

1.2 Zielsetzung

Man stelle sich folgendes Szenario vor: eine kleine Gruppe von Personen planen ein gemeinsames Projekt und müssen nun ihr Wissen gemeinschaftlich zusammentragen. Dafür eignet sich das Prinzip des Brainstormings. Verschiedene Wege sind hierfür bekannt. Eine alte Methodik war es, sich Karteikarten zu nehmen, seine Ideen und Gedanken darauf zu schreiben und an eine Pinnwand zu stecken. Haben dann alle ihre Karteikarten an der Pinnwand angebracht, wird darüber diskutiert und evtl. eine Struktur hineingebracht. Wenn notwendig, dann wird für ein Protokoll das Ergebnis, das sich auf der Wand befindet, abgeschrieben, was mit weiterem Aufwand verbunden ist.

In dieser Bachelorarbeit wird eine neuere Variante angesprochen, die das Brainstorming in einer Besprechung unter mehreren Personen einfacher machen soll.

Die Grundidee für diese Arbeit ist eine spezielle Anwendung in einer fester Umgebung und beinhaltet folgendes: ein gemeinsames Treffen von mehreren physikalisch anwesenden Personen in einem Raum - die mithilfe von „Collaborative“ Workspaces ihr gemeinschaftliches

Wissen nach dem Brainstorming-Prinzip (hier mit Hilfe eines Mindmapping-Tools umgesetzt) zusammentragen und das nur mit einem Notebook und einem großen Display oder einem aus mehreren Displays (Multihead-Display) zusammengeschaltenen Desktop, der bzw. die an einem bis mehreren Rechnern angeschlossen ist bzw. sind. Am Notebook soll nichts weiter vorausgesetzt werden, außer einem Web-browser und einem Internetzugang. Änderungen wie z.B. einen Knoten oder Kanten zu erstellen, zu bearbeiten oder zu entfernen sollen dann ermöglicht werden. Just-in-time wird das Ergebnis und jede Änderung an den Displays angezeigt.

Im Rahmen dieser Bachelorarbeit soll untersucht werden, ob man RIA bzw. AJAX-Technologien sinnvoll in Collaborative Workspace einsetzen kann. Dabei wird eine Mind Map Lösung als kollaborative Software verwendet. Diese Anwendung muss die Funktionen haben, dass mehrere Personen parallel arbeiten können, indem sie gleichzeitig von verschiedenen Clients betrachten und nebenbei noch editieren können. Der Baum des Mind Maps wird auf einer großen Leinwand - hier ist es die Powerwall - und auf mindestens einem Notebook dargestellt wird. Auf einem weiteren Notebook wird die Möglichkeit des Editierens bereitgestellt. Auf allen Ausgabegeräten, also der Powerwall und den Notebooks der Gruppenmitglieder, soll die Sicht in sekundenschnelle und gleichzeitig miteinander synchronisiert werden. Das Bearbeiten des gemeinsamen Dokumentes soll von verschiedenen Clients (zu unterschiedlichen Zeiten) möglich sein.

1.3 Gliederung

Das Grundlagenkapitel (Kapitel 2) stellt vorab die Mind Map Anwendung kurz vor. Das Collaborative Workspace und einige Konzepte für eine kollaborative Räumlichkeit werden erläutert. Dazu wird das Thema „Kollaboratives Editieren“ erklärt und Aspekte angesprochen, auf die man achten muss. Es wird AJAX-Technologie und RIA besprochen, sowie die Verbindung zwischen RIA und Collaborative Workspace.

Das Kapitel „Analyse“ (Kapitel 3) beschäftigt sich mit der kollaborativen Umgebung, hauptsächlich der Powerwall. Verschiedene Betrachtungen in Bezug auf die Powerwall werden geschildert. An die Benutzer und das System werden Bedingungen gestellt. Mit Usecases werden die Anforderungen bestimmt und des Weiteren ein für diese Bachelorarbeit geltendes Beispielszenario vorgegeben. Anschließend wird im letzten Teil des Kapitels ein Konfliktproblem angesprochen.

Im Kapitel „Design und Realisierung“ (Kapitel 4) wird - basierend auf das Szenario - ein Design entworfen und ein Prototyp realisiert. Dazu werden verschiedene Systemarchitekturen aufgeführt und in Bezug auf die Mind Map näher beschrieben. Für die Realisierung musste eine Kommunikationsmöglichkeit für das synchronisieren der verschiedenen Clients implementiert werden. Anschließend wurde ein Resümee gezogen.

Das Schlusskapitel (Kapitel 5) fasst die Themen der Bachelorarbeit zusammen und zieht ein Resümee.

2 Theoretische Einführung

Heutzutage sind Wikipedias nicht mehr wegzudenken. Früher gab es Lexikas, in denen man etwas nachschlagen konnte. Entweder in Papierform und teilweise schon online oder auf CD bzw. DVD. Die wenigsten Personen waren an der Verbreitung des Wissens beteiligt. Inzwischen existieren verschiedene Wikis im Web, sodass praktisch jeder, der mit seinem Rechner Internetzugang hat, zur Veröffentlichung von Wissen aktiv mitwirken kann. Unzählige Menschen haben ihr Wissen zusammengetragen und auf diese Weise ein großes „Lexikon“ des gemeinsamen Wissens erschaffen. Die Bezeichnung „kollektiven Intelligenz“ tritt in diesem Zusammenhang an verschiedenen Stellen auf. Zu diesem Kontext gehört auch das Thema „Kollaboratives Bearbeiten“, das ein Bestandteil in dieser Bachelorarbeit ist.

Software, die im Zusammenhang mit Web 2.0 gebracht werden können, ermöglichen computerunterstützte Interaktionen zwischen mehreren Personen untereinander. Weiterhin haben sich P2P-Technologien, in der Clients auch Server und umgekehrt sein können, im Bereich des Daten-sharings durchgesetzt und sind stark verbreitet.
Um das Verständnis der in dieser Bachelorarbeit angesprochenen Themenbereiche zu unterstützen, werden hier einige wichtige Begriffe, wie das Web 2.0, AJAX, RIA und Collaborative Workspaces näher erklärt.

2.1 Mind Map

Begriffe und deren Beziehungen zueinander werden über eine grafische Darstellung aufgezeigt. Das Thema befindet sich als Knoten kreisförmig in der Mitte. Unterkapitel eines Kapitels werden über einen Ast verwiesen, sodass die Informationen baumartig erscheinen. Der Vorteil ist einerseits die Übersichtlichkeit und andererseits die Einprägsamkeit durch die Schlüsselworte.

Das Mind Mapping ist eine vernetzte Struktur von sortierten Begriffen, die aus Brainstorming resultieren kann. Weitere Verwendung findet das Mind Mapping in der Strukturierung von

Sachtexten und bei Planung und Organisation. Ersteres kann Übersichtlichkeit unterstützen. Letzteres kann das Einfügen neuer Ergänzungen auf einfache Weise erlauben.

2.2 Collaborative Workspace

Gemeinsam etwas zu erreichen, fließt immer mehr in verschiedene Bereiche des Lebens ein, ob privat oder beruflich. Teamwork, oder auch gerne Enterprise Collaboration genannt, ist ein wichtiger Produktionsfaktor. Darin waren sich die Redner des ersten Tages der Collaboration Technology Conference in Boston einig. Die Teilnehmer, darunter Microsoft, IBM, Adobe und EMC, Open-Source-Hersteller und Entscheider aus Großunternehmen, diskutierten dort über neue Ansätze bei der Teamarbeit. Produkte, die sich im privaten Bereich (Chat, Blogs) und für soziale Netzwerke (Wikis und Foren) bewährt haben, halten inzwischen Einzug in Unternehmen. Teilweise wird in diesen Softwarelösungen Dokumentenschutz, rollenbasierte Prozesse und Workflows, sowie vollständige Verfolgung aller Vorgänge mitgeliefert.[1].

Nach einem Bericht von Ralf Grötker[2] können Wikis und Webblogs als „Verknüpfung menschlicher Gehirne zu einem funktionierenden Ganzen“ gesehen werden. Die Gesamtheit aller Informationen vernetzt sich dabei. Jedermann soll Zugang zu diesem kollektiven Wissen haben. Und allen soll es möglich sein, an der Schaffung neues Wissens teilzuhaben. Kollektive Intelligenz steht somit nicht nur für das gigantische Projekt, alles mit allem in Verbindung zu bringen, sondern könnte zugleich eine Konkurrenz für den einzelnen Experten darstellen. James Surowiecki hat in seinem Buch „The Wisdom of Crowds“[3] einige Beispiele dazu aufgeführt.

Bei einem Experiment der Soziologin Kate H. Gordon[4] mit mehreren Studenten hat sich Folgendes gezeigt: Eine Gruppe von Menschen schätzt irgendeine Menge, ein Gewicht oder eine andere Zahl. Der Durchschnitt der Tipps und Schätzungen wird berechnet. Immer wird er mindestens so nahe am Ziel liegen, meist sogar näher wie der durchschnittliche Einzelne - selbst dann, wenn die Gruppe aus nicht mehr als zwei Teilnehmern besteht. Wenn es wahrscheinlich ist, dass der Durchschnitt sowohl besser ist als ein beliebiger Einzelner als auch wie die meisten Einzelnen, dann bleibt für den einzelnen Experten nur ein geringer Spielraum übrig, in dem er sich als Sieger behaupten kann. Zusammengenommen hatte die Gruppe der zweihundert Studenten bei dem Experiment eine Trefferquote von 94 Prozent.

[1] http://www.heise.de/newsticker/Wikis-und-Blogs-fuer-Enterprise-2-0–/meldung/74506
[2] http://www.heise.de/tp/r4/artikel/23/23822/1.html
[3] http://www.chira.net/clarify/The_Wisdom_of_Crowds.html
[4] „Group Judgments in the Field of Lifted Weights,“ Journal of Experimental Psychology 7 (1924)

Fast auf den Punkt genau hat der Durchschnitt aus allen Schätzwerten die wahren Werte getroffen. Nur fünf Teilnehmer der Studie erzielten mit ihren eigenen Schätzungen ein besseres Ergebnis. Der Durchschnitt hat in dieser Studie die Einzelnen besiegt!

In einer weiteren Studie von Jack Treynor[5] hat von 56 Studenten nur eine Person eine bessere Schätzung als der Gruppendurchschnitt erzielt.

Ein Forschungsbereich zur Integration gemeinsamen Wissens rankt sich um den Begriff Computer Supported Cooperative Work (CSCW). Über das Internet kommen Menschen und Meinungen über Openspace, Wikipedia, Web 2.0 und Social Software zusammen. Ein wirklicher Kontakt zwischen Menschen fehlt bei den meisten dezentralen Ansätzen und die Kommunikation findet über Computertechnologie statt. In Collaborative Workspaces allerdings halten sich alle beteiligten Personen in einem Raum zu gleichen Zeit auf.

CSCW bezeichnet jegliche Zusammenarbeit von Menschen, die von Computertechnologie unterstützt wird. Da die Interaktion zwischen Menschen im Vordergrund stehen soll, ist es ebenfalls nötig, dass die scheinbare Bedeutung des Computers unsichtbar wird. Wenn Computer nahezu „unsichtbar“ zwischen den Menschen und deren Zusammenarbeit stehen, es sich somit um die Schnittmenge der drei Bereiche handelt, dann sprechen wir von Collaborative Workspace.

Die zwei Dimensionen Raum und Zeit sind sinnvoll, um CSCW - Systeme zu klassifizieren. Allgemeiner unterscheiden wir synchrone und asynchrone Arbeitsabläufe. Bei synchronen Arbeitsabläufen findet die Bearbeitung gleichzeitig statt, während asynchrone Arbeitsabläufe zeitlich versetzt stattfinden. Die Bearbeitung in Wikipedia z.B. ist asynchron und nicht an einen Ort gebunden. Da sich in einem Collaborativen Workspace alle beteiligten Personen zu einer bestimmten Zeit sich in einem bestimmten Raum treffen und Wissen gemeinsam zusammentragen, ist dies ein synchroner Arbeitsablauf. Die Abbildung 2.5 wird im späteren Abschnitt näher erläutert.

Wenn eine Person alles selbst wissen, entwickeln und entscheiden könnte, dann wären die Themen CSCW und CW recht uninteressant. Das verteilte Wissen würde keine große Rolle spielen. Doch wenn man in sich geht und sich die Frage stellt: „Wo kommt das Wissen her und wie fügt es sich zusammen?“. Im folgenden Abschnitt wird beschrieben, woher das Wissen kommt und wie es sich zusammenfügt.

[5] „Market Efficiency and the Bean Jar Experiment,“ Financial Analysts Journal 43 (1987)

Im Bericht von Köckritz (2006) für das Masterprojekt wird geschildert, dass Wissen durch Austausch, Reflexion, Weiterentwicklung und Benutzung erhalten bzw. gewonnen werden kann. Weiterhin wird in seiner Ausarbeitung zwischen „expliziten“ und „impliziten“ Wissen unterschieden. Ersteres, also das „explizite Wissen“, wird als bewusstes Wissen beschrieben, welches über ein definiertes Transportmedium wie Sprache oder Schrift mitgeteilt werden kann. Letzteres, das „implizite Wissen“ oder auch das unbewusste Wissen genannt, z.B. Intuition, ist nicht so einfach vermittelbar, weshalb es auch nicht so leicht mit Informationstechnologie nutzbar gemacht werden kann. Ein Zitat nach Köckritz (2006) lautet:

> „Die Sichtbarkeit des implizierten Wissens entsteht scheinbar aus allem und aus nichts und damit im jeweiligen Augenblick. Werden spontane Assoziationen (Intuition) angewendet, die zunächst fern von bekannten Voraussetzungen liegen, um ein neues, unbekanntes Phänomen zu erklären, spricht man von einer Hypothese. Charles Sanders Peirce führte 1867 die Abduktion als eine weitere Form der logischen Schlussfolgerung neben Induktion[1] und Deduktion[2] ein [Philosoph]. Ein CW unterstützt das Ineinandergreifen verschiedener geistiger Tätigkeiten zur gleichen Zeit und damit die Schlussfolgerung durch Abduktion.“

Als weiteres Unterscheidungskriterium für Wissen wird das „strukturelle“ und das gegenteilige „personengebundene“ Wissen angegeben. Letzteres wird von der Person als Wissen mitgebracht. Strukturelles Wissen[3] hingegen ist etwas, was sich in einer Struktur verbirgt in der wir uns bewegen und z.B. in einer Bibliothek auffindbar ist.

Als weiteres Beispiel für das Zusammenfügen von gemeinsamen Wissen wird Extrem-Programming aufgeführt. Dort ist es nützlich, dass sich zwei Programmierer unterstützen, da sie zu zweit eher Fehler finden und sich in Hinblick auf ihr individuelles Wissen gut ergänzen können.

2.2.1 Beispiele aus der realen Welt

Im letzten Jahrzehnt hat sich die Wahrnehmung von Computern geändert, war man damals noch der Ansicht, dass die künstliche Intelligenz mit Fortschritt der Entwicklung die natürliche verdrängen und ersetzen würde. Diesen Wandel beschreiben Winograd und Flores (1989) und sprechen Überlegungen an, wie Computer sinnvoller eingesetzt werden könnten. Dabei werden mehrere Punkte herauskristallisiert, die eine angemessen geeignete Interaktion

[1] Induktion: Folgern von Speziellem auf das Allgemeine, aus einzelnen Ergebnissen wird eine Regel erstellt.
[2] Deduktion: Ableitung des Besonderen aus dem Allgemeinen, aus einer Regel wird eine Folgerung abgeleitet.
[3] Strukturelles Wissen: Wissen über den Zusammenhang von Konzepten innerhalb einer Domäne.

zwischen Computern und Menschen möglich machen. Im Vordergrund steht, die Möglichkeiten zu nutzen, Stärken von Mensch und Computer und damit die Produktivität zu steigern. Dadurch wird der Fokus vom Computer selbst abgelenkt und rückt die Computertechnologie damit in ein anderes Licht. U.a. wird der Gedanke des „readiness-to-hand“ vorgestellt, das wie folgt beschrieben wird:

Als wichtiges Merkmal bei der Nutzung von Werkzeugen ist, dass es nicht wie ein Fremdkörper wahrgenommen wird. Der Benutzer soll nicht darüber nachdenken, auf welche Weise dieses Werkzeug eingesetzt werden muss (intuitive Bedienung, Benutzbarkeit, Bedienbarkeit, Benutzerfreundlichkeit, Usability). „Ein technisches System ist intuitiv benutzbar, wenn es durch nicht bewusste Anwendung von Vorwissen durch den Benutzer zu effektiver Interaktion führt.“ [6] Beispiel in einem Office Word Dokument: das Diskettensymbol zum Abspeichern eines Dokuments, die Schere zum Ausschneiden von Elementen oder der Pfeil nach links, um die letzte Aktion rückgängig zu machen. Das Werkzeug an sich wird vom Benutzer erst wahrgenommen, wenn es nicht wie gewollt funktioniert, nicht existiert oder kaputt ist; ähnlich ist es bei Software, z.B. wenn sie abstürzt oder eine unverständliche Fehlermeldung ausgibt.

Die Ideen von Winograd und Flores fließen in das Konzept des Collaborative Workspace mit ein.

iRoom

Im Rahmen des interactiven Workspaces wurde das Projekt iRoom [7] an der Stanford University entwickelt. Dieses Projekt ist ein weiteres Beispiel für ein Collaborative Workspace. Detaillierte Beschreibungen zu dem Raum und den Komponenten findet man in Johanson und Fox (2002) beschrieben.

Abbildung 2.1: iRoom (von links nach rechts: Beamerprojektion, iTable, Mural)

[6] Quelle: i-com, Zeitschrift für interaktive und kooperative Medien, Ausgabe 3/2006, S. 38.
[7] Abkürzung für interactive room

In der zweiten Generation des iRoom existiert ein großer Konferenztisch, genannt „iTable“, das ein Display integriert hat. An der langen Seite des Tisches befinden sich drei berührungssensitive Smartboard horizontal angeordnet. An der anderen Wand ist ein hochauflösender Bildschirm angebracht, das als Information Mural bezeichnet wird. Das Bild setzt sich aus zwölf einzelnen Bildern zusammen, die über WiredGL [8] von einem 32 PC Graphik-Cluster gerendert werden. Eine wandmonitierte Projektionsfläche - inzwischen von einem großen Display mit 1,8m Bilddiagonale ersetzt und der jeweilige Projektor befinden sich in dieser Räumlichkeit.

Im Unterschied zu dem in nächsten Abschnitt beschriebenen Roomware von Fraunhofer IPSI wurden hier fast ausschließlich Standardkomponenten verwendet. Ein Hauptziel war, möglichst viel Displayfläche zur Nutzung anzubieten, damit das vollständige Problem einer Diskussion innerhalb einer Gruppe eines Meetings auf einem Blick ersichtlich ist.

Alle Geräte sind durch Wireless LAN miteinander verbunden. Das Ziel ist, unter einer gemeinsamen Benutzerschnittstelle, was als „Overface“ bezeichnet wird, sämtliche Ein- und Ausgabegeräte, sowie Dienste zu integrieren. Damit muss sich der Benutzer nicht mehr mit einer Vielzahl von Systemen auseinandersetzen, sondern braucht „nur mit dem Raum interagieren“.

Im Vergleich zum „i-Land“-Projekt des Roomware wird bei iRoom die Konzentration in die Infrastruktur gesteckt, statt auf maßgeschneiderte Software. Es wird möglichst viel von den üblichen Anwendungen wie Microsoft Office verwendet.

Eine kurze Übersicht zur Infrastruktur, Software, Hardware und Sicherheit ist in der Tabelle 2.1 einsehbar.

Roomware

Ein Projekt des Fraunhofer Institut IPSI in Darmstadt war das Roomware, worin Raumelemente wie z.B. Wände, Türen, Möbel mit Informations- und Kommunikationstechnik ausgestattet bzw. integriert wurden. Die Idee war, die Umwelt zu einer Schnittstelle zwischen Menschen zu machen, die dann miteinander interagieren können. Dabei soll der Computer in den Hintergrund rücken bis er quasi unsichtbar ist. Nur noch die Funktionalität bleibt bestehen, die von überall erreichbar und bedienbar ist.

[8] Erweiterung der OpenGL

Abbildung 2.2: iTable

Abbildung 2.3: SmartBoards

1997-1998 wurde eine erste Version der Roomware-Komponenten im Forschungsbereich „Ambiente-Arbeitswelten der Zukunft“ im Rahmen des „i-Land“-Projektes entwickelt:

- DynaWall: 3 berührungsempfindliche Einzeldisplays zusammen zu einer elektronischen Displaywand, 4,5m breit, 1,1m hoch, 3072x768 Pixelauflösung. Damit sind große Informationsmengen anzeigbar, Informationsobjekte sind zwischen allen 3 Displays verschiebbar und Dialogboxen öffnen sich direkt vor dem jeweiligen Benutzer
- CommChair: Stuhl mit integriertem Computer, kombiniert einen Tischrechner und den Komfort eines Lehnsessels, Armlehnen sind beweglich als Schreibtisch oder Auflage und mit einer TFT-Anzeige ausgestattet, Schnittstelle für eine drahtlose Netzwerkverbindung und eine unabhängige Spg.Versorgung, Informationen können dann auf InteracTable oder DynaWall abgebildet werden
- ConnecTable: Rechner mit berührungsempfindlichem Display (Eingabe stiftbasiert), lässt sich mit einem anderen ConnecTable (für Teamarbeiten) verbinden und fügt Informationen zusammen, Verbindungsmöglichkeiten zum CommChair, DynaWall oder dem InterWall - Vorteil: kein rüberlaufen zur Wand, Funk-Lan und Akkus geben Flexibelität
- InteracTable: Tisch mit in der Tischplatte eingebautem berührungsempfindlichem Display, Auflösung 1024x768, Informationsobjekte besitzen keine feste Definition von oben, unten, links oder rechts

Abbildung 2.4: Übersicht der Roomware-Komponenten (von vorne nach hinten: ConnecTable, InteracTable, CommChair und DynaWall)

Die zweite Generation wurde 1999 auf der Basis der ersten Versionen im Forschungs- und Entwicklungskonsortium FOD („Future Office Dynamics“) in Zusammenarbeit mit GMD-IPSI, Wilkhahn (Büromöbelhersteller) und WIEGE (Designbüro) entwickelt und öffentlich vorgestellt.

Die Rechner benötigen eine eigene Software, die „Beach“ genannt wird. Es werden keine Standard Windows- oder Unix-Anwendungen unterstützt.

Zusammenfassend werden einige Fakten zur Infrastruktur, Soft- und Hardware bei iRoom und Roomware tabellarisch (2.1) gegenübergestellt.

2.2.2 Co-located Collaborative Workspace

Das Co-located Collaborative Workspace (CCW) ist eine spezielle Form des Collaborativen Workspace (CW), das für eine computerunterstützte Arbeitsumgebung für projektorientierte Zusammenarbeit von mehreren Personen steht. Das wichtigste Kriterium eines CCWs ist, dass sich die Gruppenteilnehmer physisch zur gleichen Zeit am gleichen Ort befinden. Mehr ist aus der „Raum-Zeit-Matrix“ im Kapitel „Kollaboratives Editieren“ zu entnehmen. Weitere Erklärungen zum CCW stehen in Napitupulu (2008) zur Verfügung.

Diese Form des CWs ist Ausgangspunkt dieser Bachelorarbeit.

