Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

Carola Neumann

Effizienzsteigerung von Diskussionsprozessen
in einem neu gestalteten Konferenzraum

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Carola Neumann

Effizienzsteigerung von Diskussionsprozessen in einem neu gestalteten Konferenzraum

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Softwaretechnik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Jörg Raasch
Zweitgutachter: Prof. Dr. Kai von Luck

Abgegeben am 29. Juni 2006

Carola Neumann

Thema der Diplomarbeit

Effizienzsteigerung von Diskussionsprozessen in einem neu gestalteten Konferenzraum

Stichworte

Collaborative Workplace, Projektarbeit im Team, benutzerzentrierte Entwicklung/ user centered design, Softwareentwicklungsumgebung, WAM-Ansatz, Benutzungsmodell

Kurzzusammenfassung

Die Fülle an Informationen, denen ein Mensch ausgesetzt ist, wird auch in Zukunft weiter steigen. Insbesondere Teilnehmer von beispielsweise Konferenzen und Meetings müssen nicht nur mit diesen Informationen arbeiten, sondern auch die Kommunikation innerhalb einer Gruppe koordinieren. Daher sind in einem Konferenzraum die Werkzeuge zur Informationsbewältigung besonders wichtig. Durch diesen Sachverhalt motiviert, ist an der HAW Hamburg das Projekt „kora“ zur Gestaltung eines kollaborativen Arbeitsraumes gegründet worden, das mittels diverser Arbeiten von Studierenden umgesetzt wird. Diese Diplomarbeit ist ein Teil des Projekts und beinhaltet eine Anforderungsanalyse für die Software, die in einem solchen diskussionsunterstützenden Arbeitsraum auf den eingebundenen Geräten den Benutzern zur Verfügung stehen wird. Ein besonderes Augenmerk wird hierbei auf die ergonomische Gestaltung der Benutzungsoberfläche geworfen. Aus den Anforderungen wird ein Vorschlag zur Realisierung der Anwendung unterbreitet, der anhand eines Prototyps demonstriert wird.

Carola Neumann

Title of the paper

Improvement of discussions through a computer supported conference room

Keywords

Collaborative Workplace, Project Work, User Centered Design, Application Development System, Tools & Material Approach, Usage Model

Abstract

Also in future, human beings will be exposed to more and more information. Especially members of i. e. conferences and meetings have to work with this information and they have to coordinate the communication within a group. Therefore in a conference room tools for handling the information are extremely important. Inspired by these ideas, the project »kora« has been founded at HAW Hamburg to create a collaborative workplace. It has been realised in the dissertations of several students. The present dissertation is part of the project and contains an analysis demand for the software which will be available to users of the device existing in this workplace to support the discussions. A special aspect has been the ergonomic creation of the graphical user interface. For this demand the dissertation is a proposal to realize the usability which is demonstrated by a prototype.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

1 Einleitung

1.1 Kontext und Motivation

Meetings, Diskussionen, Präsentationen und Projektarbeiten im Team sind ein elementarer Bestandteil der heutigen Arbeitswelt. Jegliche Teamarbeit benötigt Kommunikation, regen Austausch und Verständigung untereinander. Für diese Zwecke gibt es in vielen Einrichtungen Konferenzräume ähnlich dem in Abbildung 1.