2.3 Kollaboratives Editieren

Beim kollaborativen Schreiben innerhalb einer Personengruppe stehen die Kommunikation und gemeinsame Nutzung von Datenressourcen im Mittelpunkt. Zur Realisierung werden hauptsächlich Groupware-Systeme genutzt. Bei Groupware-Systemen wird die Arbeit von mehreren Personen innerhalb einer Gruppe unterstützt. Dazu kann die Unterstützung der kollaborativen Bearbeitung von gemeinsamen Dokumenten eine wichtige Aufgabe des Groupware-Systems sein. Bereiche aus dem CSCW (Computer Supported Cooperative Work) haben hier einen großen Einfluss.

Es sind verschiedene Kombinationen möglich, wie sich eine kollaborative Arbeit bildet. Dies wird in der Raum-Zeit-Matrix deutlich gemacht. Dort wird nach räumliche und zeitliche Verteilung der teilnehmenden Gruppenmitgliedern unterschieden. Die Personen können sich entweder im selben Raum oder an unterschiedlichen Orten aufhalten. Auch zeitlich können die Personen sowohl synchron oder zeitversetzt (asynchron) miteinander kommunizieren. In Abhängigkeit der Raum- und Zeit-Verteilung unterscheiden sich die Merkmale der Zusammenarbeit. In Abbildung 2.5 werden Beispiele der Merkmale in Matrix-Form dargestellt. Groupware-Systeme, die die Interaktionen zwischen den Gruppenteilnehmern unterstützen, können in diesem Schemata klassifiziert werden.

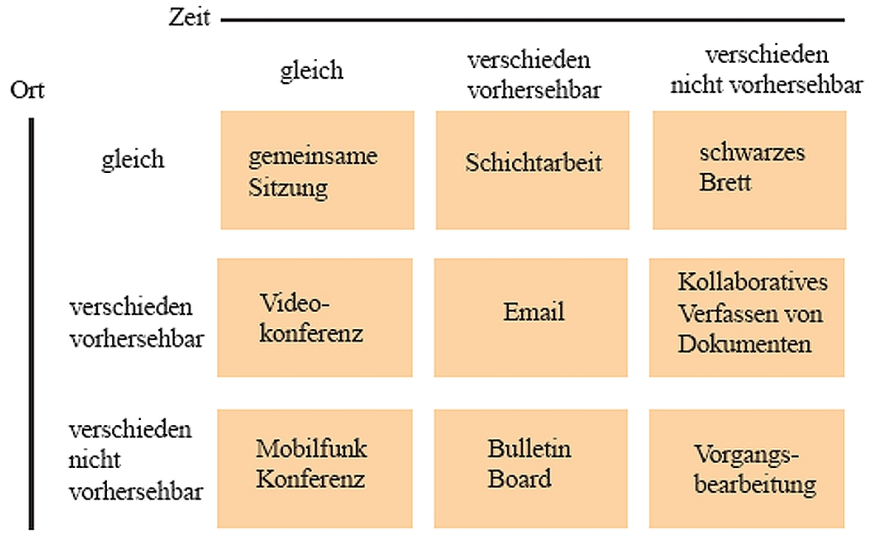

Abbildung 2.5: Raum-Zeit-Matrix [Schlichter]

Eine weitere Möglichkeit ist die Klassifizierung nach Funktionen, die hauptsächlich anwendungsorientiert ist. Es können nach Revout (2008) folgende Bereiche in CSCW herauskristallisiert werden:

- **Nachrichtensysteme:** der Austausch von Nachrichten in Text-Form zwischen den Gruppenmitgliedern wird unterstützt. U.a. Email, Foren-Systeme und Blogs können dazu gezählt werden.
- **Gruppeneditoren:** gemeinsames Verfassen von Dokumenten innerhalb einer Gruppe wird ermöglicht. Gruppeneditoren können in Realzeit-Editoren und in asynchrone Editoren unterteilt werden.
- **Gruppen-Entscheidungsunterstützungssysteme:** unstrukturierte Probleme in einer Gruppe werden erforscht und eine iterative Entscheidungsfindung unterstützt. Meist wird es als elektronischer Sitzungsraum mit mehreren miteinander vernetzten Rechnern und großen, öffentlichen Displays umgesetzt. Audio- und Video-Ausstattung sind auch enthalten.
- **Konferenzsysteme:** nach der Art der angebotenen Leistung gibt es drei Kategorien: Systeme, die Echtzeit-Konferenzen, Telekonferenzen und Desktop-Konferenzen unterstützen.
- **Agentensysteme:** als aktiver Bestandteil einer Gruppenarbeit werden diese intelligenten Agenten z.B. in Computerspielen und zur Überwachung von Sitzungen im Sinne eines Moderators eingesetzt werden.

- **Koordinationssysteme:** Arbeitsergebnisse einzelner Personen werden zur Vollendung eines gemeinsamen Zieles integriert und adaptiert.

Des weiteren kann man sich das 3K-Modell zu Nutze machen [siehe Revout (2008)]. Hier werden die Systeme des CSCW nach ihrer hauptsächlichen Unterstützungsfunktion charakterisiert. Je nach der Art der Zusammenarbeit in einer Gruppen können drei wesentliche Unterstützungsfunktionen unterschieden werden. Die Anfangsbuchstaben dieser drei Unterstützungsfunktionen bilden den Begriff des 3K-Modells [nach Borghoff und Schlichter (1998)].

- **Kommunikationsunterstützung:** Unterstützen der Verständigung von Personen mittels Informationsaustausch.
- **Koordinationsunterstützung:** Unterstützen der Abstimmung aufgabenbezogener Aktivitäten und Ressourcen.
- **Kooperationsunterstützung:** Unterstützen der Verfolgung von gemeinsamen Zielen.

Das 3K-Modell wird in Form eines Dreiecks dargestellt.

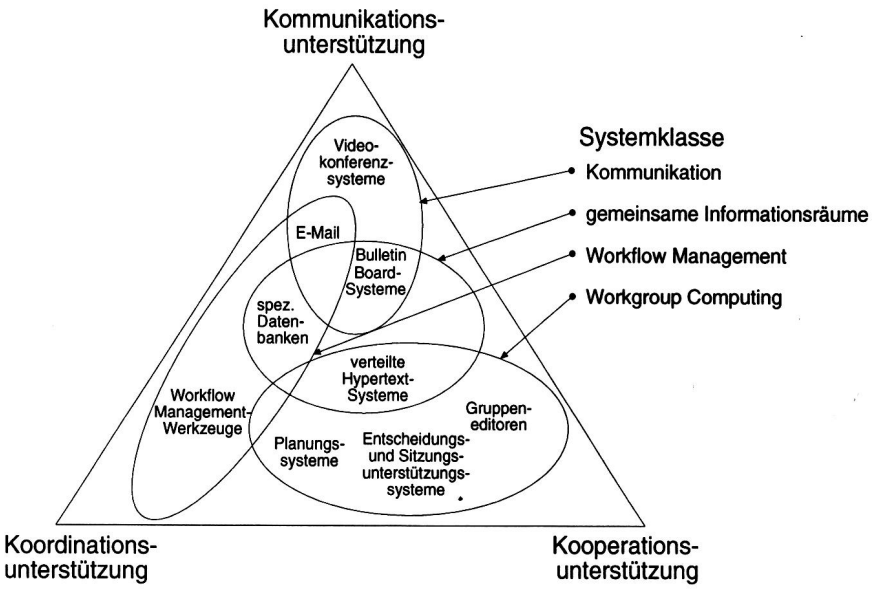

Abbildung 2.6: Klassifikation nach Unterstützungsfunktionen [nach Teufel u. a. (1995)]

In der Abbildung 2.6 wird gezeigt, dass [nach Teufel u. a. (1995)] in Abhängigkeit des Grades der Unterstützung einer jeweiligen Funktion die funktionellen Kategorien von CSCW-Systemen positioniert werden können. Je nach Grad der Ausprägung der jeweiligen Unter-

stützungsfunktion werden CSCW-Systeme nach vier Klassen unterteilt [nach Schlichter (WS 2001/02)]:

- ***Kommunikationssysteme:*** expliziter Informationsaustausch zwischen den Gruppenmitgliedern, wie z.B. über Nachrichten- und Konferenzsysteme.
- ***Gemeinsame Informationsräume:*** impliziter Informationsaustausch in der Gruppe, Verwaltung und Speicherung von gemeinsamen Informationen und Bereitstellung von Zugriffsmechanismen auf diese Daten.
- ***Workflow Management:*** Koordination der Aktivitäten der einzelnen Gruppenmitglieder und den Ressourcen.
- ***Workgroup Computing:*** Unterstützung flexibler Kooperation zwischen den Gruppenmitgliedern, z.B. Erstellung und Bearbeitung von Dokumenten unter kooperativen Bedingungen ermöglichen (entspricht dem Begriff des *Kollaborativen Schreibens*). Ein Beispielsystem sind Gruppeneditoren.

Da es sich in dieser Bachelorarbeit um kollaboratives Bearbeiten eines gemeinsamen Dokuments handelt, in unserem Fall das Mind Map, entspricht es dem letzten aufgeführten Punkt, dem *Workgroup Computing*.

2.3.1 Dokumentenstrukturen

Ein klassisches Dokument, wie z.B. das von MS Word, ist *linear* strukturiert und mit nur einer existierenden Datei nicht für kollaboratives Arbeiten brauchbar. Die Schwierigkeit liegt dabei, dass ein solches Dokument, dass sich in der Bearbeitung befindet, gesperrt werden müsste und andere somit keinen Zugriff hätten. Alternativ wäre ein Versionskontrollmechanismus möglich, das jedes Mal auf Inkonsistenzen oder Abweichungen der verschiedenen Versionen prüft. Beide Verfahren führen dazu, dass das „gemeinsame“ Dokument nur sequentiell von verschiedenen Personen bearbeitet werden kann.

Im Vergleich dazu sind *hierarchische* Dokumentenstrukturen weit besser für kollaboratives Arbeiten geeignet. Das Dokument kann in hierarchischen Strukturen auf mehrere physikalische Segmente aufgeteilt werden, die unabhängig voneinander behandelt werden können. Die einzelnen Teildokumente können konfliktlos synchron bearbeitet werden. [Revout (2008)].

2.3.2 Beispiel für kollaboratives Schreiben: Wiki

Bestes Beispiel für kollaboratives Bearbeiten von gemeinsam nutzbaren Dokumenten stellt das Wiki dar. Eine webbasierte Software, die es beliebigen Personen erlaubt, den Inhalt zu lesen bzw. zu ändern oder Inhalt zu ergänzen. Alle diese Aktivitäten werden über den Browser als Benutzerinterface getätigt. Da fast alle Rechner mit mindestens einem Browser ausgestattet sind, können Personen, die Zugang zu einen Rechner haben, das Wiki benutzen.

Das Wiki ist ein passendes Beispiel für asynchrone Kommunikation zwischen mehreren Personen an verschiedenen Orten. Ein Fallbeispiel mit der Systemannahme, dass bereits eine Wiki-Webanwendung existiert und über den Web-Browser erreichbar und nutzbar ist. Jede Person (A, B und C) befinden sich an verschiedenen Orten und arbeiten chronologisch, aber zeitlich versetzt, am gemeinsamen Dokument zum Thema „X“.

- **Situation 1:** Person A will etwas zu einem Thema „X“ schreiben, das noch keinen Eintrag im Wiki hat.

  Inhalt-Einfügen-Aktivität: Person macht einen Eintrag zum gewählten Thema „X“.

- **Situation 2:** Person B will sich über das Thema „X“ informieren.

  Such-Aktivität: Person B sucht im Wiki einen Eintrag zum Thema „X“.

  Inhalt-Lese-Aktivität: Person B findet den Eintrag zum Thema „X“ und liest es.

- **Situation 3:** Person C hat erst eine Such-Aktivität zum Thema „X“ ausgelöst, den Eintrag zum Thema X in diesem Wiki gelesen (Inhalt-Lese-Aktivität) und möchte weitere Informationen hinzufügen oder den vorhandenen Inhalt ändern.

  Inhalt-Bearbeiten-Aktivität: Person C geht innerhalb des Themas „X“ in den Bearbeiten-Modus und schreibt den neuen Inhalt zum Thema „X“.

Bearbeitungskonflikte können über den Sperr-Mechanismus der gerade bearbeiteten Seite oder durch ein vom Wiki-System integrierten Überwachungs-Mechanismus vermieden werden. Das Wiki ist auf einer Client-Server-Architektur aufgebaut. Trotz allem kann das Wiki nach Angaben von Revout (2008) nicht dem kollaborativen Erstellen von Dokumenten zugeordnet werden, auch wenn gemeinsames, asynchrones Schreiben von Texten möglich ist. Es wird der Grund angegeben, dass Wikis nicht für das Erstellen von klassische Dokumente gedacht sind, sondern als Wissensmanagement in Form eines Informationsspeichers, der öffentlich oder eingeschränkt zugänglich ist.

2.3.3 Kollaborative Bearbeitungssysteme

Hier wird zwischen *synchronem*, *asynchronem* und *hybridem* Schreiben unterschieden:

- ***synchron:*** keine Verzögerungen in der Aktualisierung zulässig (sofortige Sichtbarkeit).
- ***asynchron:*** einzelne Personen editieren zu unterschiedlichen Zeiten, daher ist die sofortige Aktualisierung unwichtig. Zu einem späteren Zeitpunkt werden die Teildokumente zu einem ganzen Dokument zusammengefügt.
- ***hybrid:*** synchrone und asynchrone kollaborative Schreib-Systeme je nach aktuellen Bedingungen, z.B. Verfügbarkeit von anderen Personen (z.B. online oder offline).

Wichtige Eigenschaften von kollaborativen Bearbeitungssystemen:

- **Kooperationsunterstützung:** Funktionalitäten, die gemeinsame Dokumente zur Verfügung stellen und gemeinsame, sowie individuelle Sichten auf die Inhalte ermöglichen.
- **Kollaborative Awareness:** Um ein konsistentes Verständnis der gemeinsamen Aufgabe zu erhalten, muss die Wahrnehmung der anderen Gruppenteilnehmer und ihrer Handlungen gegeben sein.
- **Nebenläufigkeitskontrolle:** Schreibzugriffe, die zeitgleich von mehreren Personen auf ein gemeinsames Dokument ausgeführt werden, müssen kontrolliert werden.
- **Versionierung, Annotation:** Eine Historie der Bearbeitung eines Dokumentes wird aufgebaut, die durch die Versionierung und Änderungsvermerke der einzelnen Gruppenmitglieder erstellt wird.
- **Zugriffskontrolle:** Zugriffsrechte innerhalb einer Arbeitsgruppe bestimmen, welche Mitglieder auf welche Dokumente einen Lese- oder Schreib-Zugriff haben.
- **Benutzerverwaltung:** Dynamische Gruppenarbeit wird durch das Hinzufügen oder Entfernen von Benutzern gewährleistet.
- **Konflikterkennung:** Erkennung der Konflikte bei einem gleichzeitigen Zugriff von mehreren Personen auf ein gemeinsames Dokument bzw. Erkennung der Existenz von mehreren parallelen Versionen eines Dokumentes.

[Revout (2008)]

2.3.4 Strategien bei der Nebenläufigkeitskontrolle

Ein Hauptthema im CSCW ist die Kontrolle von verschiedenen Schreibzugriffen auf ein gemeinsames Dokument. Es werden zwei Verfahren für die Nebenläufigkeitskontrolle unterschieden: das optimistische Verfahren und das pessimistische Verfahren [siehe Revout (2008)].

Das optimistische Verfahren

CSCW-Systeme, die eine geringe Wahrscheinlichkeit an parallelen Zugriffen und einer niedrigen Anforderung an die Konsistenz der gemeinsamen Dokumente haben, können [nach Borghoff und Schlichter (1998)] die optimistische Nebenläufigkeitskontrolle verwenden. Die Vorbeugung von Inkonsistenzen ist hier nicht relevant, sondern die Konfliktentdeckung und Konfliktauflösung sind die zentralen Aufgaben. Den Benutzern ist zu jedem Zeitpunkt möglich, auf ein gemeinsames Dokument zuzugreifen und den Inhalt zu manipulieren. Datenkonsistenzen werden nicht garantiert. Bei eventuell entstehenden Konflikten werden die durchgeführten Modifikationen nicht zurückgesetzt, sondern als eine weitere Version des Dokumentes gespeichert, worüber der Benutzer umgehend informiert wird. Dieser kann dann anschließend seine Änderungen durch das Zusammenführen von zwei auseinander gelaufenen Versionen in das originale Dokument übertragen und eine neue Version des gemeinsamen Dokumentes erstellen. Gewöhnlich ist dies mit zusätzlichem Aufwand verbunden. Daher sind optimistische Verfahren eher nur für Systeme geeignet, die das asynchrone Bearbeiten von Dokumenten unterstützen, dessen Teildokumente von verschiedenen Personen bearbeitet werden und weshalb nur wenige oder keine Konflikte entstehen können.

Das pessimistische Verfahren

Die pessimistische Nebenläufigkeitskontrolle eignet sich in CSCW-Systemen, die hohe Anforderungen an die Konsistenz der gemeinsamen Dokumente stellt. Parallele Schreibzugriffe müssen hier synchronisiert werden, wie bei verteilten Datenbanken. Beim pessimistischen Verfahren wird zwischen zentrale und dezentrale Kontrolle getrennt. Folgend wird ein kurzer Überblick dazu beschrieben.

Zentrale Kontrolle kann in zwei Verfahren aufgesplittet werden.

- **Verfahren mit ausgezeichneter Kontrolleinheit:** es existiert genau ein (ausgezeichneter) Teilnehmer, der alle Zugriffe auf die Dokumente serialisiert und synchronisiert.

- **Token-Verfahren:** ein Token, der ein Ring aus den beteiligten Teilnehmern reihenach durchwandert, ist für ein Dokument verantwortlich. Jeder Teilnehmer erhält den Token für eine vorgegebene Zeit und der aktuelle Besitzer des Tokens hat die gleichen Rechte, wie die ausgezeichnete Einheit im vorangegangenen Verfahren - Serialisieren und Synchronisieren aller Zugriffe auf ein repliziertes Dokument.

Dezentrale Kontrolle führt keine Koordination von allen Zugriffsoperationen durch, da keine zentrale Stelle existiert. Es gibt u.a. folgende Verfahren (ausgewählte Verfahren nach Revout (2008)) mit dezentraler Kontrolle (ohne Votierung):

- **Einfache Sperrverfahren:** ein exklusives Recht für einen Teilnehmer auf die Bearbeitung einer Ressource. Entweder auf ein ganzes Dokument und auf Teile setzbar. Dieses Verfahren ist für synchrones Bearbeiten nicht zu empfehlen, da bei einer zu groben Sperrgranulität (z.B. Sperre auf ganzes Dokument) kein paralleles synchrones Arbeiten möglich ist.
- **Floor-Passing:** ein Verfahren mit wechselnder Kontrolle. Zu einem bestimmten Zeitpunkt besitzt ein einziger Teilnehmer die Kontrolle über das Dokument und somit die Zugriffsberechtigung.
- **Transaktionsverfahren:** ein Ansatz zur Konsistenzerhaltung der Daten und stützend auf Serialisierungs-Mechanismen. Dieses Verfahren wird aufgrund eventueller langer Transaktionsdauer hauptsächlich bei der asynchronen kollaborativen Arbeit genutzt, da eine Verzögerung bei synchroner Dokumentenbearbeitung als störend für den Arbeitsfluß empfunden werden kann. Ist eine Transaktion erfolgreich abgeschlossen worden, werden die Änderungen an den Server übergeben, der wiederum die Änderungen an alle Clients propagiert.
- **Operationale Transformationen:** ein pessimistischer Ansatz zur Nebenläufigkeitskontrolle beim synchronen Bearbeiten. Im Vergleich zu den anderen Verfahren, in denen auf ganze oder Teile eines Dokumentes zugegriffen wird, werden hier nur sehr kleine Einheiten wie Wörter oder Zeichen zugegriffen. Daher ist dieses Verfahren gut für synchrones Bearbeiten geeignet. Kurze Antwortzeiten, die Bewahrung der Kausalität, die Konvergenz der Ergebnisse und Intentionserhaltung zeichnen dieses Verfahren positiv aus. Darüber hinaus wird jede Operation vor der Ausführung nach Bedarf transformiert, um auf die durch andere Operationen ausgelösten Änderungen zu reagieren [Gerlicher, Yang].

Zugriffsrechte

Ein Modell für Zugriffsrechte eines kollaborativen Systems kann sehr komplex sein und damit ist die Implementierung nicht trivial. Einerseits muss es für den Benutzer einfach handzuha-

ben und zu verstehen sein, andererseits möchte man die ersten Prototypen des Systems erst einmal funktionsfähig haben und die Zugriffskontrolle später implemenentieren. Es ist sehr schwierig bis unmöglich eine Sicherheitsarchitektur nachträglich in einem System als Erweiterung einzubauen. Viele Implementierungen von Zugriffsmodellen für kollaborative Umgebungen erreichen keinen höheren Status als das eines Prototyps.

Das Need-to-know Prinzip hat sich als grundlegendes Prinzip [nach Eckert (2006)] bei der Vergabe von Zugriffsrechten etabliert. D. h. jeder Benutzer erhält soviel Rechte, wie für die Erledigung seiner Aufgabe notwendig ist.

Im von Mund (2006) beschriebenen Objektmodell existieren verschiedene Zugriffszonen, wo für unterschiedliche Benutzergruppen verschiedene Rechte gelten. Es wird nach privatem und nach gemeinsamen Arbeitsbereichen unterteilt. Im privatem Bereich agiert der Benutzer in der Rolle des Manager und kein anderer Benutzer hat Zugriff auf diese sich dort befindlichen Daten. Im gemeinsamen Arbeitsbereich können mehrere Benutzer an gemeinsamen Dokumenten arbeiten.

Rollen und Gruppen vereinfachen das Sicherheitsmodell und lässt es effizient umsetzen. Die Kette der Zugriffsrechte sind folgend nur in Aufzählungsform dargestellt.

- Benutzer und Gruppen
- Hierarchien
- Berechtigungsklassen
  - Information (Konvertieren, Kommentieren)
  - Erweiterte Information
  - Bearbeitung (Metadaten ändern)
  - Erweiterte Bearbeitung
  - Endgültig Löschen
  - Kollaboration (Benutzer einladen)
  - Erweiterte Kollaboration
  - Unterschreiben (Signieren des Objektes)
  - Gerätesteuerung
  - Geräteadministration

- Rollen (fachlich, administrativ, Lebenszyklus)
- Vererbung
- Konditionen
- Autorisierung
- Authentifikation
- Betrachtung der Kommunikation
- Protokollierung

[Details zu diesen Punkten siehe Mund (2006)].

In interaktiven Konferenzräumen steht die Koordination von Zugriffen mehrerer Benutzer und Anwendungen auf ein Objekt im Vordergrund. Ein Beispiel ist der Transaktionsmanager, der in Mund (2006) im Kapitel 4.2.6 beschrieben wird. Das Konzept der Zugriffsrechte und Transaktionen sind aus dem Bereich der Datenbanken schon länger bekannt. Eine Transaktion ist eine Folge von Operationen, die als eine logische Einheit betrachtet wird. Transaktionen gelten als abgeschlossen, wenn alle Operationen erfolgreich durchlaufen sind. Wurde eine Ausführung unterbrochen, werden alle vorangegangenen Operationen rückgängig gemacht. Ist dies nicht der Fall, d. h. die früheren Operationen der abgebrochenen Transaktion nicht rückgängig gemacht, führt es zu fehlerhaften Konsequenzen, da fälchlicherweise ungültige Werte gespeichert sind, auf denen nachfolgende Operationen arbeiten.

Eine weitere Möglichkeit, konkurrierende Zugriffe zu minimieren ist über das Gruppenbewusstsein, das durch das Informieren der Teilnehmer über Dokumente, die in Bearbeitung sind, gefördert wird. Dies wird in der Präsentationsschicht realisiert. Es ist eine Absprache der Teilnehmer in einem interaktiven Konferenzraum notwendig. Dies beinhaltet, dass die Benutzer nicht z.B. versuchen, gleichzeitig ein Objekt zu verändern. Die Teilnehmer sollten sich vor oder während der Sitzung auf verschiedene Arbeitsbereiche einigen. In dieser Bachelorarbeit wird von der Nutzung dieses sozialen Protokolls ausgegangen, dass eine soziale Einigung stattfindet, wer über welche Objekte in welchem Zeitraum schreibende Zugriffe erhält.

2.3.5 Architekturen für kollaborative Bearbeitungssysteme

Es gibt zwei Typen von Architekturen: die zentralisierte und die replizierte Systemarchitektur. Die hybride Systemarchitektur ist eine Kombination der zentralisierten und replizierten Architektur und nutzt die Vorteile beider Architekturen. Sie ist sowohl für synchrone, als auch

für aynchrone kollaborative Arbeiten geeignet.

Zentralisierte und replizierte Systemarchitekturen

Gemeinschaftliche Dokumente werden auf einem oder mehreren Servern gespeichert. Die Aufgaben eines solchen Servers sind:

- Verwaltungsaufgaben,
- Authentifizierungsaufgaben,
- Zugriffskontrolle und
- Propagierung der Änderungen an einem gemeinsamen Dokument.

Um ein gemeinsames Dokument zu bearbeiten, greifen die einzelnen Clients auf den Server zu. Der Server besitzt immer eine aktuelle, gültige Version des Dokumentes. Wenn der Client eine Kopie des zu bearbeitenden Dokumentes besitzt. Es wird nach Revout (2008) von Schwabe u. a. (2001) zitiert, als so genanntes „Information Sharing“ bezeichnet. Für die Sequenzialisierung der Operationen und die Synchronisierung der verschiedenen Versionen des Dokumentes ist der Server verantwortlich.

In dieser Architektur sind Datenkonsistenzen leicht zu wahren. Diese Systemarchitektur hat jedoch auch seine Schwächen, wie z.B. lange Antwort- und/oder Benachrichtigungszeiten, die ausgelöst werden, da die Propagierung der Änderungen über den zentralen Server stattfinden. Im schlimmsten Fall kann es durch einen Engpass die Verfügbarkeit der gemeinsamen Dokumente verschlechtern. Dadurch eignet sich diese Architektur mehr für asynchrone Bearbeitungssysteme als für synchrone Kollaboration.

Die Tabelle 2.2 stellt die zentralisierte und die replizierte Systemarchitekturen gegenüber. Dabei wird auch die Datenhaltung und die Zustandsänderung betrachtet.

Hybride Systemarchitektur

Die zentralisierte und die replizierte Systemarchitektur werden im hybriden Konzept miteinander kombiniert. Damit wird versucht, die Vorteile beider Architekturen zu nutzen und ihre Nachteile zu kompensieren.

In der hybrid-Architektur besitzen Server und die einzelnen Clients jeweils eine Kopie des gemeinsamen Dokumentes. Die Clients benachrichtigen andere Clients und den zentralen

Server, wenn Änderungen stattgefunden haben und übermitteln diese. Dadurch wird gewährleistet, dass auf dem Server immer die aktuellste Version des Dokumentes vorliegt. Dieses Dokument kann jederzeit von neuen Teilnehmern als Kopie abgerufen werden und nach Bedarf zur Synchronisation genutzt werden.

Peer-to-Peer-Architektur

Peer-to-Peer (P2P) lässt sich den replizierten Systemarchitekturen zuordnen. In P2P-Systeme existiert kein expliziter Server, sondern dort sind Clients zum Teil gleichzeitig auch Server und umgekehrt.

Peer-to-Peer-Architekturen sind nach folgenden Merkmalen zu erkennen:

- keine zentrale Kontrolle
- keine zentrale Datenbasis. Jeder Peer besitzt einen Teil des gesamten Datenbestandes
- keine globale Sicht auf das System vom einzelnen Peer aus
- Interaktion der einzelnen Peers führen zu einem globalen Verhalten des Systems
- Peers sind autonom
- Peers und Verbindungen untereinander sind nicht zwingenderweise zuverlässig
- trotz verteilter Speicherung, möglicher Verbindungsausfälle etc. müssen die gesamten im System gespeicherten Daten verfügbar sein

Folgende Tabelle der Abstraktionsebenen ist nach der Ebene aufgeteilt und führt eine Beschreibung, die jeweilige Applikationsebene, die Dienstklasse und einige Beispiele auf: Tabelle 2.3:

Aus Sicht der Teilnehmer in einem CSCW wird der P2P-Ansatz auf der Benutzerebene verwendet: Gruppierung von Teilnehmern und Unterstützung der Interaktionen untereinander.

Folgende Hierarchie-Grade sind weitere Merkmale einer P2P-Architektur:

- **Zentralisiertes Modell:** eine zentrale Stelle, die den globalen Index aller Peers verwaltet und die Peers koordiniert.
- **Dezentralisiertes Modell:** kein globaler Index existent. Peers interagieren untereinander über den direkten Weg.

- **Hierarchisches Modell:** beide vorangegangenen Modelle kombiniert. Normale Peers und hierarchische Super-Peers, die den globalen Index verwalten.

Ein Hauptvorteil bei P2P-Systemen ist, dass sich die Aufgaben auf die einzelnen Knoten verteilen. Durch die Aufgabenverteilung stehen viel größere Rechenleistung, Speicherkapazität und Bandbreite zur Verfügung. Dank dieser Eigenschaften nehmen die P2P-Systeme an Leistung mit steigender Teilnehmerzahl zu, während in der Client/Server-Architektur bei höherer Teilnehmeranzahl die Leistung abnimmt.

Eine Grundannahme eines P2P-Systems ist, dass jeder Knoten jederzeit ausfallen könnte. Das System ist dadurch vorbereitet, entsprechende Maßnahmen beim Auftreten einer solchen Situationen vorzunehmen. Die Übernahme der Verwaltungsaufgaben (z.B. um das Netz aufrecht zu erhalten) durch die einzelnen Teilnehmer unterstützt bei einem Ausfall die Möglichkeit, den ausfallenden Knoten zu ersetzen. Dank dessen ist ein P2P-System robuster als eine Client/Server-Architektur [siehe Wendt (2007)].

2.4 Web 2.0 und Rich Internet Applications

Am 30.09.2005 veröffentlichte Tim O'Reilly vom gleichnamigen Verlag seinen richtungsweisenden Artikel „What is Web 2.0“[9]. Es gab eine technische Revolution, das Zerplatzen der Dot-Com-Blase im Herbst 2001 stellte einen Wendepunkt für das World Wide Web dar.

Das Konzept „Web 2.0“ begann mit einem Brainstorming zwischen O'Reilly und MediaLive International, wie auf 2.7 abgebildet wurde.

Im anfänglichen Brainstorming wurden folgende Beispiele für die Bedeutung von Web 2.0 formuliert:

[9] Quelle: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

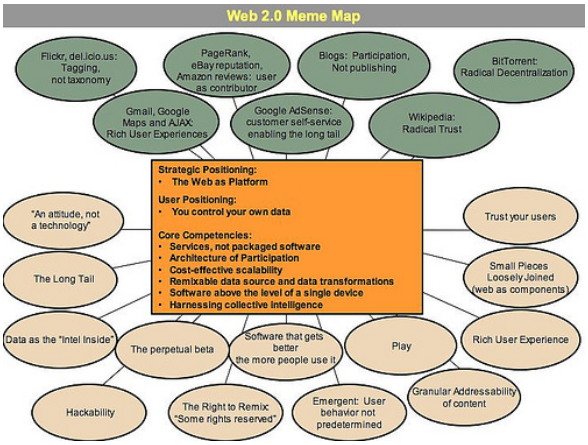

Abbildung 2.7: Das Resultat Web 2.0 nach dem Brainstorming

Web 1.0		**Web 2.0**
DoubleClick	–>	Google AdSense
Ofoto	–>	Flickr
Akamai	–>	BitTorrent
mp3.com	–>	Napster
Britannica Online	–>	Wikipedia
Persönliche Webseiten	–>	Blogs
Spekulation mit Domain Namen	–>	Suchmaschinen-Optimierung
Seitenaufrufe	–>	„cost per click“
Extraktion mittels Screen Scraping	–>	Web Services
Veröffentlichung	–>	Beteiligung
Content Management Systeme	–>	Wikis
Taxonomie (Verzeichnisse)	–>	„Folksonomy“ (Tagging)
Feststehend („stickiness“)	–>	Zusammenwachsen („syndication“)

Nach welchen Kriterien die einzelnen Anwendungen und Ansätze als „Web 1.0“ oder „Web 2.0“ klassifiziert wurden, kann auf der Webseite von Oreilly[10] verfolgt werden.

Das Fazit für Web 2.0: Kunden-Selbstbedienung und algorithmisches Datenmanagement zu Nutze machen, um jeden Winkel des Web zu erreichen, nicht nur die schmale Spitze, sondern auch die breite Masse („The Long Tail“).

BitTorrent und andere Pioniere des P2P-Bereichs wählen den radikalen Weg zur Dezentralisierung des Internet. Jeder Client ist auch Server und Dateien werden in Fragmente

[10] http://www.oreilly.de/artikel/web20.html

aufgeteilt, die von verschiedenen Orten bezogen werden können, sodass das Netzwerk der Downloader auf transparente Weise anderen Nutzern Bandbreite und Daten zur Verfügung stellt. Je beliebter eine Datei ist, desto besser kann sie angeboten werden, da dementsprechend viele User Bandbreite und Fragmente für diese Datei bereitstellen.

BitTorrent zeigt somit als Schlüsselprinzip für Web 2.0: Ein Dienst wird umso besser, je mehr Leute ihn nutzen. Während Akamai zur Verbesserung des Service mehr Hardware zur Verfügung stellen muss, steuert jeder BitTorrent-User automatische seine Ressourcen zum Netzwerk bei. Es existiert also eine implizite Ärchitektur der Partizipation", oder auch eine ëingebaute Ethik der Kooperation", in der der Dienst selbst als intelligenter Vermittler auftritt, der die Enden des Netzwerks verbindet, die Ressourcen der User bündelt und ihnen wieder zur Verfügung stellt.

Webbasierte Anwendungen, die überwiegend nur abhängig davon sind, einen Internetzugang zu haben, eignen sich sehr gut für den Thin-Client-Ansatz. Eine nähere Beschreibung dazu wird noch in den nächsten Unterkapiteln folgen. Bis vor einiger Zeit war es ein Problem mit der Datenversorgung bei Webanwendungen, da an den Server ganze Seitenaufforderungen geschickt wurden. Die Lösung darauf wird XMLHttpRequest genannt, das die Grundlage von der AJAX-Technologie ist, d. h. Datenversorgung ohne Seitenladen. Im nächsten Kapitel wird detaillierter auf diese Technologie eingegangen.

2.4.1 AJAX

Die ursprüngliche Idee des Webbrowsers zur Darstellung von Inhalten wurde bereits soweit weiterentwickelt, wie sie vernünftigerweise leisten können. *Asynchronous JavaScript and XML* (AJAX) wurde als relativ neuer Begriff von Jesse James Garrett von Adaptive Path (Februar 2005) im Artikel „A New Approach to Web Application“[11] geprägt. AJAX ist jedoch keine neue Technologie - genau genommen kann es noch nicht einmal als Technologie beschrieben werden - sondern verbindet schon bestehende Techniken miteinander. Die schon länger bekannte Dynamic HTML und Remote Scripting sind nun einige Bestandteile von AJAX. Somit ist AJAX eine Bezeichnung für schon vorhandene Begriffe und Technologien, die nun zu einem Marketing-Wort zusammengefasst wurden. Dieser Artikel zog einen Boom nach sich und löste den bis dahin verwendeten Ausdruck „Web 2.0“ für die folgende Generation der Webentwicklung ab.

Aber selbst die Abkürzung AJAX ist merkwürdig, denn das zweite *A* steht für das englische *And*. Sogar aus fachlicher Sicht lässt sich über die anderen Buchstaben diskutieren. Es

[11] Nachzulesen unter http://www.adaptivepath.com/publications/essays/archives/000385.php

muss nicht notwendigerweise *asynchron* sein, *XML* wird nur selten eingesetzt, nur das *J* ist passend für *JavaScript*. Etwas später habe auch Garrett es eingesehen und verwendet die gemischte Schreibweise Ajax/AJAX, mit dem zusätzlichen Hinweis, dass es kein Akronym mehr sei, sondern einfach ein Begriff.

Einige Beispiele für Ajax-Anwendungen sind:

- das Google Suggest[12],
- eine Bildbearbeitung[13],
- einige Tabellenkalkulationen[14] und
- eine Textverarbeitung[15].
- Google Maps[16].
- Google Docs[17] (Google Write, Tables) für Dokumente und Tabellen.
- Nachbildungen von Office Anwendungen, wie z.B. von OpenOffice oder MS Office, werden auch von „Ajax13“ (ajaxWrite, ajaxSketch, ajaxXLS, ajaxPresents) [18] zur Verfügung gestellt, ähnlich wie beim Google Docs.
- „Flickr“[19], das digitale Fotoalbum zum veröffentlichen eigener Fotos.
- „ObjectGraph Dictionary“[20] ist ein Wörterbuch der besonderen Art. Dort können dynamisch unterschiedliche Ergebnisarten einer Suche gewählt werden, u. a. als Thesaurus, als Wörterbuch oder in detaillierter dokumentarform.
- Mit „Zimbra“[21] hat man eine desktopähnliche, mächtige Anwendung mit vielen Funktionen erschaffen, mit dem man Emails verwalten kann.

Mit Hilfe von Ajax wird ermöglicht, dass eine Datenübertragung zwischen Client und Server im Hintergrund ablaufen kann, ohne dass es für den Anwender sichtbar wird. Die jeweilige Seite wird dabei nicht neu geladen, lediglich die entsprechenden Inhalte mit den angeforderten Informationen aktualisiert. Die Abbildung 2.8 von Jesse James Garrett stellt dieses

[12] http://www.google.com/webhp?complete=1&hl=en
[13] http://demo.neximage.com/
[14] http://spreadsheets.google.com/, http://www.numsum.com/, http://ajaxxls.com/
[15] http://www.writely.com/
[16] http://maps.google.com
[17] http://docs.google.com
[18] http://www.ajax13.com
[19] http://flickr.com/
[20] http://www.objectgraph.com/dictionary/
[21] http://www.zimbra.com/

Konzept dar.

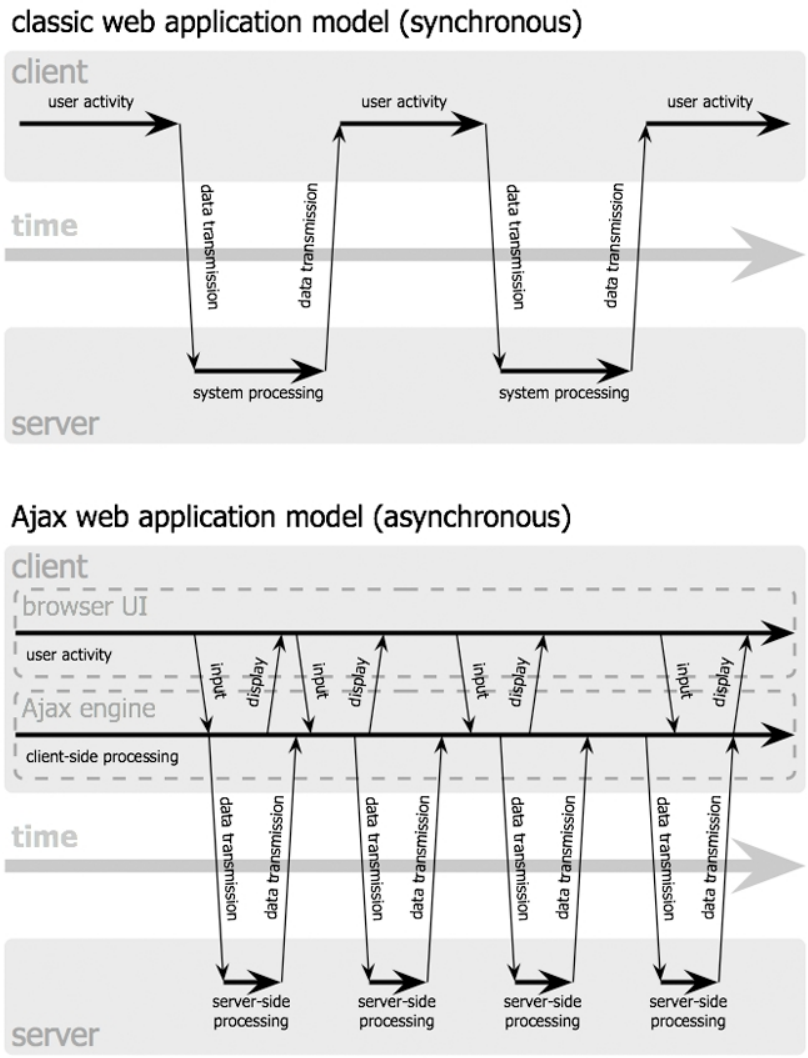

Abbildung 2.8: synchrone und asynchrone Kommunikation [Garrett (2005)]

Im wesentlichen stützt sich Ajax auf folgenden Techniken:

- *XHTML* und *CSS* ist für die Formatierung der Webseite zuständig
- *DOM (Document Object Model)* sorgt für den dynamischen Zugriff auf dem Dokumentenbaum
- *XML* oder *JSON*, das für den Datenaustausch und *XSLT*, das für die Transformation verantwortlich ist

- *XMLHttpRequest*-Objekt, kümmert sich um die Client/Server-Kommunikation auf asynchroner Basis
- *JavaScript* dient als Schnittstelle für alle diese Komponenten

[Wenz (2007), Gamperl (2006)]

Der Ursprung von Ajax begann in der Outlook-Abteilung von Microsoft. Für die Web-Schnittstelle von Outlook (OWA - Outlook Web Access) wurde die Funktionalität der HTTP-Anfragen im Hintergrund gebraucht, um beispielsweise regelmässig nach neu eingetroffene Emails zu prüfen, ohne jedoch permanent neu laden zu müssen. Das Team für den Internet-Explorer integrierte in den Browser die Technologie, im Hintergrund die HTTP-Anfragen abzusetzen und die Rückgabe auszuwerten. Das war die Lösung für die Anforderung des Outlook-Team. Dieses Feature wurde als ein ActiveX-Control namens XMLHttpRequest implementiert und wurde zu einem späteren Zeitpunkt von anderen Browser-Herstellern als natives Browserobjekt eingebaut. Und somit gibt es auch in alternativen Browsern die Ajax-Unterstützung.

Das XMLHttpRequest schickt also die HTTP-Anfrage an einen Webserver und wertet die Rückgabe aus. Die Kunst besteht darin, die Server-Rückgabe auf der Seite darzustellen, wofür dann JavaScript - DOM, manchmal auch DHTML - verwendet wird. [Wenz (2007)]

Der Ablauf findet folgendermaßen statt: der asynchrone Prozess wird in einem eigenen Thread angelegt, der im Hintergrund läuft. Der Benutzer kann währenddessen andere Sachen machen. Der Threadstart selbst blockiert für einen kurzen Moment den Benutzer (für die erste Rückmeldung vom Server), danach merkt der Benutzer nichts von den Hintergrundaktivitäten.

Da HTTP unidirektional arbeitet, kann die Kontaktaufnahme nur vom Client zum Server stattfinden, und nicht andersherum. Daher muss man den Server um eine Ergänzung erweitern: zum HTTP wird der serverseitige Status (das wäre die asynchrone Callback-Lösung) hinzu genommen. Beim asynchronen Callback wird der Client zweimal benachrichtigt, das erste Mal, wenn der Thread angelegt wurde, das zweite Mal, wenn der Thread abgeschlossen ist.

Einige Grundprinzipien von Ajax im Vergleich zum klassischen, seitenbasierten Anwendungsmodell:

- ein Teil der Anwendungslogik wird in den Browser verschoben (Lebenszyklus einer Sitzung). Das Dokument bleibt während der ganzen Sitzung beim Benutzer, kann das Aussehen während Interaktion oft verändern. Das Dokument kann Daten speichern,

z.B. wird der Inhalt eines Warenkorbs im Online-Shop statt in der Serversitzung des klassischen Modells im Browser gespeichert.

- nur ein Teil eines Dokumentes wird aktualisiert, statt die gesamte Seite. Im Beispiel des Warenkorbs wird in Ajax eine asynchrone Anfrage an den Server gesendet, der wiederum nur die erforderlichen Daten an den Client zurückschickt und nicht mehr die gesamte Seite. Realisiert wird es z.B. durch JavaScript-Code, Stream, einfacher Text oder ein kleines XML-Dokument als Rückgabe an den Client.
- klassisch konnte man mit dem Server nur über Hyperlinks oder das Senden von Formularen in Kontakt treten. Mit Ajax funktioniert das über Mausbewegung oder Tastendruck, wie bei Google Suggest.
- hochleistungsfähiger, wartungsfreundlicher Code, der umfangreicher ist als bei der klassischen Webanwendung. Ordentliche Strukturierung des Codes ist dabei ein Muss. Ajax-Anwendung besteht aus komplexen, funktionalen Code, der effizient mit dem Server kommunizieren muss, während der Benutzer weiterarbeitet.

[Crane u. a. (2006)]

Mit AJAX erhofft man sich, dass sich das Einsatzgebiet von Browser-Anwendungen etwas mehr im Rich-Client-Anwendungsgebiet verbreitet, da mit der AJAX-Technologie einige Funktionalitäten möglich sind, die wir aus den Rich-Client-Applikationen kennen.

2.4.2 Code on Demand

Code on Demand ist eine Technologie, in dem der Client nach seiner Anfrage vom Server ausführbarer Code übermittelt bekommt. Danach wird der Code in diesem Fall auf dem Client ausgeführt.

Es wird auch als Fähigkeit beschrieben, zur Laufzeit dynamisch die Verbindung zwischen Software-Komponente der jeweiligen Applikation und deren physikalische Lokation in einem Computer-Netzwerk zu rekonfigurieren.

In Kabalkin (2008) wird Carzaniga zitiert:

> „The capability to reconfigure dynamically, at run-time, the binding between the software components of the application and their physical location within a computer network.“

2.4.3 Rich Internet Applications

Die Begriffe der klassischen Web-Anwendung und der Desktop-Anwendungen sind inzwischen im Allgemeinen bekannt. Bei der Web-Anwendung muss der Benutzer keine aufwendige Installationen durchführen, es reicht ein Webbrowser, in manchen Fällen einige zusätzliche Browser-Plug-Ins, aus. Eine Beispielanwendung als unterstützende Software für Gruppenarbeit ist der Email-Client. In so einer Anwendung ist nur wenig Logik im Einsatz, als Beispiel werden mit Hilfe von Javascript Formulareingaben ausgewertet und geprüft. Dennoch sind die klassischen Web-Anwendungen den Desktop-Anwendungen gegenüber im Nachteil, z.B. im Funktionsumfang (Drag and Drop, Verschieben von Fenstern und Dialogen). [Hartmann (2007)].

RIA (Rich Internet Applications) überbrückt das, was bei der klassische Web-Anwendung an Funktionen der Desktop-Anwendung fehlt und vereint von beiden Anwendungstypen die Vorteile. Die RIA wird als Erweiterung an die Web-Anwendung gesehen, die sich an Funktionalitäten der Desktop-Anwendungen anlehnt. Dadurch wird eine Umgewöhnung des Benutzers vereinfacht.

Das, was die RIA im Vergleich zur klassischen Web-Anwendung ausmacht ist, dass je nach Entscheidung des Entwicklers sich der Schwerpunkt der Logik auf die Client- oder Serverseite verteilt. Im Abschnitt „Kompetenzverteilung auf Client und Server“ des Kapitel 4.1.1 wird auf die folgend dargestellte Abbildung näher eingegangen.

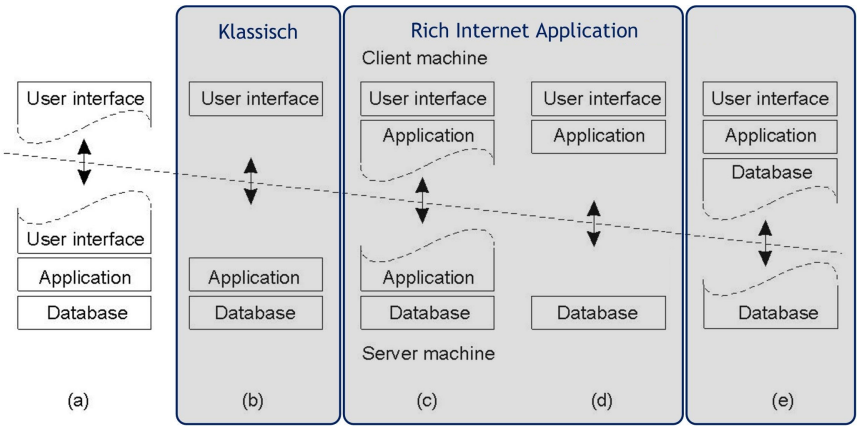

Abbildung 2.9: Verteilung auf Client und Server in Anlehnung an Tanenbaum und van Steen (2003)

Die Clientanfrage wird an den Server gestellt und erst nach einer unbestimmten Zeit, also asynchron, wird die Antwort darauf zurückgesendet. Der Benutzer kann in dieser Zeit weiter mit der Anwendung arbeiten. Da nur Teilinformationen und nicht die gesamte Seite gesendet werden, verringert es die Netzwerklast.

Die RIA-Technologie unterscheidet sich in rein Browser-basierte Anwendungen, in dem der Client ausschließlich einen Browser verwendet und in Plugin-basierte oder Stand-Alone-Anwendungen, wo zusätzliche Laufzeitumgebungen auf Clientseite verwendet werden. In der ersten Variante laufen AJAX-Anwendungen ohne weitere Browser-Plugins. Letztere nutzt Flash oder Java.

Teile eines Anwendungscode (GUI und teilweise Anwendungslogik) werden bei einer RIA vom Server zum Client übertragen. Dadurch werden Zugriffe auf den Server eingespart, die die Bedienung der RIA flüssiger werden lässt. Um eine RIA zu entwickeln wird folgendes verwendet:

- AJAX (Asynchronous JavaScript and XML)
- AJAX-Frameworks (z.B.: Prototype, Dojo, ASP.NET AJAX, Google Web Toolkit)
- Alternativen, wie z.B.: Adobe Flex 2, Microsoft Silverlight, Sun JavaFX, Google Gears

[Haase u. a. (2008)]

Vorteile der RIA:

- flüssigere Bedienung im Vergleich zur klassischen Web-Anwendung durch Laden der notwendigen Datenteile
- mehr Daten und Logik auf Clientseite, d.h. weniger Serverzugriffe
- bekanntes User-Interface, da Anlehnung an Desktop-Anwendungen, daher kaum große Umgewöhnung
- geringer Installations- und Administrationsaufwand, da das Updaten der Software wesentlich einfacher als bei normalen Desktop-Anwendungen ist
- einfacher Zugriff auf RIA - wenn Zugang zum Server existiert - da die Laufzeitumgebung, vorallem der Browser auf den meisten Clients bereits installiert ist
- kleine Servernachrichten (nötige Datenteile) über asynchrone Kommunikation, also der Client wartet nicht auf die Serverantwort
- Server-Push wird von vielen RIA-Technologien angeboten; die Alternative des periodischen Pollens erzeugt viel Overhead, stattdessen ist „deferred reply“ („verzögerte Antwort“) sinnvoller, jedoch ist bei großer Anzahl an Clients die Skalierbarkeit schlecht.

2.4.4 Thick Client und Thin Client im Vergleich

Ein Server ist ein Programm, das einen Dienst (Service) anbietet und passiv auf Anforderungen wartet. Ein Client kann dann diesen Dienst nutzen und aktiv anfordern. Das Protokoll für die Kommunikation ist abhängig vom Dienst.

Dem Client werden die Daten möglichst vollständig vom Server übermittelt und seine Funktion auf Ein- und Ausgabe beschränkt.

Ziel ist eine Thin Client Anwendung, d.h. dass eine Software, die auch z.B. in einem Unternehmen eingesetzt werden könnte, nicht mehr auf jedem Client installiert werden muss, sondern über einem Browser nutzbar ist. Somit wird schnell sichergestellt, dass bei jedem Nutzer Software-Updates sofort integriert sind, ohne zusätzliche teilweise auch aufwändige Nachinstallationen. Die Anwendungslogik läuft auf dem Server. Die Browser-Anwendungen eignen sich bei Abrufen von Daten. Im Gegensatz zum Thin Client ist der Installationsaufwand der Desktopanwendungen auf Thick Clients hoch.

Ein weiterer Vorteil eines Thin Clients ist, dass es hardware-, sprach- und betriebssystemunabhängig ist, da ja alles über den Webbrowser funktioniert. Für AJAX- bzw. JavaScript-Anwendungen ist es nicht nötig, weitere Plug-Ins installieren zu müssen, da die gängigen Browser Javascript „verstehen“.

2.5 Notwendigkeit von RIA in Collaborative Workspace

Die Kommunikation zwischen Menschen in einer Gruppe steht mehr und mehr im Vordergrund, während Computer im Collaborative Workspace in den Hintergrund treten. Dabei unterstützt die RIA-Technologie ungemein.

Betrachtet man die Aspekte Thick Client und Thin Client, kommt man zu dem Gesichtspunkt, wie vorteilhaft Thin-Clients in einer Gruppe von unterschiedlichen Gruppenteilnehmern in einer kollaborativen Zusammenarbeit sein kann. Die Teilnehmer könnten von verschiedenen Standorten, mit verschiedener Hardware zu dieser Zusammenkunft eintreffen. Durch die sehr wahrscheinliche unterschiedliche Infrastruktur ist ein Thick-Client ziemlich ungeeignet, da die verwendete Software auf jedem einzelnen Client teils aufwändig installiert werden muss und möglicherweise administrative Schwierigkeiten auftreten können. Bei Thin-Clients umgeht man diese Probleme, da inzwischen auf den meisten Geräten ein Web-Browser

installiert ist.

Als weitere Gesichtspunkte kann man das Web 1.0, also die klassische Webanwendung und das Web 2.0, die neue Webanwendung, die sich an Desktop-Anwendungen anlehnt betrachten. Letzteres wirkt durch die AJAX-Elementen dynamisch, flexibel und schnell, während die klassischen Web-Anwendungen langsam im Übertragen und Aktualisieren von Daten sind (da nur vollständige Seiten synchron kommuniziert werden) und dadurch benutzerunfreundlicher erscheinen.

Kategorie	iRoom	Roomware
Infrastruktur	Miteinander vernetzte Ein- und Ausgabegeräte im selben Raum mit Zugriff auf den gleichen Datenbestand	
Software	Eigenes Betriebssystem **„iROS“** [a]: - Unterstützung der gemeinsamen Benutzung von Displays, - orts- und typenunabhängige Datenspeicherung und - Koordination von Anwendungen. EventHeap: - Koordination zwischen Anwendungen. - Erweiterung von Tuplespaces[b]. - Kommunikations- und Koordinationsebene. iCrafter: Instanziierung einer Benutzerschnittstelle zur verteilten Anwendung für beliebige Eingabemedien. DataHeap: - Typenunabhängigkeit durch Transformationsframework für Datenspeicherung. - Daten durch Indizes auf typenunabhängige Metadaten zugreifbar.	Framework **„Beach“**: - Displaysteuerung, - Objektdarstellung und deren jeweilige Transformation. Framework **„COAST“**: synchrone Nutzung der Datenobjekte.
Hardware	Hochauflösende Displays, Smartboards, tableTop mit Touchfunktion. Notebooks über WLAN eingebunden.	CommChair, ConnectTable, DynaWall, InteracTable mit Touchfunktion. ViewPort (z.B. PDA).
Beispielanwendung	Präsentationssoftware für die koordinierte Darstellung von Objekten auf mehreren Displays.	„MagNet“ für hierarchisches Sortieren von virtuellen Karteikarten. „Palmbeach“ und „BeachMap“ ermöglichen autonomes Erstellen von Karten mit PDAs und deren Koordination.
Sicherheit	Keine Sicherheitsimplementierungen, gleiche Rechte für alle Benutzer.	Benutzeridentifizierung.

[a] „iROS“: Interactive Room Operating System

[b] Tuplespace: Globaler, langlebiger Speicher, in dem Anwendungen Tuples zur Koordination mit anderen Anwendungen abspeichern können. Ähnlich dem Blackboard aus dem Bereich Künstliche Intelligenz. Tuple ist eine Ansammlung von strukturierten Daten (type-value field), die von jeder Anwendung in den Tuplespace geschrieben, gelesen und gelöscht werden können.

Tabelle 2.1: Einige Daten über iRoom und Roomware im Vergleich (in Anlehnung an Köckritz (2007))

Kategorie	**zentralisiert (C/S)**	**repliziert (P2P)**
Allgemein:	***Server:*** Funktionalität und Daten. ***Client:*** verbunden mit Server-Komponenten des CSCW-Systems	***Client:*** Anwendungskopie mit Systemkomponenten, Funktionalität und Anwendungszustand.
Beispiel:	Application Sharing	Shared Whiteboard
Datenhaltung:	***Server:*** Master-Kopie des Zustandes	Jede ***Anwendungsinstanz*** verwaltet den Zustand
Zustandsänderung:	- **C** sendet **U** an **S** - **S** sammelt und serialisiert **U** aller **C** - **S** führt Zustandsänderungen aus - **S** übermittelt Zustandsänderungen an alle **C**	- Urheber (**B**) sendet **U** an alle anderen Instanzen (**B**) - Jede Instanz (**B**) sammelt und serialisiert alle Zustandsänderungen - Jede Instanz (**B**) schreibt den Zustand fort

Tabelle 2.2: Systemarchitekturen im Vergleich (Legende: C = Client, S = Server, U = Änderung/Update, B = Benutzer)

Ebene	Beschreibung	App.-Ebene	Dienstkl.	Beispiel
Netzwerk	Routing von Anfragen über das physikalische Netzwerk in applikationsunabhängiger Weise	Internet	Routing	TCP/IP, DNS
Datenzugriff	Suche und Änderung von Ressourcen mit Hilfe von applikationsspezifischer Zugriffsstrukturen	Overlay-Netzwerke	Ressourcen-Lokation „Data sharing“	Gnutella, Freenet
Dienst	Kombination und Erweiterung von Funktionalitäten der Datenzugriffsebene, um höherwertige Dienste zur Verfügung zu stellen. Die Dienste können von einfachen File-Sharing bis hin zu komplexen Geschäftsprozessen reichen	P2P-Applikationen	Nachrichtendienste, verteilte Bearbeitung	Napster, Groove, Skype
Benutzer	Gruppierung von mehreren Benutzern („Communities“) und Unterstützung von Benutzer-Interaktionen unter der Verwendung der Dienstebene für Community-Management und Informationsaustausch	Benutzer-Communities	Kollaboration	eBay, Ciao

Tabelle 2.3: P2P-Abstraktionsebenen (angelehnt an Hauswirth und Dustdar (2005))

3 Analyse

In einer kollaborativen Umgebung und Zusammenarbeit können verschiedene Sichten kategorisiert werden. Die Powerwall ist die globale Sicht auf das Gesamte und die Notebooks eine lokale Sicht beispielsweise auf einen Teilbereich der gesamten Informationsquelle. In der Masterarbeit von Napitupulu (2008) werden verschiedenen Aspekte im Zusammenhang der Powerwall und dem Arbeiten in verschiedenen Sichtbereichen aufgeführt. Im Kapitel 3.1 werden nur die für die Bachelorarbeit relevanten Punkte angesprochen, weitere Details sind aus der Sekundärquelle zu entnehmen.

Die Rahmenbedingungen und Eigenschaften der Benutzer und des Systems werden im Kapitel 3.2 festgelegt.

In welcher kollaborativen Umgebung sich das Thema dieser Bachelorarbeit bewegt, wird im Kapitel 3.3 beschrieben.

Anschließend werden die Anforderungen an das System mit Usecases im Kapitel 3.4 und ein Beispielszenario im Kapitel 3.5 dargestellt. Das hier vorgestellte Szenario soll eine kollaborative Situation in einem CSCW veranschaulichen.

Zum Schluss wird im Kapitel 3.6 die Editierproblematik bei mehreren Benutzern angesprochen.

3.1 Sichtbereiche

Es werden zwei mögliche Nutzungsformen von Powerwalls unterschieden: zum Einen der Interaktionsgrad und zum Anderen die Form der Informationsdarstellung. Diese können parallel verlaufen und schließen sich nicht gegenseitig aus.

Form der Informationsdarstellung: Von den drei angesprochenen Formen der *einzelnen Anwendung*, der *Desktopumgebung* und der *virtuellen Umgebung* passt die *einzelne Anwendung* auf unser Szenario am Besten. Mehrere Anwendungen in veschiedenen Fenstern darzustellen, wie es die *Desktopumgebung* macht oder das Darstellen von realitätsgetreuen 3D-Modellen (*virtuelle Umgebung*) ist in unseren Fall nicht gefragt.

Interaktionsgrad: Der Interaktionsgrad wird nach *keine Interaktion*, *einfache Interaktion* und *komplexe Interaktion* klassifiziert.

Je größer ein Anzeigemedium ist, desto mehr deckt es den Bereich des menschlichen Sichtfeldes ab. Jedoch hat es je nach Größe der Anzeigefläche Auswirkung auf die Aufnahme und Verarbeitung der Informationen. Dabei wurden u.a. folgende kognitive Vorteile von großen Anzeigemedien innerhalb einer Forschung festgestellt:

- **Unterstützung des Erinnerungsvermögens** (besseres Erinnern an Informationen durch erhöhten Wahrnehmungsraum über große und sichtfeldeinnehmende Anzeigemedien)
- **Ausnutzung der peripheren Sicht** (unterschieden zwischen zentrale und periphere Sicht fokusiert die zentrale Sicht bewusst ein Objekt oder ein Bildausschnitt, während die periphere Sicht die in der Umgebung liegende Information *unbewusst* wahrnimmt)
- und **Unterstützung der räumlichen Orientierung** (das erweiterte Sichtfeld unterstützt die räumliche Orientierung im Vergleich zu kleinen Anzeigemedien, indem sich der Benutzer nicht nur virtuell, sondern auch physisch orientiert und navigiert).

Weiterhin werden die Interaktionskonzepte angesprochen, die Erstens die Navigation durch bzw. über Inhalte und Zweitens die Selektion und Manipulation von Inhalten beinhalten. Powerwalls entspringen einem dem Menschen bereits vertrautem Konzept: das Arbeiten und Interagieren mit großflächigen, senkrechten Medien. Ziele der Forschung bei der Verwendung von Powerwalls ist das benutzen von natürlichen und intuitiven Interaktionstechniken. Daraus ergibt sich, dass bei großen Darstellungsflächen und hohem Informationsgrad als Eingabegeräte die Maus und die Tastatur nicht mehr sinnvoll eingesetzt werden kann. Hierfür gibt es andere Konzepte, die für solche Umstände optimiert sind. In der Ausarbeitung von Napitupulu (2008) kann man die wesentlichen Herausforderungen als Zusammenfassung nachlesen. Es betrifft z.B. die „Verfolgung des Eingabezeigers“ und den „Übergang zwischen Interaktionen“.

Abhängig davon, aus welcher Distanz die Powerwall genutzt werden soll, ergeben sich folgende Betrachtungen:

- Interaktionsebenen
- Nutzungsbereiche
- Konzepte der Interaktion

Im Bereich der Interaktionsebenen wird folgendes Konzept erstellt:

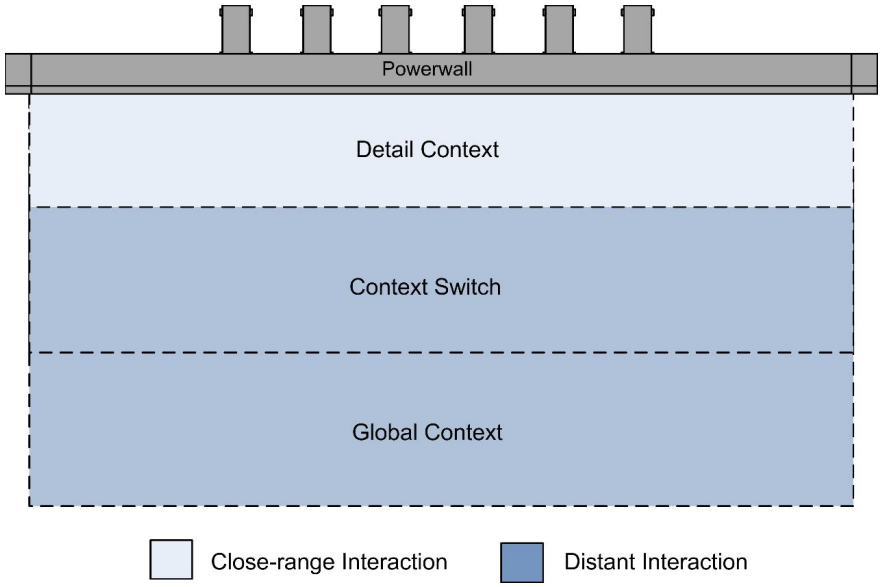

Abbildung 3.1: Ebene der Interaktion und Nutzung [Napitupulu (2008)]

Es gibt zwei Interaktionsebenen (*Close-range* und *Distant Interaction*) und drei Kontextbereiche (*Detail Context*, *Context Switch* und *Global Context*). Je näher man sich der Powerwall befindet, desto feingranulierter werden die Informationen, beim vergrößern des Abstandes zur Powerwall erhält man eine mehr und mehr globale Gesamtansicht. Die *Close-range Interaction* beschreibt Situationen wie an einer Pinnwand oder Whiteboard, an dem die Benutzer mit den Objekten direkt interagieren können. Als neuere Technologie steht der berührungssensitive Bildschirm (entweder mit dem Finger oder mit einem Stift) zur Auswahl. Die *Distant Interaction* ist eine indirekte, „intuitive“ Form der Interaktion. Hierbei werden Objekte nicht direkt angefasst und bearbeitet, statt dessen wird auf die Objekte gedeutet, die bearbeitet werden sollen. Mit sensorischen Hilfsmitteln wie z.B. Kameras und Infrarot-Geräten, die die Position des Zeigers vom Benutzer errechnet, können die Nachteile der klassischen Eingabe- bzw. Zeigegeräte (Tastatur, Maus) entgegengewirkt werden.

3.2 Rahmenbedingungen und Eigenschaften

In den nächsten Unterabschnitten werden die Bedingungen und Eigenschaften der Benutzer in einer Gruppenbesprechung und an die Annahmen über das System gestellt.

3.2.1 Benutzeranforderung

Der Benutzer des Systems sollte mit der Arbeit am PC im Allgemeinen vertraut sein. Alle Personen innerhalb der Zielgruppe, die einem Projekt zugeordnet sind, verfolgen ein gemeinsames Ziel. Die Gruppentreffen finden an einem Ort zu einer bestimmten Zeit statt, um z.B. mit Brainstorming über die Projektthemen zu diskutieren.

3.2.2 Systemannahmen

Die Display-Wand unterstützt den einheitlichen Informationsstand der Teilnehmer. Es kann in einer zukünftigen Version möglich werden, dass mehrere Benutzer sich über ihr eigenes Notebook Zugriff auf die Display-Wand beschaffen und Daten ändern. Dies ist jedoch [nach Dreyer:] mit Problemen verbunden, wie z.B.:

- gleichzeitiges Arbeiten: Änderungen propagieren, Sperren
- Datenpersistenz: lokal <-> global, Arbeiten auf Kopien, Zusammenführen der Daten
- nachträgliche Änderungen einer Kopie

In dieser Bachelorarbeit wird davon ausgegangen, dass sich die Teilnehmer mündlich absprechen, wer neue Inhalte über das eigene Notebook an die Display-Wand übermitteln darf. Alle anderen haben das Lese-Recht und können sich den aktuellen Stand auf dem eigenen Notebook darstellen lassen - vorrausgesetzt, man verändert nicht die inhaltliche Struktur.

3.3 Kollaborative Umgebung

Das Labor an der HAW besteht aus mehreren kollaborativen Hardware- und Software-Komponenten. Zur Verfügung steht:

- eine interaktive Wand -> ein multitouchfähiges 30“-FullHD-Display,
- ein Tabletop, ein physikbasierter Desktop in Form eines Leuchttisches, das multitouchfähig sein soll

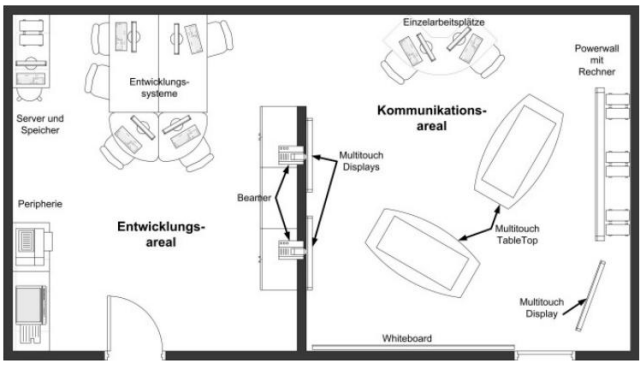

Abbildung 3.2: Übersichtsplan [v. Luck u. a. (2008)]

Abbildung 3.3: Live-Aufnahme [v. Luck u. a. (2008)]

- und die Powerwall, auf die im Unterabschnitt weiter eingegangen wird.

Eine Gesamtübersicht des Labors ist in Abbildung 3.2 zu erkennen. Abbildung 3.3 zeigt Studenten am Tabletop arbeiten. Im Hintergrund befindet sich die Powerwall und die interaktive Wand als 30“-FullHD-Display. In dieser Bachelorarbeit wird von den oben aufgeführten Geräten ausschließlich mit der Powerwall gearbeitet. Im nächsten Abschnitt wird der Aufbau der Powerwall beschrieben.

3.3.1 Aufbau der Powerwall

Die Powerwall wird durch einen Cluster von neun Displays realisiert, die von einem Master-Rechner und 5 Slave-Rechnern als Rendering-Maschinen betrieben wird. Es wird eine Gesamtauflösung von 7500 x 4800 Pixeln erreicht. Zur Verteilung der Anzeige auf die Displays wurde das Chromium-System eingesetzt. [v. Luck u. a. (2008), Napitupulu (2007)]

X-Window-System/X-Server

Das X-Window-System dient der Ansteuerung von grafischen Bildschirmen. Es ist eine Protokollsammlung und stellt ein Framework zum Aufbau der grafischen Benutzeroberfläche auf Unix und ähnlichen Systemen zur Verfügung. Es hat die Eigenschaften, netzwerktransparent zu sein und baut auf dem Client-Server-Modell auf.

Der X-Server steuert die Ein- und Ausgabegeräte, wie z.B. Tastatur, Maus, Bildschirm und Grafikkarten. Er stellt dem X-Client seine grafischen Dienste zur Verfügung.

Der X-Client ist ein Anwendungsprogramm, das die grafischen Ein- und Ausgabedienste des X-Servers benutzt. Der X-Client kann auf dem gleichen Rechner wie der X-Server oder auf einem entfernten Rechner verwendet werden.

Xdmx

Das Xdmx ist ein X-Server Proxy, der die Verwaltung von mehreren X-Server auf verschiedenen Rechnern in einem Netzwerk möglich macht. Ein einzelner X-Server kann zwar mehrere Anzeigegeräte verwalten, jedoch müssen alle Bildschirme an dem gleichen Rechner angeschlossen sein, auf dem auch der X-Server läuft. Mit Xdmx als Proxy kann man die physikalische Einschränkung durch die simulative Verwendung mehrerer verteilter X-Server umgehen. Damit kann man die Anzahl der parallel verwendbaren Anzeigegeräte wesentlich steigern.

Chromium

Chromium ist ein System für interaktives Rendering auf Clustern von Arbeitsplätzen.

Funktionen von Chromium sind u.a.:

- Sort-first (Fliesen)-Rendering: der Frame-Puffer ist unterteilt in rechteckige Fliesen, die vom Hosts eines Rendering-Cluster parallel gerendert werden.
- Sort-last (Z-Zusammensetzung)-Rendering: der 3D-Datensatz ist in *N* Teile zerteilt, die parallel von *N* Prozessoren gerendert werden. Das Resultatbild ist eine Zusammensetzung entsprechend dem *Z* Puffer, um das fertige Bild abzubilden.
- Parallel Hybrid-Rendering: Sort-First und Sort-Last-Rendering kann in eine Hybrid-Konfiguration kombiniert werden.
- OpenGL-Befehl Stream Filter - OpenGL-Befehl-Streams können durch einen Stream Processing Unit (SPU) abgefangen und verändert werden - zur Durchführung nicht-fotorealistischer Rendering (NPR) Effekte, etc.
- Viele OpenGL-Programme können unverändert mit Chromium verwendet werden.
- Man kann Chromium-spezifische Anwendungen schreiben, die paralleles Rendering mit Hilfe von speziellen Synchronisierungs-Primitiven ausführen.
- Chromium läuft unter Linux, IRIX, AIX, SunOS und unter Windows-basierte Systeme.
- Chromium ist ein Open-Source Project.

Wie auch Xinerama, ist Chromium eine Erweiterung des X-Window-Systems. Alle von einem X-Server verwalteten Anzeigegeräte werden zu einem großen, virtuellen Desktop zusammengelegt. Der große Vorteil im Vergleich zum Xinerama zeichnet sich im Bereich von OpenGL-Anwendungen aus. Wo bei Xinerama das OpenGL-Rendering nur auf ein Bildschirm beschränkt ist, kann bei Chromium das OpenGL-Rendering über den gesamten virtuellen Desktop dargestellt werden. Dabei werden die Rendering-Aufgaben von einem zentralen Rechner zerlegt und an die jeweils zuständigen Back-End Rechner verteilt. Dank des Transfers der OpenGL-Befehle erhält man das Bild schneller.

Im Labor ist die Kombination aus X-Server, Xdmx und Chromium realisiert worden.

Die technische Realisierung kann man in der Abbildung 3.4(a) erkennen. Nebenan einige Fotos 3.4, in denen die Powerwall abgebildet ist. In der Aufnahme 3.4(d) kann man sich das Beispielszenario dieser Ausarbeitung im kleinen Rahmen vorstellen.

Wie auf der Abbildung 3.4(a) zu erkennen ist, wird Chromium als Erweiterung des X-Server Proxies eingesetzt. Die Ansteuerung der neun Bildschirme wird mithilfe von fünf Back-End Rechnern realisiert. An vier dieser Back-End Rechner sind jeweils zwei Bildschirme - am fünften Rechner ein Bildschirm - angeschlossen, die über den lokalen X-Server verwaltet werden. Auf einem anderen Rechner, der über Netzwerk mit allen fünf Back-End Rechnern verbunden ist, wird Xdmx mit dem X-Server Proxy betrieben. Damit werden auf diesem Rechner alle neun Bildschirme ansteuerbar. Um diese neun Bildschirme einheitlich als einen Desktop bedienen zu können, wird Xinerama oder Chromium als Erweiterung auf das X-Server Proxy eingesetzt.

Dem System ist bei Xinerama unbekannt, dass sich hinter dem X-Server Proxy mehrere X-Server verbergen. Bei Chromium ist es glücklicherweise anders. Deshalb werden die Renderprimitiven über das Netzwerk direkt an die Back-End Rechner geschickt, um dadurch eine weitere Performanzsteigerung zu erzielen.

Die Eingaben werden über Clients - die Notebooks der Gruppenteilnehmer, die spontan ins Netzwerk eingebunden werden können - getätigt. Zur Ausgabe der Daten wird die Powerwall verwendet. Die Powerwall wurde bereits im Kapitel 3.3.1 näher beschrieben.

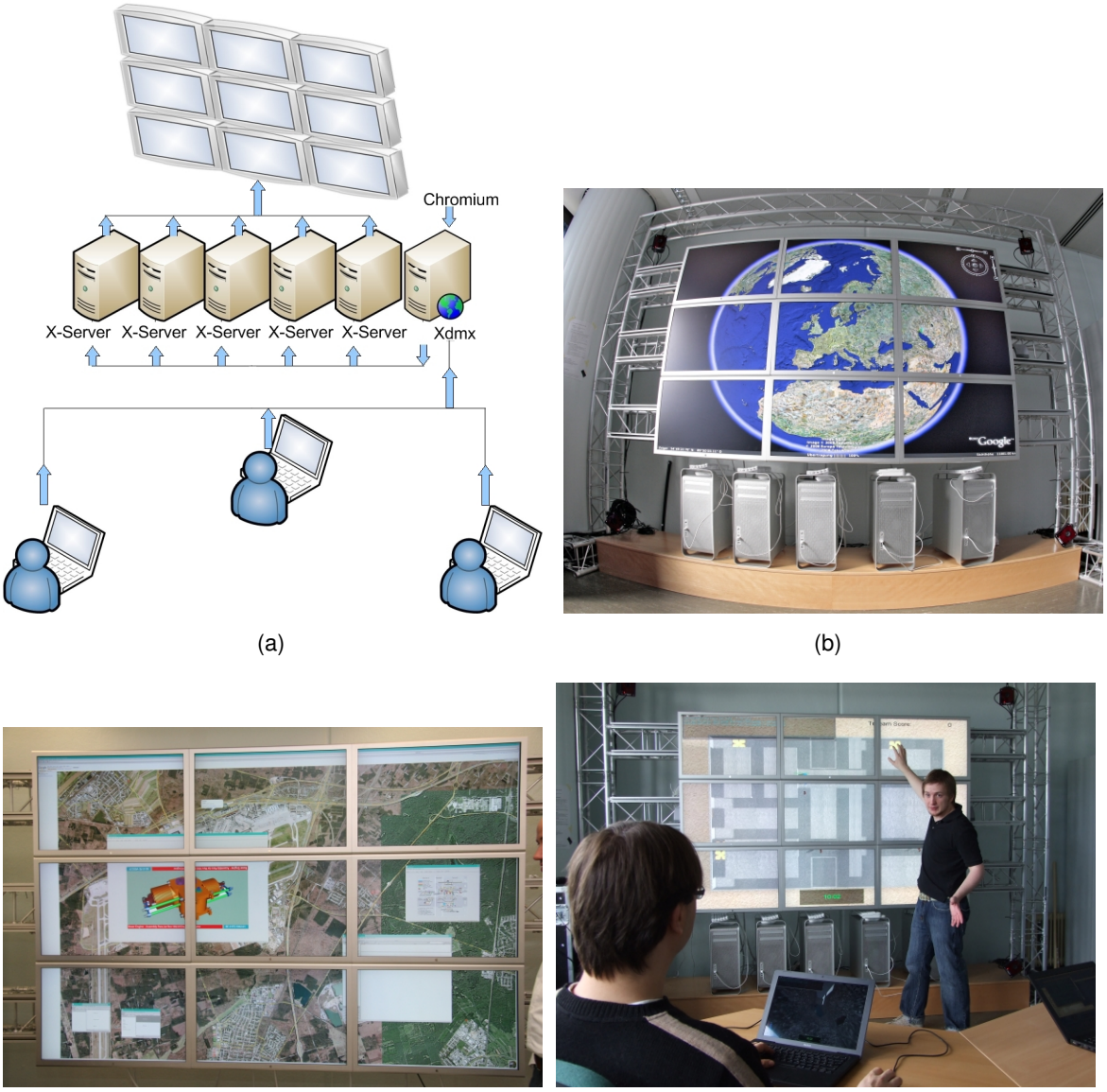

Abbildung 3.4: Powerwall im CSCW-Labor

3.4 Anforderungen

Die bereits vorliegenden Anwendungsfälle von Carfagno (2007) sind folgend zusammengefasst:

- Mind Map erstellen
- Titel bearbeiten
- Unterknoten erstellen
- Unterknoten entfernen
- Unterknoten verschieben
- Knoten expandieren / kollabieren
- Knoten vollständig expandieren
- Rahmen festlegen
- Link erstellen
- Link folgen
- Link entfernen
- Notiz erstellen
- Notiz anzeigen
- Notiz entfernen

Um hier die Details zu erfahren, kann man in Carfagno (2007) die jeweiligen Anwendungsfälle mit den Akteuren, die in Benutzer und Mind Map unterschieden werden und die Anwendungsfallbeschreibungen lesen. Es folgen sechs Beispiel-Interaktionen. Die Tabelle 3.1 beschreibt das erstellen eines Mind Maps, Tabelle 3.2 beschreibt das Szenario „Titel bearbeiten“ und die Tabelle 3.3 führt das Erstellen eines Unterknotens auf. In der Abbildung 3.5 wird das Erstellen eines Unterknotens bildhaft veranschaulicht. Weiterhin werden die Szenarien „Unterknoten entfernen“ (Tab.: 3.4, Abb.: 3.6) und „Unterknoten expandieren/kollabieren“ (Tab.: 3.5, Abb.: 3.7) gezeigt.

Schritt	Akteur	Beschreibung
1	B	gibt die URL von M in die Browseradressleiste ein.
2	M	stellt den Zentralknoten im Zentrum des Browserfensters dar. Der Zentralknoten besitzt den Titel „Edit this node“ und besitzt einen Kreisrahmen.

Tabelle 3.1: Anwendungsfall: Mind Map erstellen

Schritt	Akteur	Beschreibung
1	B	klickt mit rechter Maustaste auf den Knoten dessen Titel bearbeitet werden soll.
2	M	zeigt Popup-Menü über dem angeklickten Knoten an.
3	B	klickt auf Menü-Eintrag „Edit Title“.
4	M	zeigt an Stelle des Knotens ein Texteingabefeld an.
5	B	schreibt den Titel in das Texteingabefeld und bestätigt mit der Taste „Enter“.
6	M	zeigt den eingegebenen Titel für den bearbeiteten Knoten an.

Tabelle 3.2: Anwendungsfall: Titel bearbeiten

Schritt	Akteur	Beschreibung
1	B	klickt mit rechter Maustaste auf den Knoten von dem ein Unterknoten erstellt werden soll.
2	M	zeigt Popup-Menü über den angeklickten Knoten an.
3	B	klickt auf Menü-Eintrag „Create Subnode“.
4	M	expandiert den Knoten und zeigt an der späteren Position des zu erstellenden Unterknotens ein Texteingabefeld an. Es wird eine Kante zwischen dem angeklickten Knoten und dem Texteingabefeld gezeichnet.
5	B	schreibt den Titel für den zu erstellenden Unterknoten in das Texteingabefeld und bestätigt mit der Taste „Enter“.
6	M	zeigt den neuen Unterknoten mit dem eingegebenen Titel an. Der Unterknoten besitzt einen Unterlinierahmen.

Tabelle 3.3: Anwendungsfall: Unterknoten erstellen

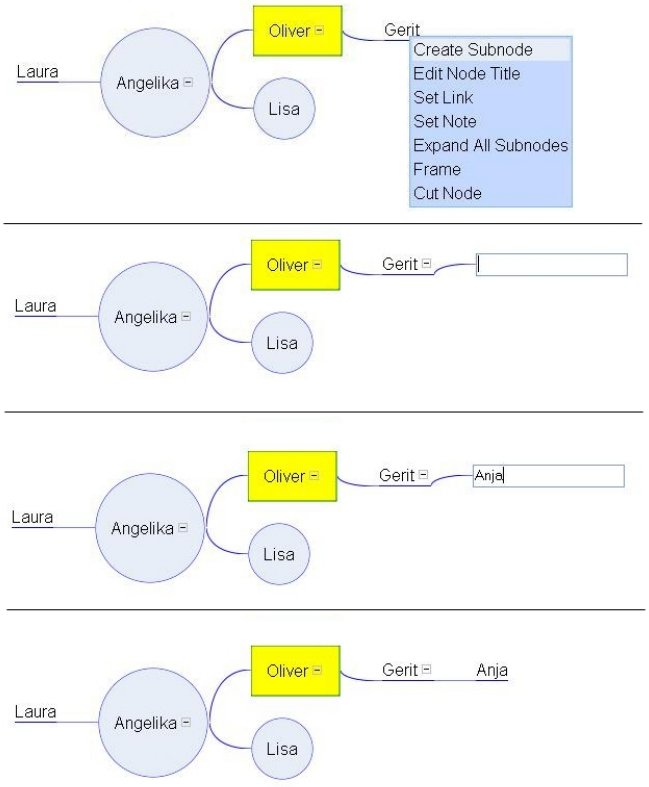

Abbildung 3.5: Unterknoten erstellen

Schritt	Akteur	Beschreibung
1	B	klickt mit rechter Maustaste auf den Knoten von dem ein Unterknoten entfernt werden soll.
2	M	zeigt Popup-Menü über den angeklickten Knoten an.
3	B	klickt auf Menü-Eintrag „Cut Node“.
4	M	entfernt den Unterknoten mit allen darunterliegenden Unterknoten.

Tabelle 3.4: Anwendungsfall: Unterknoten entfernen

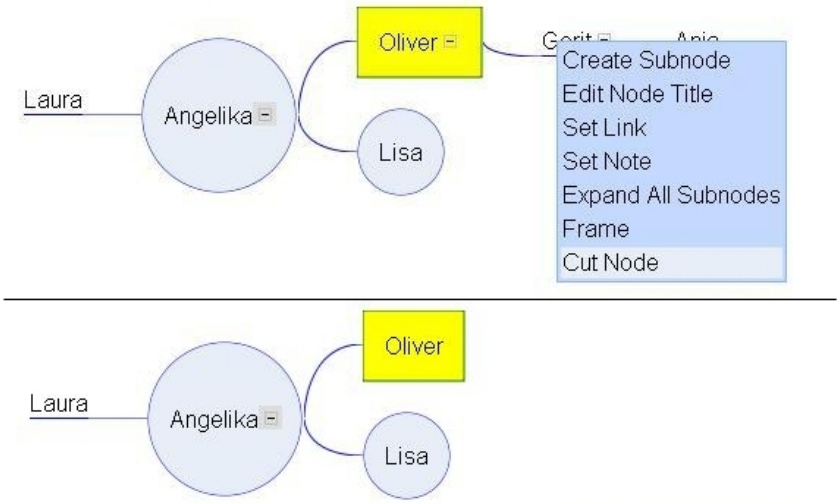

Abbildung 3.6: Unterknoten entfernen

Schritt	Akteur	Beschreibung
1	B	klickt mit linker Maustaste auf den Knoten, der expandiert/kollabiert werden soll.
2	M	expandiert den Knoten, falls der Knoten kollabiert ist. Ist der Knoten expandiert, wird kollabiert.

Tabelle 3.5: Anwendungsfall: Unterknoten expandieren/kollabieren

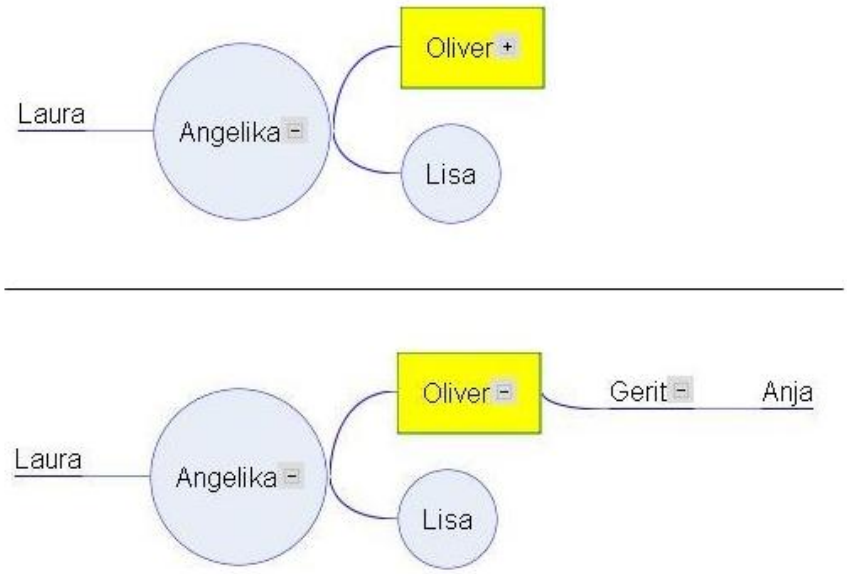

Abbildung 3.7: Unterknoten expandieren

Neue Anforderungen sind:

- Anzeige auf der Powerwall
- Bereitstellen vom Mind Map im View-Modus auf anderen Geräten, z.B. Notebook
- Bereitstellen vom Mind Map im Edit-Modus, z.B. auf einem Notebook
- Abgabe der Schreibrechte, d.h. die Moderatorrolle wechselt zu einem anderen Benutzer

3.4.1 Anzeige auf der Powerwall

Beispieltabelle 3.6:

Schritt	Akteur	Beschreibung
1	B	gibt die URL der M (View-Sicht) in die Adresszeile des Browsers ein und bestätigt
2	B	drückt „Connect View“-Button
3	M	zeigt eine leere Mind Map mit einem Wurzelknoten in der Mitte
4	M	refresht in regelmäßigen Zeitabständen die Mind Map, wenn sie verändert wurde

Tabelle 3.6: Anwendungsfall: Anzeige auf der Powerwall

Der Schritt 4 wiederholt sich beliebig oft.

3.4.2 Bereitstellen vom Mind Map im View-Modus

Beispieltabelle 3.7:

Schritt	Akteur	Beschreibung
1	B	gibt per Notebook die URL der M (View-Sicht) in die Adresszeile des Browsers ein und bestätigt
2	B	drückt „Connect View“-Button
3	M	zeigt die aktuelle Mind Map an, parallel zum Mind Map, das auf der Powerwall zu sehen ist
4	M	refresht in regelmäßigen Zeitabständen die Mind Map, wenn sie verändert wurde

Tabelle 3.7: Anwendungsfall: Bereitstellen vom Mind Map im View-Modus

Der Schritt 4 wiederholt sich beliebig oft - solange die Anwendung auf dem Notebook läuft.

3.4.3 Bereitstellen vom Mind Map im Edit-Modus

Beispieltabelle 3.8:

Schritt	Akteur	Beschreibung
1	B	gibt per Notebook die URL der M (Edit-Sicht) in die Adresszeile des Browsers ein und bestätigt
2	M	zeigt die aktuelle Mind Map an
3	B	ändert ggf. den Inhalt der Mind Map
4	M	refresht bei Änderungen in regelmäßigen Zeitabständen die Mind Map

Tabelle 3.8: Anwendungsfall: Bereitstellen vom Mind Map im Edit-Modus

Die Schritte 3 und 4 können sich beliebig oft wiederholen.

3.4.4 Anwendungsfallbeschreibung: Moderator-Wechsel/Abgabe Schreibrechte

Wechsel der Rolle des Bearbeiters eines gemeinsamen Dokuments zwischen zwei Benutzern:

Zielbeschreibung: Wechsel der Benutzer in die Rolle des Bearbeiters. Der Modus zum Editieren wird vom bisherigen Benutzer verlassen und der Benutzer wechselt in den View-Modus. Der neue Benutzer kann dann in den Editier-Modus gehen.
Vorbedingung: Ein Benutzer hat die Rolle des Bearbeiters und möchte diese an einen anderen Benutzer abgeben. Ein zweiter Benutzer möchte diese Bearbeiter-Rolle erhalten.
Erfolgsbedingung: Die Rolle zum Editieren wurde an den zweiten Benutzer abgegeben und auf den Edit-Modus zugegriffen.
Benutzerrollen: Ein Benutzer in der Editier-Rolle und mindestens ein Benutzer in der View-Rolle.
Auslöser: Der vorige Benutzer hat einige Arbeitsschritte im gemeinsam genutzten Dokument - in diesem Fall das Mind Map - getätigt und der aktuelle Benutzer möchte darauf aufbauen.
Beschreibung:

1. Erster Benutzer hat Arbeitsschritte im gemeinsamen Dokument beendet.
2. Erster Benutzer ist bereit für die Abgabe der Editier-Rolle.
3. Erster Benutzer wechselt in den View-Modus.
4. Zweiter Benutzer nimmt die Editier-Rolle an.
5. Zweiter Benutzer wechselt in den Edit-Modus.
6. Zweiter Benutzer tätigt weitere Arbeitsschritte im gemeinsamen Dokument.

Beispieltabelle 3.9:

Schritt	Akteur	Beschreibung
1	B1	gibt die URL der M (Edit-Sicht) frei, indem er die Seite verlässt (neue URL eingeben oder Fenster schliessen)
2	B1	kann anschließend die URL der View-Sicht eingeben, um das weitere Geschehen mitzuverfolgen
3	B2	gibt nun die URL der Edit-Sicht in die Adresszeile ein und bestätigt

Tabelle 3.9: Anwendungsfall: Abgabe Schreibrechte, Moderatorwechsel

Nun ist der B2 in der Lage, weitere notwendige Aktionen am Mind Map durchzuführen, um die Diskussion fortzuführen.

In diesem Anwendungsfall wird sichergestellt, dass - ohne Verwendung von Software-Lösungen, wie Schreibrechte - nur eine von mehreren Personen zur Zeit Zugriff auf die Mind Map (bzw. auf den entsprechenden Teil des Mind Maps) hat.

3.4.5 Automatisches Layout

Zusätzlich wird über die Anforderung eines automatischen Layouts nachgedacht. Gemeint ist der Zustand, dass alle Sichten bei Änderung eines Baumes geändert werden. Wenn ein Zweig gelöscht wird, aktualisieren sich alle Sichten auf diesen Baum.

Grund für die Überlegung ist Folgendes:
Gehen wir von der Situation aus, dass zwei Personen an dem Mind Map editieren. Wir verwenden das vereinfachte Beispiel von Abbildung 3.8. Es werden von beiden Personen verschiedene Zweige bearbeitet. Somit hat jeder seinen eigenen Bereich, den er editiert. Während Person A an einem Unterknoten von dem Zweig „Lisa“ etwas ändert, z.B. einen neuen Unterknoten hinzufügt, löscht Person B einen anderen Zweig, in diesem Fall „Laura“. In diesem Moment balanciert sich der Baum neu und der Zweig der ersten Person wechselt auf die linke Seite. Bei der Annahme, dass der Baum sehr komplex ist, verliert diese Person die Sicht auf seinen Zweig innerhalb seines Ausschnitts und ist vermutlich zunächst verwirrt.

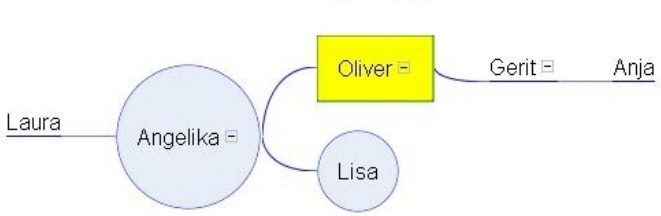

Abbildung 3.8: Beispiel eines Mind Maps

Ein möglicher Lösungsansatz wird im Kapitel 4.1.4 besprochen. Jedoch wird es in der Implementierung des aktuellen Prototyps nicht weiter verfolgt.

3.5 Allgemeines Beispielszenario

Durch Neumann (2006) inspiriert, stellen wir uns vier fiktive Personen vor: A, B, C und D. Diese Personengruppe trifft sich zu einer Besprechung im Sinne des CCW. Jeder der Teilnehmer hat sein Notebook (zur lokalen Sicht) dabei und kann sich mit dem lokalen Netzwerk verbinden. Die Clientsoftware - in unserem Fall der Browser und die Mind Map Anwendung - die nicht mehr auf den einzelnen Notebooks installiert werden muss, ist die Diskussionsgrundlage. Es gibt zwei Zustände des Mind-Maps auf den Notebooks: den Edit-Modus (schreibender Zugriff) und den Lese-Modus (nur lesender Zugriff). Auf der Powerwall wird der Lese-Modus der Mind-Map-Anwendung betrieben.

Zu Beginn der Diskussion wird eine Person als Gruppenleiter festgelegt, in diesem Fall B. Diese Person hat durch Absprache das Recht auf schreibenden Zugriff der Mind Map Anwendung. Die anderen drei Personen nehmen den Lese-Modus in Anspruch und projizieren den aktuellen Stand des Mind-Maps auf ihre Notebooks.

Jeder Begriff, der im Brainstorming zwischen den vier Teilnehmern fällt, wird von B über sein Notebook (im Edit-Modus) in den Mind Map Baum als Knoten eingefügt. Bei jeglicher Veränderung des Mind Map Baums wird auch die Ansicht des Mind-Maps auf der Powerwall und den Notebooks aktualisiert. Somit kann die Gruppendiskussion anhand der aktuellen Informationen auf der Powerwall, sowie lokal auf den Notebooks fortgesetzt werden.

Zu einem späteren Zeitpunkt entscheidet B, dass er am Mind Map keine editierende Rolle mehr einnehmen will. Demnach muss sich die Gruppe auf eine neue Person einigen, die manipulierende/editierende Rechte am Mind-Map haben soll. In diesem Fall A. Er wechselt an seinem Notebook in den Edit-Modus und kann dann Veränderungen am Mind-Map vornehmen, während B wieder in den Lese-Modus wechselt. Der Rollenwechsel zwischen Edit-Modus und Lese-Modus kann beliebig oft durchgeführt werden, solange die mündliche Einigung eingehalten wird, dass nur eine Person editieren darf.

Das Resultat der Gruppendiskussion und des vorangegangenen Brainstormings ist ein Baum, der mit den verschiedenen Knoten und Verzweigungen die Inhalte des Gesprächs aufweist.

3.6 Fazit

Im vorigen Kapitel wurde ein Szenario dargestellt, das korrekt nach den mündlich abgesprochenen Regeln verläuft. Es ist jedoch eine Lösung notwendig, falls zwei Personen zeitgleich im Editiermodus an demselben Knoten(strang) etwas ändern. Wir nehmen für dieses Beispiel die Abbildung 3.9 als Vorlage mit den Personen A und B.

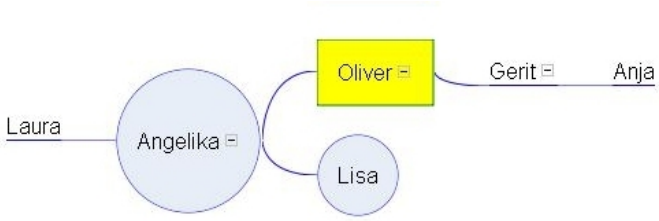

Abbildung 3.9: Beispiel eines Mind Maps

A möchte an den Knoten „Anja“ einen Unterknoten „Hannah“ anhängen. B löscht in dieser Zeit den Knoten „Gerit“, der alle jeweiligen Unterknoten beinhaltet. Hierbei lösen diese zwei Aktionen ein Kollisionsproblem aus, wofür Lösungsmöglichkeiten gefunden werden müssen. Dabei kann die Verwaltung der Eingabeverteilung nützlich sein, um diese Problematik bei unterschiedlichen Sichten auf den selben Systemdatenbestand zu beheben.

4 Design und Realisierung

Das Mind Map System ist eine AJAX-Anwendung, die auf eine Client-Server-Architektur basiert. Dennoch soll sie aufgrund ihrer fehlenden Authentifikation und Autorisierung nicht öffentlich für das Internet zur Verfügung stehen. In Kapitel 4.1 werden die Systemarchitekturen vorgestellt. Dabei wird das Client-Server-Modell, das Schichtenmodell und der Model-View-Controller im Zusammenhang mit dem Mind Map System besprochen.

Um die verschiedenen Sichten der Powerwall und der Notebooks zu synchronisieren, müssen Server und Clients miteinander kommunizieren. Dies ist das Hauptthema in der Realisierung und wird im Abschnitt Kapitel 4.3 erläutert.

Anschließend wird in Kapitel 4.4 ein Resümee gezogen.

4.1 Systemarchitektur

Die allgemeinen Beschreibungen von „Client-Server-Modell“ Kapitel 4.1.1 und „Model-View-Controller“ Kapitel 4.1.6 in den folgenden Unterabschnitten wurden Napitupulu (2008) entnommen.

In den Unterabschnitten Kapitel 4.1.2 und Kapitel 4.1.7 werden die verwendeten Architekturen in Bezug zum Mind Map System geschildert. Zwischendurch wird das Schichtenmodell in Kapitel 4.1.3 kurz erläutert und in Kapitel 4.1.4 auf das Mind Map System bezogen.

4.1.1 Client-Server-Modell

Das Client-Server-Modell beschreibt eine Möglichkeit, Aufgaben und Dienstleistungen innerhalb eines verteilten Systems zu verteilen. Die Aufgaben werden von Systemkomponenten erledigt, die in Clients und Server unterteilt werden. Der Client kann auf Wunsch eine Aufgabe vom Server anfordern, wie z.B. ein Betriebsmittel. Der Server, der sich auf einem

beliebigen anderen Rechner im verteilten System befindet, beantwortet die Anforderung, d.h. er stellt z.B. das gewünschte Betriebsmittel bereit.

Allgemeines Client-Server-Modell

Das Client-Server-Modell ist das Standardkonzept für die Verteilung von Aufgaben innerhalb eines Netzwerkes. Aufgaben werden mittels Server auf verschiedenen Knoten in einem Netzwerk verteilt und können bei Bedarf von mehreren Clients zur Lösung ihrer eigenen Aufgaben angefordert werden. Bei den Aufgaben kann es sich um Standardaufgaben (z.B. E-Mail- oder Webseiten-Zugriffe) oder um spezifische Aufgaben eines Softwaresystems handeln. Eine Aufgabe wird im Client-Server-Modell als Dienst bezeichnet.

Die Abbildung 4.1 zeigt die Aufteilung der Verantwortlichkeiten in dem Client-Server-Modell. Ein Server ist ein System, das einen Dienst (Service) anbietet. Im Rahmen des Client-Server- Konzepts kann ein anderes System, der Client, diesen Dienst nutzen. Die Kommunikation zwischen Client und Server ist abhängig vom Dienst. Er bestimmt, welche Daten zwischen beiden ausgetauscht werden. Der Server ist in Bereitschaft, um jederzeit auf die Kontaktaufnahme eines Clients reagieren zu können. Im Unterschied zum Client, der aktiv einen Dienst anfordert, verhält sich der Server passiv und wartet auf Anforderungen. Die Regeln der Kommunikation für einen Dienst (Format, Aufruf des Servers, und die Bedeutung der zwischen Server und Client ausgetauschten Daten), werden durch ein Protokoll festgelegt. Das Protokoll ist spezifisch für den jeweiligen Dienst.

Abbildung 4.1: Allgemeines Client-Server-Modell [Abb. aus Napitupulu (2008)]

Clients und Server können als Programme auf verschiedenen Rechnern oder auf demselben Rechner ablaufen. Allgemein kann das Konzept zu einer Gruppe von Servern, die eine Gruppe von Diensten anbietet, ausgebaut werden (Mail-Server, Web-Server, Applikations-Server oder Datenbank-Server).

Da in der Praxis diese Server meist gesammelt auf bestimmten Rechnern laufen, hat es sich eingebürgert, diese Rechner selber als Server zu bezeichnen.

Kompetenzverteilung auf Client und Server

Die Schichten der Daten, Anwendungslogik und Darstellung lässt sich in verschiedenen Kombinationen auf dem Server und dem Client verteilen, wie in der Abbildung 4.2 zu erkennen ist. Hier wird der Unterschied zwischen den klassischen Web-Anwendungen und RIAs deutlich abgebildet.

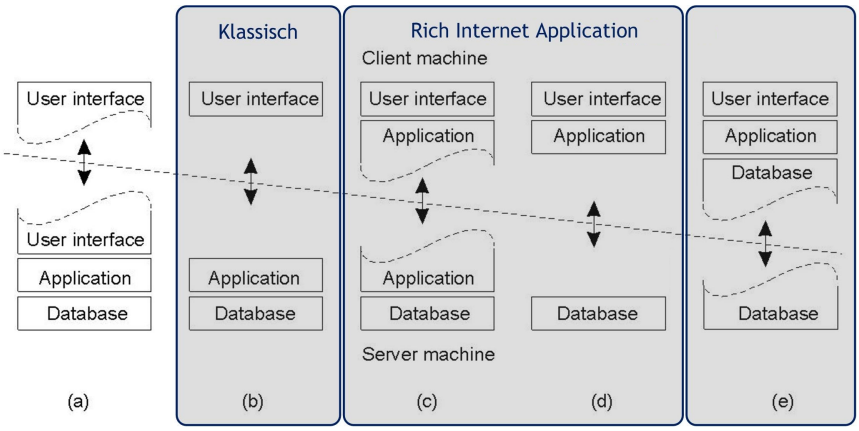

Abbildung 4.2: Kompetenzverteilung [nach Hartmann (2007) wird Tanenbaum und van Steen 2003 zitiert]

a) Darstellung liegt sowohl auf Client- als auch auf der Serverseite, Anwendungslogik und Daten liegen auf der Server-Seite

b) Darstellung liegt auf der Client-Seite, Anwendungslogik und Daten liegen auf der Server-Seite

c) Darstellung und ein Teil der Anwendungslogik liegen auf der Client-Seite, der andere Teil der Anwendungslogik und die Daten liegen auf der Server-Seite

d) Darstellung und Anwendungslogik liegen auf der Client-Seite, Daten auf der Server-Seite

e) Darstellung, Anwendungslogik sowie ein Teil der Daten liegen auf der Client-Seite, der restliche Teil der Daten auf der Server-Seite

4.1.2 Client-Server-Modell im Mind Map System

Aus Benutzersicht: der Client ist der vom Benutzer verwendete Browser, der über die URL auf die Mind Map Anwendung zugreift.

Aus Sicht der Implementierung: Ein Großteil der Mind Map Anwendung wurde clientseitig implementiert. Der Server kümmert sich um die Synchronisierung aller Clients auf Kommunikationsbasis und ist im Package `server` und `service` enthalten. Die Funktionsweise geht nach dem RPC-Prinzip[1] und ist serverseitig.

Das RPC im GWT erfordert u. a. Service-Klassen. Die `ServiceImpl`-Klasse, die einem `RemoteServiceServlet` entspricht, implementiert das Interface `Service`, die eine Unterklasse von `RemoteService` ist. Die `ServiceAsync`-Klasse wird clientseitig verwendet bzw. aufgerufen. Im Kapitel 4.3.1 wird im Detail auf diese Serviceklassen eingegangen.

4.1.3 Schichtenmodell

Das Architekturmuster des Mind Map Systems nennt sich „Separated-Presentation“ und teilt sich in eine Domänen- und eine Präsentationsschicht auf. In der Domänenschicht ist die gesamte Logik (z.B. Verarbeitung von Daten) enthalten. GUI-Elemente und die Interaktionen zwischen dem Benutzer und der Software wird von der Präsentationsschicht geregelt. Da in dieser Prototyp-Version keine langfristige Datenpersistenz vorhanden ist, spricht man in diesem Fall aufgrund der fehlenden Schicht der Persistenz nicht von der „3-Schichten-Architektur“. Die Lebensdauer der Daten ist in der momentanen Anwendungsversion auf die Laufzeit beschränkt.

Die Domänenschicht kann für die Entwicklung neuer Benutzerschnittstellen wiederverwendet werden. Die Wiederverwendbarkeit von einzelnen Komponenten ist ein zentraler Vorteil im Schichtenmodell. Es führt dazu, dass Änderungen an einzelnen Modulen nur geringe bis keine Auswirkung auf das gesamte System haben. Dies wiederum begünstigt eine bessere Wartbarkeit. Einzelne Komponenten können einfach unter Beibehaltung der Schnittstellen ausgetauscht werden.

In Abbildung 4.3 hat man dazu eine kleine Übersicht.

[1] Remote Procedure Calls

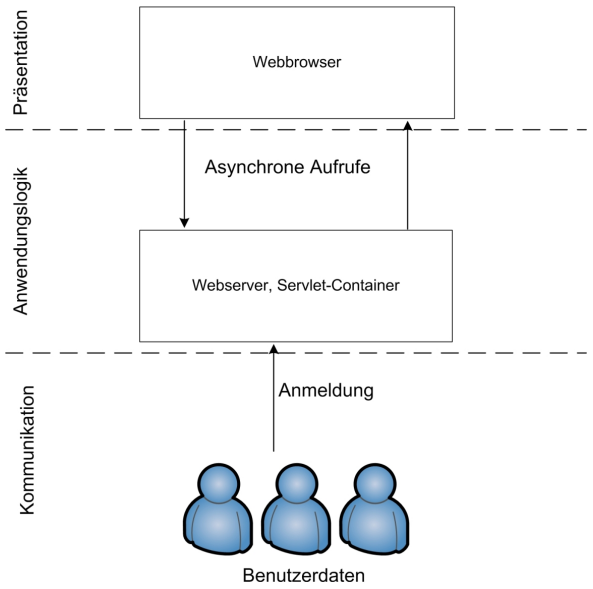

Abbildung 4.3: Einteilung der Anwendung in Schichten

4.1.4 Schichtenmodell im Mind Map System

Die nachfolgenden Abschnitte beschreiben die Architektur des Mind-Maps und können in der Sekundärliteratur Carfagno (2007) detaillierter nachgelesen werden. In der Diplomarbeit von Virginio Carfagno wird u. a. nachgeforscht, ob mit dem GWT nach dem State-of-the-Art-Entwurfsprinzipien entwickelt werden kann. Im Kapitel „Systementwurf“ wird nach dem Prinzip der Objektorientierung gehandelt. Die Mind Map Anwendung wird in mehrere Layer nach dem Architekturmuster „Separated-Presentation“[2] organisiert:

- Domänenlogik
- Präsentationslogik

Es besteht eine gerichtete Abhängigkeitsbeziehung, die Präsentationsschicht darf auf die Domänenschicht zugreifen, umgekehrt jedoch nicht.

Es ist möglich, dass sich mit der Zeit herausstellt, z.B. die grafische Oberfläche zu verändern oder mit anderen Mitteln zu implementieren. Die Mind Map Anwendung wurde mit dem GWT-Framework erstellt und bringt damit größtenteils eigene grafische Elemente mit. Wird zukünftig festgestellt oder entschieden, dass die Benutzerschnittstelle, die auf GWT basiert, nicht mehr für die Anwendung geeignet ist und man sich anderweitig orientiert, z. B. Swing

[2] Spezialfall von der Schichtenarchitektur/Layer-Architekturmuster

oder SWT, muss ein einfacher Weg existieren, dies möglichst problemlos zu ersetzen.

Dieses Architekturmuster hat den Vorteil, dass die Anforderung für eine neue Benutzerschnittstelle erfüllt ist. Wenn eine Absicht besteht, eine neue Schnittstelle zu entwickeln, kann alles aus der Domain-Schicht verwendet werden. Dieser Vorteil wird durch die starke Trennung der Präsentations- und Domänenlogik realisiert. Mehr Details zu dieser speziellen Abwandlung des Layer-Musters kann man in der Diplomarbeit von Carfagno (2007) im Kapitel 4 nachlesen.

Die beiden Schichten wurden nach der Ausarbeitung von Virginio Carfagno in weitere Schichten zerlegt. Es existieren zusätzlich die Schichten GUI und Foundation.

Basisklassen, wie z. B. Strings, Integer, Collection werden in der Regel bei einer Anwendungsentwicklung benötigt und sind in der Foundation-Schicht angesiedelt.

Die GUI-Schicht enthält viele wiederverwendbare Elemente, wie z. B. Menüs, Eingabefelder, Schaltflächen.

In Abbildung 4.4 werden alle Schichten der Mind Map-Anwendung mit ihren Abhängigkeitsbeziehungen dargestellt.

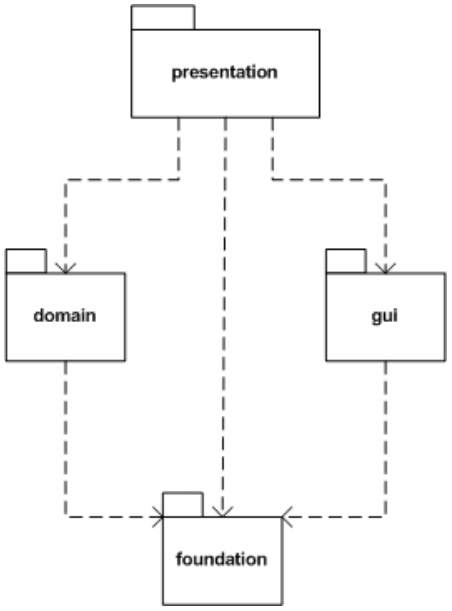

Abbildung 4.4: Schichten der Mind Map-Anwendung [nach Carfagno (2007)]

Im Folgendem werden die einzelnen Schichten etwas genauer betrachtet. Da in der Schicht Foundation Basiskomponenten wie Integer, Boolean, String etc. enthalten sind, die Bestandteile jeder Software-Entwicklung sind, wird darauf nicht weiter eingegangen.

Die Domain-Schicht

Die Schicht Domain ist nach dem objektorientierten Ansatz implementiert und braucht keine Persistenzschicht. Die Domain-Schicht bietet über eine Schnittstelle einige Dienste für die Presentation-Schicht an (z.B. `Node, CentralNode, SubNode`). Einige Operationen sind z. B. `setTitle(:String)` und `getTitle():String`, die den Titel bearbeiten bzw. darstellen. Mehrere Operationen und deren Kontext, die von der Presentation-Schicht aus der Schicht Domain verwendet werden, kann man aus der Tabelle 6.1 entnehmen.

Mind Map Baumstruktur

Ein Mind Map Baum besteht immer aus genau einem zentralen Knoten - auch Wurzelknoten genannt - das mehrere Unterknoten besitzen kann. Die Unterknoten können entweder den Zentralknoten oder einen anderen Unterknoten als Oberknoten referenzieren; es gibt für alle Unterknoten exakt ein Oberknoten. Alle Gemeinsamkeiten sind in der Klasse `Node` implementiert, die aufgrund dessen, dass nur ein Zentralknoten und Unterknoten erzeugt werden kann und nicht ein Knoten an sich, als abstrakt definiert ist. Optional kann ein Knoten einen Titel, Links und Notizen enthalten. Für den Rahmen hat man einen blaugefüllten Kreis, ein gelbes Rechteck und eine Unterlinie zur Auswahl. Standard ist der blau ausgefüllte Kreis. Über das Popup-Menü kann man andere Rahmen auswählen.

Die Unterknoten werden je nach Menge der linken bzw. der rechten Seite zugeordnet, d. h. wenn die Anzahl der rechten Unterknoten des entsprechenden Oberknotens kleiner oder gleich der Anzahl auf der linken Seite sind, wird der neue Knoten rechts hinzugefügt. Wenn dies nicht der Fall ist, wird der neue Unterknoten links angehängt. Solange Knoten ergänzt werden, bleiben beide Seiten gleich oder haben nur einen Knoten mehr. Sobald aber Knoten entfernt werden, kann es durchaus passieren, dass die Differenz größer als ein Knoten ist, weil die Links-Rechts-Verteilung der noch vorhandenen Knoten im Baum auch beim Entfernen von einzelnen Unterknoten erhalten bleibt.

Die GUI-Schicht

Die mit dem GWT erstellte GUI besteht aus mehreren Komponenten.

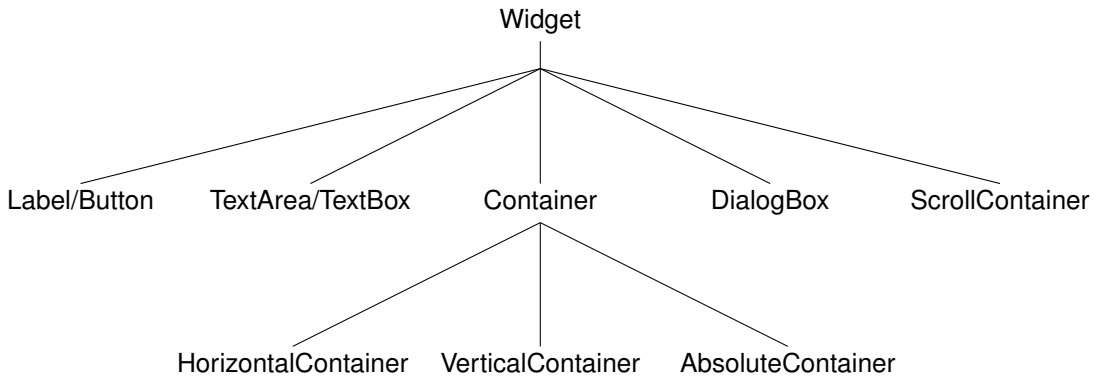

Abbildung 4.5: Einige Komponenten von Widget

In der Darstellung 4.5 kann man gut erkennen, in welcher Beziehung die Komponenten zum Widget stehen.

Die meisten Komponenten sind in der Funktion schon aus dem Bereich GUI in der Programmiersprache Java bekannt. In der Mind Map Anwendung werden die Labels für die Textdarstellung der Knotentitel verwendet. Die Buttons u.a. für die Bestätigung oder das Abbrechen einer Aktion oder für die Darstellung des Notiz-, Link- und Expansion-Icons. Im TextArea bzw. in der TextBox können dann Notizen oder Hyperlinks eingegeben werden. Die Container beinhalten einen Teilbaum an Knoten, die die optische Darstellung bestimmen und anordnen. DialogBox s. TextArea. Die gesamte Mind Map Baumabbildung ist in einem ScrollContainer eingebunden, dass das Navigieren über die Scrollbalken erlaubt. Die genaue Bedeutung, Verwendung und Beziehung der einzelnen Komponenten untereinander wird in den Kapiteln „Anforderungen“ und „Systementwurf“ der Diplomarbeit von Virginio Carfagno gegeben.

Die Presentation-Schicht

In diesem Abschnitt wird das Thema „Presentation“ angesprochen. Dabei werden einige Pakete aus dieser Schicht erläutert. Weiterhin wird im Unterabschnitt Kapitel 4.1.4 kurz auf das in der Analyse angedeutete Problem des „automatischen Layouts“ eingegangen.

Pakete

Für detailierte Erklärungen liest man in der Diplomarbeit Carfagno (2007) im Kapitel „Systementwurf“ - Kapitel 6.5 nach. Dort wird näher auf die einzelnen Pakete eingegangen, die für

die grafische Oberfläche der Mind Map zuständig sind.

Folgend nur eine kleine Übersicht:

- `noteview` - Notiz-Anzeigedialogbox
- `nodeeditor` - Link- und Notiz-Bearbeitungsdialogbox
- `nodecomponent` - Framework-Unterstützung für das Austauschen des Knotenrahmens
- `nodeframes` - erweitert das Framework des Paket nodecomponent für konkrete Knotenrahmen (Kreis, Rechteck, Unterlinie)
- `nodecore` - erweitert das Framework des Paket nodecomponent für die Realisierung des Knotenkerns

Ein vollwertiger Knoten wird im Paket `node` realisiert, indem die Pakete `nodeframe` und `nodecore` den äußeren und inneren Teil eines Knotens umsetzen. Für die Anordnung der Knoten ist das Paket `mindmap` verantwortlich.

Anforderung: Automatisches Layout

In den Anforderungen aus Kapitel 3.4.5 wurde die Problemsituation geschildert, wenn sich die Struktur eines Baums durch den Eingriff einer zweiten Person verändert und somit zum Verlust der Übersicht auf einem bestimmten Ausschnitt hervorruft.

Eine mögliche Lösung auf der visuellen Ebene ist, die jeweiligen Ausschnitte der Gruppenmitglieder auf der Gesamtsicht der Powerwall mit einer Markierung darzustellen. Von jedem Teilnehmer werden die jeweiligen Knoten des Ausschnitts z.B. farblich auf der Powerwall markiert, sodass man seinen eigenen Arbeitsbereich auf der Powerwall zurückverfolgen kann. Wenn sich die Zweige nun verschieben, muss der jeweilige Betroffene nicht lange in einem komplexen Baum nach seinem Zweig suchen, sondern orientiert sich aufgrund der Markierung in kürzester Zeit auf der Powerwall.

Die Abbildung 4.6 steht für die Sicht der Powerwall auf die Mind Map. Dort sind zwei schlichte, farbliche Markierung zu sehen; zugeschnitten auf die jeweilige Sicht des Notebooks, die die Bereiche der Editierenden darstellen.

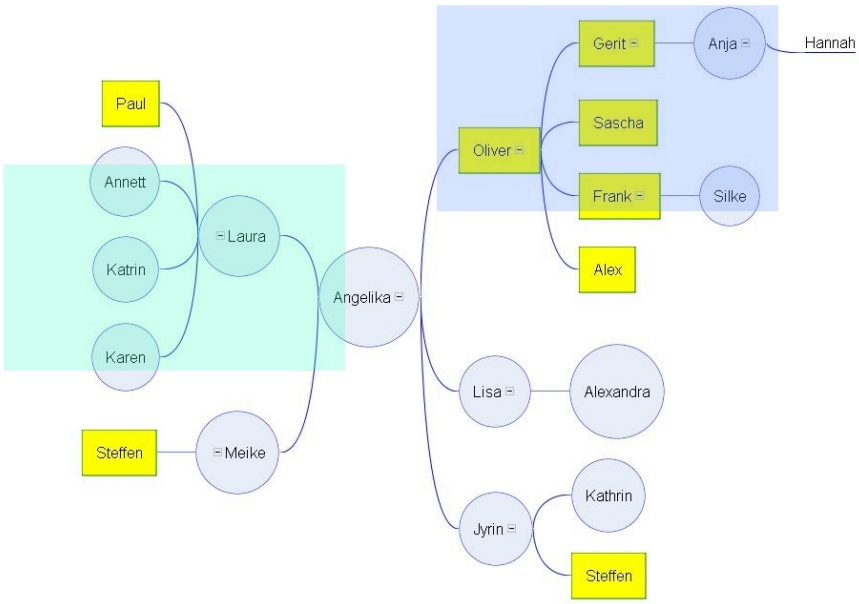

Abbildung 4.6: Markierungsbeispiel auf der Sicht der Powerwall

In dieser Bachelorarbeit wird von der idealen Situation aus Kapitel 3.5 ausgegangen, an dem die mündliche Absprache einer schreibberechtigten Person erfolgreich durchgeführt wird. Somit wird das von mehreren Editierenden ausgelöste Problem umgangen.

4.1.5 Kopplung der Schichten Presentation und Domain

Wie aus der Diplomarbeit von Carfagno (2007) hervorgeht, sind die Richtlinien für Paketstrukturen folgende:

- möglichst hoher Zusammenhalt eines Pakets
- möglichst niedrige Kopplung zwischen Paketen
- keine zyklischen Abhängigkeiten von Paketen
- unabhängig funktionierende Typen sollten in verschiedenen Paketen angesiedelt sein

Das Zusammenspiel zwischen den Schichten Presentation und Domain, also die Kopplung, wird in diesem Abschnitt betrachtet.

Es gibt in der Presentation-Schicht Klassen, die den Zustand der Domain abfragen oder manipulieren und es gibt Klassen, die den Zustand abfragen und verändern. Im Beispiel

des Mind Maps: Die Klasse `NodeView` verändert den Domain-Zustand und die Klassen `NodeCore`, sowie `Tree` sind abhängig vom Domain-Zustand. `NodeCore` und `Tree` rufen die entsprechende Methode auf, um sich zu aktualisieren. Wenn der Domain-Zustand verändert wird, sollen die Klassen `NodeCore` und `Tree` benachtrichtigt werden, um ihre Zustände zu synchronisieren.

Die Synchronisierung der beiden Schichten wird mithilfe des Entwurfsmusters Beobachter realisiert. In der Domain-Schicht, hier in der `Node`-Klasse, werden die `NodeObserver`-Objekte in einer Liste verwaltet. Die Klassen `Tree`, `NodeCore` und `NodeView` implementieren als Beobachter die Schnittstelle `NodeObserver` und registrieren sich bei `Node`, um über Zustandsänderungen von `Node` benachrichtigt zu werden. Sobald in `Node` eine Zustandsveränderung eintritt, wird die Methode `fireNodeChanged()` ausgelöst. Diese Methode schickt allen registrierten Beobachter-Klassen - den `NodeObservern` - die Nachricht `nodeChanged()`. Erhält der `NodeObserver` diese Nachricht `nodeChanged()`, fragt er `Node` nach dem aktuellen Zustandswert und aktualisiert sich bzw. in diesem Fall die Darstellung dementsprechend. Weitere Informationen dazu sind in der Sekundärliteratur von Carfagno (2007) enthalten.

4.1.6 Model-View-Controller

Das Konzept des Model-View-Controller (MVC) wurde erstmals 1979 von Trygve Reenskaug in Smalltalk für die Benutzeroberfläche beschrieben. Die originale Implementierung kann man im Detail in der Publikation „Applications Programming in Smalltalk 80 - How to use Model-View-Controller“ nachlesen.

Model-View-Controller (MVC) ist ein Design-Pattern, das zum Ziel hat, die Präsentation einer Anwendung von seiner Logik und den Daten zu trennen. Vor allem die Wartbarkeit soll auf diese Weise verbessert werden. Außerdem werden Projekte strukturierter und flexibler, da die Logik und die grafische Benutzerschnittstelle einfacher austauschbar sind.

MVC in „klassischen“ Anwendungen
Das MVC-Entwurfsmuster organisiert ein System in drei Komponenten (siehe Abbildung 4.7 links):

- Die ***Model-Komponente*** verwaltet die Daten der Anwendung und liefert diese größtenteils an die View-Komponente. Ferner nimmt sie aufgrund von Instruktionen, die von der Controller-Komponente gesendet werden, Änderungen an den Anwendungsdaten vor.

- Die ***View-Komponente*** ist für die grafische Präsentation der Anwendungsdaten zuständig, die von der Model-Komponente verwaltet werden.
- Die ***Controller-Komponente*** interpretiert Interaktionen des Anwenders, die mit Hilfe von Eingabegeräten, wie z.B. Maus oder Tastatur, erzeugt werden. Wenn Anwendungsdaten oder deren Darstellung sich aufgrund von Benutzereingaben ändern sollen, informiert der Controller die Model- und View-Komponente.

Üblicherweise spielen eine Control- und eine View-Komponente paarweise zusammen. Sie „kennen“ sich gegenseitig und beziehen sich auf dasselbe Model. Ein Model kann von mehreren solcher View-Control-Paare gesteuert werden. Jedoch kann das Model nicht auf die anderen beiden Komponenten zugreifen. Diese Entkoppelung bringt den Vorteil, dass die Model-Komponente unabhängig von den beiden anderen Komponenten entwickelt und getestet werden kann.

Es gibt nach Angaben von Rossberger (2008) noch eine andere Variante des MVCs, die sich von der bisherigen unterscheidet. Der Observer kommt neu hinzu, wie in Abbildung 4.7 auf der rechten Seite erkennbar ist.

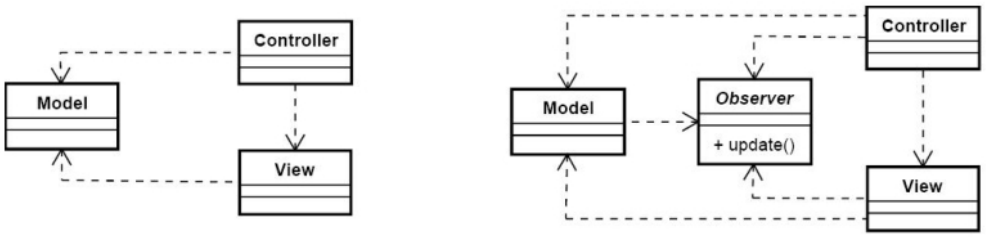

Abbildung 4.7: MVC-Klassenstruktur bei passiver (links) und aktiver (rechts) Model-Komponente mit Observer-Pattern (nach Rossberger (2008) wird Burbeck (1992) zitiert).

Im CCW, in dem auf gemeinsame Dokumente bzw. Datenobjekte zugegriffen wird, können Änderungen am Datenbestand auch extern und ohne Einbeziehung der Controller-Komponente erfolgen. In solch einem Fall muss das Model die Controller- und die View-Komponente über die entsprechende Änderung informieren. Im Vergleich zur passiven Variante des MVCs erfolgt hier die Kommunikation zwischen den Komponenten bidirektional und wird als „MVC mit aktiver Model-Komponente“ bezeichnet.

Wie schon oben erwähnt, ist die Gewährleistung der Unabhängigkeit des Models von den anderen beiden Komponenten eines der Hauptgründe, weshalb das MVC-Muster eingesetzt wird. In der Variante mit der aktiven Model-Komponente können jedoch ungewollte Abhängigkeiten entstehen, wenn die Benachrichtigung von View- und Controller-Komponente direkt durch die Model-Komponente erfolgen würde. Um dies zu vermeiden, empfielt sich der Einsatz des Observer-Patterns[3] [Gamma u. a. (1995) und Rossberger (2008)).

Beim Observer-Pattern implementieren alle Komponenten, die über Zustandsänderungen am Datenbestand der Anwendung informiert werden wollen, das abstrakte Observer-Interface und registrieren sich bei der Model-Komponente. Sobald sich was am Datenbestand ändert, informiert die Model-Komponente alle registrierten Komponenten, indem sie über die Liste der registrierten Observer iteriert. Das ist vorteilhaft, da die Model-Komponente keine Kenntnis über die registrierten Komponenten oder deren Logik verfügen muss. Dank dem Einsatz des Observer-Patterns bleibt die Unabhängigkeit zwischen dem Model und der anderen beiden Komponenten erhalten.

4.1.7 Model-View-Controller im Mind Map System

Wie schon im vorangegangenen Abschnitt beschrieben, eignet sich für Anwendungen, bei denen Änderungen am Datenbestand durch externe Anwendungen erfolgen, das MVC-Muster mit aktiver Model-Komponente am Besten. Die Architektur der Mind Map Anwendung basiert auf dieser MVC-Variante.

Nach der MVC-Funktionsverteilung lassen sich die Bestandteile der Mind Map Anwendung folgendermaßen aufgliedern:
Model: die Verwaltung aller Informationen des Mind Maps (=Anwendungsdaten), z.B. Knotenobjekte.
View: die grafische Darstellung der aktuellen Mind Map mit den Knoten und deren Verzweigungen.
Control: Der Benutzer kann mit Hilfe verschiedener Eingabegeräte, wie Maus und Tastatur, die Mind Map manipulieren. Interpretation der Benutzereingaben und deren Umsetzung (Änderung) auf das Model.

In der Abbildung 4.8 wird der Aufbau der Komponenten dargestellt. Die public View aktualisiert sich mit Hilfe des Observer-Patterns mit den neuen Informationen aus der Logik. Auf

[3] zu dt. Beobachter-Muster

der Applikation-Ebene befindet sich das Model und die Logik der Mind Map Anwendung - auch als Mind Map Server in Abbildung 4.9 beschrieben.

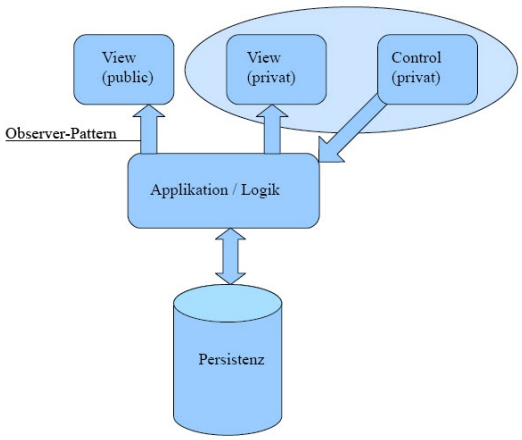

Abbildung 4.8: Design: Komponenten

Die public View-Komponente steht in unserem Szenario für die Powerwall, die die Mind Map für alle Teilnehmer darstellt. Das Notebook, das in der Grafik 4.8 oben rechts als private View- und Control-Komponente dargestellt wird, erhält die View-Daten von der Logik-Komponente, also dem Mind Map Server zur Darstellung des Mind Map Baums. Die private Control-Komponente kommuniziert alle Interaktionen des Benutzers an die Applikationslogik bzw. dem Mind Map Server.

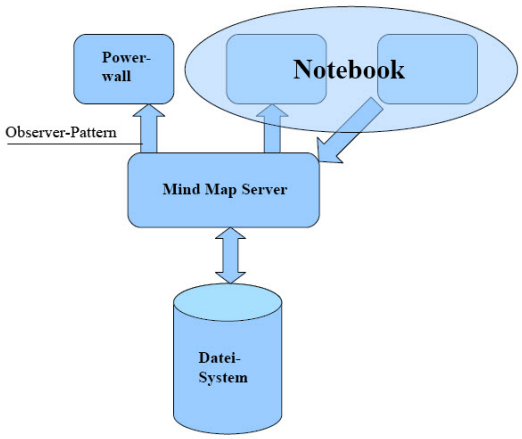

Abbildung 4.9: Design: Hard- und Software

4.2 Zusammenfassung des Entwurfs

Zusammengefasst sieht das Design folgendermaßen aus:

- Client/Server-Modell: per RPC-Protokoll erhält der Client alle neuen Daten vom Server übermittelt.
- RIA: browserbasierte Client-Anwendung. Client enthält Benutzerschnittstelle und einen Teil der Logik. Der andere Teil der Logik und die „Database“ liegen auf der Serverseite. Der Thin-Client holt sich bei Bedarf die nötigen Daten vom Server.
- Framework: ajaxbasiert, GWT (Google Web Toolkit). Schwerpunkt der Logik liegt auf dem Client, RPC auf dem Server.
- Schichtenarchitektur: Domänenschicht und Präsentationsschicht.
- Entwurfsmuster: Model-View-Controller, Observer-Pattern (Beobachter-Entwurfsmuster) zur Synchronisierung der Präsentations- und Domänenschicht, Dekorierer und Schablonenmethode wurden verwendet.

4.3 Implementierung

Wie man in der Diplomarbeit von Carfagno (2007) im Kapitel 7 nachlesen kann, ist die Mind Map Anwendung mit dem GWT auf einer Schichtenarchitektur aufgebaut. Es gibt die Schichten *Domain*, *Gui*, *Presentation* und *Foundation*, wie auf der Abbildung 4.10 zu sehen ist.

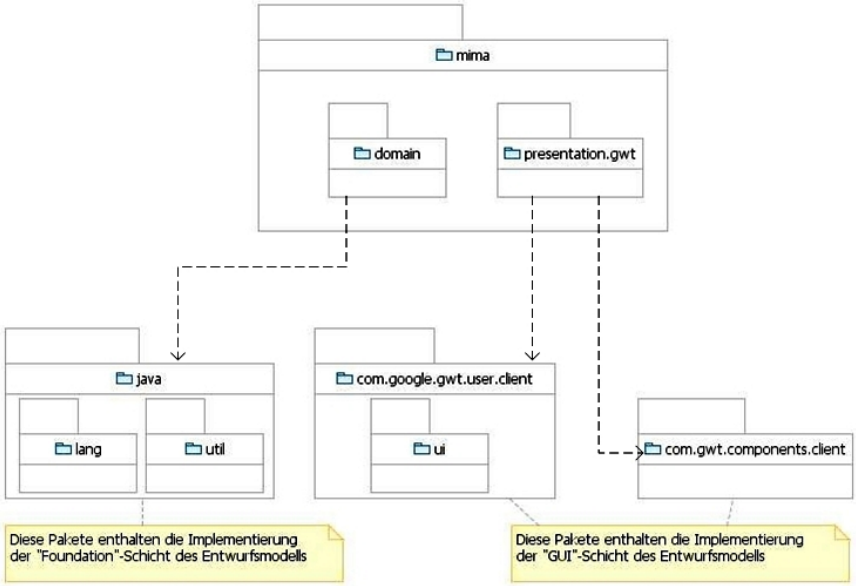

Abbildung 4.10: Implementierung der Schichtenarchitektur

Virginio Carfagno hat auch schon zu dem Zeitpunkt mit der Entwicklungsumgebung Eclipse gearbeitet. Seine Systemversion bestand aus Java 1.4 und GWT 1.2.22. Auf diesen Versionen wurde hier weiterentwickelt. Die Funktionalität der Mind Map Anwendung wurde mit der Firefox Version 2.0.0.20 erfolgreich getestet.

Asynchrone Kommunikation: Remote Procedure Calls (RPC)

Ursprünglich war die Mind Map nur eine Einzelplatzanwendung. Daher musste für diese Bachelorarbeit eine asynchrone Kommunikation mit einem Server hergestellt werden, um die Mehrbenutzerfähigkeit zu realisieren. In dieser Ausarbeitung musste zusätzlich noch zur Abgleichung der Dateninformationen serverseitig der Datentransfer-Service programmiert werden, um die Kommunikation zwischen den einzelnen Clients zu realisieren. Dies wurde mit Hilfe des RPC der GWT-Bibliothek realisiert. Im Folgenden wird in den Unterkapiteln näher auf die client- und serverseitige Entwicklung eingegangen.

4.3.1 Service für den Datentransfer

Zu der Mind Map Anwendung von Virginio Carfagno mussten für die aynchrone Kommunikation drei weitere Klassen im GWT implementiert werden, die für den Servicedienst zuständig

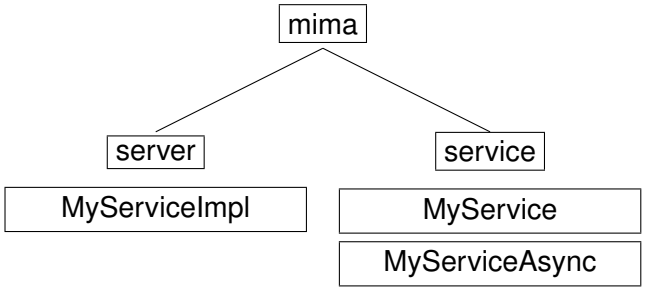

Abbildung 4.11: Service-Transfer-Klassen

sind.

- `service.MyService`
- `service.MyServiceAsync`
- `server.MyServiceImpl`

Abbildung 4.11 stellt die Struktur grafisch dar.

`MyService`, ein Interface, in dem Service-Methoden deklariert sind. Diese Klasse ist serverseitig implementierter Code.

`MyServiceAsync`, auch eine Interface-Klasse, die für den clientseitigen Code verwendet wird. Der Name muss derselbe sein, wie das serverseitige Interface, mit dem Anhang „Async“. Alle Methoden aus der serverseitigen Klasse müssen auch hier implementiert werden. Wichtig ist, dass das `AsyncCallback`-Objekt als zusätzliche Parameter zu jeder Methode ergänzt wird. Die beiden Service-Klassen sind identisch, bis auf den Parameter des `AsyncCallback`-Objekts und dem Rückgabewert, der im letzteren Fall durch das void ersetzt wird.

`MyServiceImpl`, eine Service-Klasse, die serverseitig implementiert wurde. Dort werden die eigentlichen Verhalten und Eigenschaften programmiert, während in den anderen beiden Service-Klassen nur die Methoden-Signaturen vorhanden sind.

`MiMa` besitzt nur die Methode `onModuleLoad()`, die alle relevanten Objekte und Methoden für die Darstellung der Mind Map und die Kommunikation zwischen Server und Clients erzeugt bzw. aufruft. Für die Kommunikation sind die Service-Klassen notwendig. Der Verbindungsaufbau zwischen Server und Client wurde zu der Klasse `CallService` herausfaktorisiert, um Code-Redundanzen zu vermeiden. Verwendung findet die `CallService`-Klasse bei der Prüfung nach geänderten Dateninhalten und zum Aktualisieren des Mind Map.

Listing 4.1: Code-Ausschnitt aus MiMa.onModuleLoad()

```
1 CallService.getService().getCentralNode(new AsyncCallback(){
2         public void onFailure(Throwable caught) {
3                 Window.alert("failure im AsyncCallback - 
                      MiMa.getService().getCentralNode():" + caught.getMessage());
4         }
5         public void onSuccess(Object result) {
6                 CentralNode resultCNode = (CentralNode) result;
7                 mindMap.setCentralNode(resultCNode);
8         }
9 });
```

Clientseitige Entwicklung

`Frame` enthält das Handling der Canvas-Formen. Es ist eine abstrakte Klasse mit statischen final-Variablen (`CIRCLE, RECTANGLE, UNDERLINE, NONE`).

`MindMapInfo` ist eine Hilfsklasse, die sich u.a. um den Zoomfaktor und Movefaktor kümmert. Dort wird der Zoomfaktor inkrementiert und die Coordinaten über den Movefaktor neu gesetzt. Der Zoomfaktor ist für die optische Vergrößerung zuständig, der Movefaktor für die Navigation der Sicht auf den Mind Map Baum.

`CallService` wird verwendet, um einen RPC-Aufruf vom Client auszulösen.

- Mithilfe der Methode `GWT.create(MyService.class)` wird das Service-Interface instanziiert.
- Die Start-URL für den Service-Proxy wird per `ServiceDefTarget` spezifiziert.
- Ein asynchrones `Callback`-Objekt wird erzeugt und anschließend der Aufruf ausgeführt.

Listing 4.2: Code-Ausschnitt der Klasse CallService

```
10 MyServiceAsync myService = (MyServiceAsync) GWT.create(MyService.class);
11 ServiceDefTarget endpoint = (ServiceDefTarget) myService;
12 String moduleRelativeURL = GWT.getModuleBaseURL().equals("") ? "/" :
       GWT.getModuleBaseURL();
13 endpoint.setServiceEntryPoint(moduleRelativeURL + "myService");
```

Diese Klasse besitzt eine Methode - `getService()` - mit dem Objekttyp `MyServiceAsync` als Rückgabe. Dieses Rückgabe-Objekt wird in der Methode `onModuleLoad()` der `MiMa`-Klasse verwendet und ruft damit die Service-Methoden `getStatusCounter()` und `getCentralNode()` auf. Weiterhin wird in der Klasse

`NodeView` bei den Command-Methoden, also einer Änderung des Baumes die Service-Methode über das Rückgabe-Objekt von `CallService` der `StatusCounter` inkrementiert. Der `StatusCounter` wird in `MiMa.onModuleLoad()` in sehr kurzen Zeitabständen abgefragt. Somit wird immer nur ein Flag geprüft und erst bei einem neuen Wert des `StatusCounters` wird auch der Baum aktualisiert. Die neuen Daten des Baums werden daraufhin an die Clients verschickt.

`Coordinates` ist eine einfache Bean-Klasse, die Getter- und Setter-Methoden für die Koordinaten implementiert.

Folgende Klassen von Virginio Carfagno's Fassung mussten noch um einige neue Implementationen ergänzt werden:

`Node`

- Die Klasse implementiert das `IsSerializable`-Interface, da sie über den Server geschickt werden soll. Das schließt einen public Standard-Konstruktor (ohne Parameter) mit ein.
- Die innere Klasse `Frame` wurde extrahiert in eine eigene Klasse im `domain`-Package.
- Der alte Typ `Frame` der Variable `_frame` wird vom primitiven Typ `int` ersetzt.
- Die `_nodeObservers` Collection wird mit dem Modifikator `transient` belegt, da es nicht serialisiert werden darf.
- Das Erzeugen eines neuen HashSet `_nodeObservers` wird aus dem Konstruktur extrahiert und in eine eigene Methode `initObservers()` implementiert. Somit kann von einer anderen Stelle erneut die Collection mit einem neuen HashSet definiert werden. Das ist zum „Wiederaufbau“ nötig, da es durch das `transient` „verloren“ geht.
- Wenn ein Observer registriert wird `registerObserver(param)`, wird zusätzlich geprüft, ob die Collection `_nodeObservers` `null` ist oder nicht. Falls ja, dann wird `initObservers()` aufgerufen.
- Im Falle einer Änderung des Mind Map Baumes, die vom Benutzer ausgelöst wird, wird die Methode `fireChangeToAllSubNodes()` aufgerufen, die iterativ und rekursiv alle vorhandenen Subknoten durchläuft und bei jedem Subknoten das `fireNodeChanged()` auslöst. Dort werden alle Beobachter über eine Änderung der Daten informiert.

`CentralNode`

- Die Klasse implementiertn das Interface `IsSerializable`[4].
- Der Typ `Frame` der Variablen `DEFAULT_FRAME` wird hier vom Typ `int` ersetzt.
- Ein public Standard-Konstruktor ohne Parameter.

`SubNode`

- Die Klasse implementiertn das Interface `IsSerializable`.
- Der Typ `Frame` der Variablen `DEFAULT_FRAME` wird hier vom Typ `int` ersetzt.
- Ein public Standard-Konstruktor ohne Parameter.

4.4 Fazit

Folgendes konnte partiell umgesetzt werden:

1. Anzeige der Mind Map auf der Powerwall (im View-Modus)
2. Bereitstellen der Mind Map im View-Modus, z.B. auf einem Notebook
3. Bereitstellen der Mind Map im Edit-Modus auf einem Notebook
4. Abgabe der Schreibrechte

Die AJAX-Technologie machte es in dieser Bachelorarbeit möglich, eine browserbasierte Anwendung zu implementieren, deren Clients auf ein gemeinsames Mind Map Dokument zugreifen können. Die Powerwall, sowie mehrere Notebooks können im Lese-Modus das aktuelle Mind Map betrachten. Von einem Notebook aus kann man im Schreib-Modus an dem Mind Map editieren. Änderungen am Mind Map werden über die zentrale Serverinstanz synchronisiert. Den beobachtenden Clients werden bei Bedarf der neue Code übermittelt. Damit wurden die ersten drei Punkte erfüllt. Die Abgabe der Schreibrechte wurde durch zwei verschiedene Modi (Lesen/Editieren) realisiert. Durch mündliche Absprache wird festgelegt, welche Person den Edit-Modus verwenden darf. Alle anderen Personen verwenden den Lese-Modus.

[4] in der Version GWT 1.4 ist dieses Interface notwendig. Diese ist nicht vererbbar. Neuere GWT-Versionen implementieren das Interface Serializable, die vererbt werden können

Im CSCW ist der Gesichtspunkt der Reaktionszeit von Interesse. Wenn mehrere Personen zur selben Zeit am selben Ort eine Diskussion auf Basis eines gemeinsamen Dokumentes führen, sind lange Wartezeiten für den Gesprächsfluss/Diskussionsverlauf nicht von Vorteil. In der `MiMa.onModuleLoad()` ist es möglich, das Zeitfenster für die Abfrage nach Änderungen selbst zu bestimmen. Bis zu einer bestimmten Größenordnung der Mind Map kann die Frage der Reaktionszeit von weniger als einer Sekunde positiv beantwortet werden. Da der gesamte Baum geschickt wird, kann es bei einer komplexen Struktur zu leichten Verzögerungen kommen.

Bei der Kompetenzverteilung von Client und Server im Zusammenhang mit dem CSCW muss noch untersucht werden, an welcher Stelle der Schnitt innerhalb der Applikationsebene am meisten Sinn macht. Es ist wichtig, den Schnitt der Kompetenzverteilung an der richtigen Stelle zu setzen. Je besser man den Schnitt wählt, desto mehr Vorteile einer RIA kann man daraus ziehen, wie z.B. das Ausmaß des Installationsaufwandes, die flüssige Bedienung und die Reaktionszeit. Für die Weiterentwicklung dieser Software muss sich der Entwickler darüber Gedanken machen.

Weiterhin muss folgendes Browser-Problem berücksichtigen: es gibt in der Mind Map Anwendung einige nicht standartisierte Grafikelemente, die bei einem Browserupdate nicht mehr korrekt ausgeführt werden. Dieses Defizit muss langfristig gelöst werden.

Ob sich das Framework GWT im kollaborativen Kontext mit mehrfachen Views durchsetzt, bleibt noch abzuwarten. Es gibt heutzutage viele vergleichbare AJAX-Toolkits, wie das Eclipse-Projekt Rich AJAX Platform[5]. Vorteilhaft ist, dass die Ajax-Anwendung in der Programmiersprache Java entwickelt werden kann. Weiterhin gibt es noch die anderen bekannten JavaScript-Frameworks, wie Prototype oder Script.aculo.us. Deshalb müsste man in entsprechenden Zeitabständen das vorhandene Angebot an Frameworks, sei es JavaScript-basiert oder Java-basiert, neu evaluieren und die einzelnen Vor- und Nachteile miteinander abwägen.

[5] http://www.eclipse.org/proposals/rap/

5 Zusammenfassung

In dieser Bachelorarbeit wurde mit dem Google Web-Toolkit (GWT) eine browserbasierte Mind Map Anwendung zur Unterstützung von kollaborativen Arbeitssituationen entwickelt.

Im Grundlagenkapitel Kapitel 2 wurde das CSCW beschrieben, kollaboratives Arbeiten an gemeinsamen Dokumenten und ajaxbasierte Anwendungen besprochen. Die Begriffe des CSCW und der RIA wurden in einen gemeinsamen Kontext gebracht. Dabei zeigten sich die Vorteile einer RIA in einem CSCW als nützlich.

Anschließend wurden in Kapitel Kapitel 3 die Interaktionsebenen und Nutzungsbereiche gegenüber einer Powerwall beschrieben. Die Bedingungen an die Benutzer und das System wurden gestellt. Die kollaborative Umgebung wurde analysiert und Anforderungen an die Software erstellt. Weiterhin wurde ein Beispielszenario vorgestellt. Es ergab sich am Ende des Kapitels die Frage, wie mit zeitnahe unterschiedlichen Aktionen auf denselben Dateninhalt umgegangen werden soll.

In Kapitel Kapitel 4 wurde das Szenario als Basis für das entworfene Design verwendet und anschließend ein Prototyp umgesetzt. Dabei wurde auf verschiedene Systemarchitekturen eingegangen. Die Mind Map Anwendung basiert auf dem Client-Server-Modell. Die Anwendung ist größtenteils eine Client-Anwendung, die über dem Webbrowser bedient wird. Bei Bedarf wird Code vom Server angefragt. Die Applikation ist auf einer Schichtenarchitektur aufgebaut, die *Domain* und *Presentation* beinhaltet. Weiterhin wird das Konzept des Model-View-Controllers verwendet. In unserem Fall wird die aktive Variante mit dem Observer-Pattern eingesetzt. Die Kommunikation geschieht über das RPC und wird mit Service-Klassen des GWTs realisiert. Anschließend wurde evaluiert, welche Funktionen erfolgreich implementiert werden konnten und inwieweit sich die Mind Map Anwendung im CSCW als nützlich erweist. Die Frage der Kompetenzverteilung in dieser Mind Map Anwendung wurde am Ende gewertet. Letztlich werden die Anforderungen an die Applikation im Rahmen eines CSCW, wie sie im Beispielszenario (Kapitel 3.5) dargestellt wird, erfüllt.

Da in dieser Bachelorarbeit hauptsächlich untersucht wurde, ob das Mind Map System über den Browser kollaborativ umsetzbar und einsetzbar ist, müssen weitere Aspekte in zukünftigen Weiterentwicklungen beachtet werden. Dazu gehören langfristige Datenpersistenz, Sicherheit des Systems und systemunterstützte Verwaltung von parallel bearbeiteten Dateninhalten, die betrachtet und vervollständigt werden müssen. Diese Arbeit beschäftigte sich mit den angesprochenen Aspekten nur geringfügig.

Das Ziel hier war, im CSCW die Vorteile eines RIA-basierten Systems unter den Bedingungen selber Raum und selbe Zeit optimal zu nutzen. Die Arbeitssituation einer Gruppe wird durch die verteilte Anwendung verbessert. Synchrones Arbeiten an einem gemeinsamen Dokument innerhalb einer Diskussion wird möglich, vorausgesetzt man berücksichtigt die möglichen Konflikte und verwendet entsprechende Konfliktlösungen. Jedes Gruppenmitglied kann ohne Installationsaufwand, nur mit einem Webbrowser ausgestattet, in diese Kollaborationsarbeit einsteigen.

In dieser Ausarbeitung wurde von einer kleinen Diskussionsgruppe von ca. 4 Personen ausgegangen. Die Komplexität eines Mind Maps dieser Größenordnung hält sich meistens in kleinem Rahmen. Somit genügt diese Mind Map Anwendung den hier gestellten Anforderungen und Kriterien. Im Falle von größeren Verhältnissen muss man sich erneut mit dem Thema der Kompetenzverteilung unter CSCW-Bedingungen auseinandersetzen und alle relevanten Fragen, wie z.B. die Reaktionszeit, neu evaluieren.

Inwieweit sich die Unterstützung der Gruppenarbeit in einer kollaborativen Umgebung, dem CSCW, entwickeln kann, muss sich in Zukunft noch zeigen. In den vergangenen Jahren stand die klassische Desktop-Anwendung immer mehr im Wettbewerb zu den browserbasierten Anwendungen. Inzwischen hat man verschiedene Möglichkeiten an gemeinsamen Dokumenten parallel zu arbeiten. Personen können trotz unterschiedlicher Standorten dasselbe Dokument bearbeiten (Google Docs).

Die Mind Map Applikation hat einen Einblick über Möglichkeiten der Optimierung mit Hilfe der AJAX-Technologien verschafft.

Literaturverzeichnis

[Bartnik 2006] BARTNIK, Roman: *Weiterentwicklung einer Technologiebasis für interaktive Gruppenarbeitsräume*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/bartnik.pdf

[Borghoff und Schlichter 1998] BORGHOFF, Uwe M. ; SCHLICHTER, J. H.: *Rechnergestützte Gruppenarbeit - Eine Einführung in verteilte Anwendungen*. Springer, 1998. – ISBN 3-540-62873-8

[Burbeck 1992] BURBECK, Steve: *Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)*. Webseite. 1992. – URL http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html

[Carfagno 2007] CARFAGNO, Virginio: *Evaluation des Google Web Toolkits durch Entwicklung einer ajaxbasierten Mind-Mapping-Anwendung*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/carfagno.zip

[Crane u. a. 2006] CRANE, Dave ; PASCARELLO, Eric ; JAMES, Darren: *Ajax in action - Das Entwicklerbuch für das Web 2.0*. Addison-Wesley, 2006. – ISBN 3-8273-2414-9

[Eckert 2006] ECKERT, Claudia: *IT-Sicherheit. Bd. 4. Auflage*. Oldenbourg Verlag, 2006. – ISBN 3-486-20000-3

[Fischer 2006] FISCHER, Christian: *Multimodale Interaktionen in Collaborative Workspaces*. Hochschule für Angewandte Wissenschaften Hamburg- Institut für Informatik, Master Seminararbeit. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/fischer/abstract.pdf

[Gamma u. a. 1995] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995

[Gamperl 2006] GAMPERL, Johannes: *AJAX - Web 2.0 in der Praxis*. Galileo Computing, 2006. – ISBN 3-89842-764-1

[Garrett 2005] GARRETT, Jesse J.: *Ajax: A New Approach to Web Applications*. www.adaptivepath.com. 2005. – URL http://www.adaptivepath.com/ideas/essays/archives/000385.php

[Goldflam 2008] GOLDFLAM, André: *Evaluation von Flex 2.0 für die Entwicklung kollaborativer Anwendungen*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/goldflam.pdf

[Haase u. a. 2008] HAASE, Oliver ; REISER, Wolfgang ; WÄSCH, Jürgen: *Technologische Grundlagen von Rich Internet Applications*. Hochschule Konstanz Technik, Wirtschaft und Gestaltung. 2008. – URL http://www.forschung.htwg-konstanz.de/inhalte/Service/Archiv/fb_htwg_forum_screen_2008.pdf

[Hartmann 2007] HARTMANN, Leif: *Rich Internet Applications*. Hochschule für Angewandte Wissenschaften Hamburg- Institut für Informatik, Master Seminararbeit. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08/hartmann/bericht.pdf

[Hauswirth und Dustdar 2005] HAUSWIRTH, Manfred ; DUSTDAR, Schahram: *Peer-to-Peer: Grundlagen und Architektur*. Technische Universität Wien. 2005. – URL http://www.infosys.tuwien.ac.at/Staff/sd/papers/DBS-P2P.pdf

[Hollatz 2007] HOLLATZ, Dennis: *Einsetzbarkeit von AJAX-basierten Applikationen für kooperatives Arbeiten*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/hollatz.pdf

[Johanson und Fox 2002] JOHANSON, Brad ; FOX, Armando: *The Event Heap: A Coordination Infrastructure for Interactive Workspaces*. 2002. – URL http://graphics.stanford.edu/papers/eheap3/eheap_wmcsa.pdf

[Kabalkin 2006] KABALKIN, Mykhaylo: *Gemeinsamer Speicher in Collaborative Workspace*. Hochschule für Angewandte Wissenschaften Hamburg- Institut für Informatik, Master Seminararbeit. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/kabalkin/abstract.pdf

[Kabalkin 2008] KABALKIN, Mykhaylo: *Migration einer Rich Client Applikation auf Code on Demand Technologie*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/kabalkin.pdf

[Köckritz 2006] KÖCKRITZ, Oliver: *Geschichte und Konzept von Collaborative Workspaces*. Hochschule für Angewandte Wissenschaften Hamburg- Institut für Informatik, Master Seminararbeit. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/koeckritz/abstract.pdf

[Köckritz 2007] KÖCKRITZ, Oliver: *Verteilter 3D-Desktop mit Remote-Windows für Collaborative Workspaces*. Hochschule für Angewandte Wissenschaften Hamburg- Institut für Informatik, Master Seminararbeit. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-aw/koeckritz/report.pdf

[v. Luck u. a. 2008] LUCK, K. v. ; BOETZER, J. ; RAHIMI, M. ; VOGT, M. ; WENDT, P.: *Gestenbasierte Interaktion mit Hilfe von Multitouch und Motion-tracking*. Hochschule für Angewandte Wissenschaften Hamburg- Institut für Informatik. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/papers/WIWITA2008.pdf

[Mund 2006] MUND, Horst: *Berechtigungsstrukturen in kollaborativen Umgebungen*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/mund.pdf

[Napitupulu 2007] NAPITUPULU, Jan: *Multihead-Display als interaktives Informationssystem*. Hochschule für Angewandte Wissenschaften Hamburg- Institut für Informatik, Master Seminararbeit. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07/napitupulu/report.pdf

[Napitupulu 2008] NAPITUPULU, Jan: *Ein System mit skalierbarer Visualisierung zur Entwicklung kollaborativer Serious Games*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/napitupulu.pdf

[Neumann 2006] NEUMANN, Carola: *Effizienzsteigerung von Diskussionsprozessen in einem neu gestalteten Konferenzraum*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/neumann.pdf

[Revout 2006] REVOUT, Alexandra: *Dokumentenmanagement: Kollaboratives Schreiben*. Hochschule für Angewandte Wissenschaften Hamburg- Institut für Informatik, Master Seminararbeit. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/revout/abstract.pdf

[Revout 2008] REVOUT, Alexandra: *Ein System für das kollaborative Bearbeiten von Dokumenten in mobilen Umgebungen*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/arevout.pdf

[Rossberger 2006] ROSSBERGER, Philipp: *Tabletop-Arbeitsflächen in Collaborative Workspaces*. Hochschule für Angewandte Wissenschaften Hamburg- Institut für Informatik, Master Seminararbeit. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/rossberger/abstract.pdf

[Rossberger 2008] ROSSBERGER, Philipp: *Physikbasierte Interaktion in kollaborativen computergestützten Umgebungen*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/rossberger.pdf

[Schlichter WS 2001/02] SCHLICHTER, Johann H.: *Computergestützte Gruppenarbeit*. Institut für Informatik, Technische Universität München. WS 2001/02. – URL http://www11.informatik.tu-muenchen.de/lehre/lectures/ws2001-02/cscw/extension/latex/cscw_course-student.pdf

[Schwabe u. a. 2001] SCHWABE, Gerhard ; STREITZ, N. ; UNLAND, R.: *CSCWKompendium*. Springer, 2001. – ISBN 3-540-67552-3

[Tanenbaum und van Steen 2003] TANENBAUM, Andrew S. ; STEEN, Maarten van: *Verteilte Systeme*. Pearson Studium, 2003. – ISBN 3-8273-7057-4

[Teufel u. a. 1995] TEUFEL, Stephanie ; SAUTER, C. ; MÜHLHERR, T. ; BAUKNECHT, K.: *Computerunterstützung für die Gruppenarbeit*. Addison-Wesley, 1995. – ISBN 3-89319-878-4

[Wendt 2007] WENDT, Piotr: *Trailblazers - eine Plattform zur Community-gestützten kollaborativen Erweiterung von Navigationskarten*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/wendt.pdf

[Wenz 2007] WENZ, Christian: *JavaScript & AJAX - Das umfassende Handbuch*. Galileo Computing, 2007. – ISBN 3-89842-859-1

[Winograd und Flores 1989] WINOGRAD, Terry ; FLORES, Fernando: *Erkenntnis Maschinen Verstehen*. Rotbuch Verlag, 1989

Glossar

AJAX Asynchronous JavaScript and XML - Konzept der asynchronen Datenübertragung zwischen und Browser, das es ermöglicht, innerhalb einer HTML-Seite eine HTTP-Anfrage durchzuführen, ohne die Seite komplett neu laden zu müssen. Nur gewisse Teile einer HTML-Seite oder auch reine Nutzdaten werden sukzessiv bei Bedarf nachgeladen.

CCW Co-located Collaborative Workspace

CSCW Computer Supported Collaborative Work - dt. computerunterstützte oder rechnergestützte Gruppenarbeit.

GWT Google Web Toolkit - Framework zur Entwicklung von Webanwendungen. Wandelt Java-Code in JavaScript um.

MVC Model View Controller - Architekturmuster zur Strukturierung von Software-Entwicklung in drei Einheiten Datenmodell, Präsentation und Programmsteuerung. Ziel ist es, eine spätere Änderung oder Erweiterung zu erleichtern und eine Wiederverwendbarkeit der einzelnen Komponenten zu ermöglichen.

P2P Peer-to-Peer - Rechner-Rechner-Verbindung. Kommunikation unter Gleichen. Im Gegensatz zum Client-Server-Modell kann bei P2P ein Rechner Client und Server gleichzeitig sein und somit Dienste nutzen und anbieten.

RIA Rich Internet Application - Anwendung, die Internet-Techniken benutzt und eine intuitive Benutzeroberfläche bietet. Möglichkeiten wie z. B. Drag-und-Drop-Fähigkeit oder Bedienbarkeit über Tastenkürzel existieren. Interaktion mit dem Nutzer. Funktionen auf Clientseite können ausgeführt werden, ohne eine Anfrage zum Server starten zu müssen, das zu flüssige Bedienung führt. Keine Installation ist notwendig. Funktioniert im Browser über Java, Flash, HTML, CSS, JavaScript, AJAX.

RPC Remote Procedure Call - Protokoll für verteilte Anwendungen. Damit sind Methoden-Aufrufe von entfernten Rechnern möglich.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 14. Mai 2009	
Ort, Datum	Unterschrift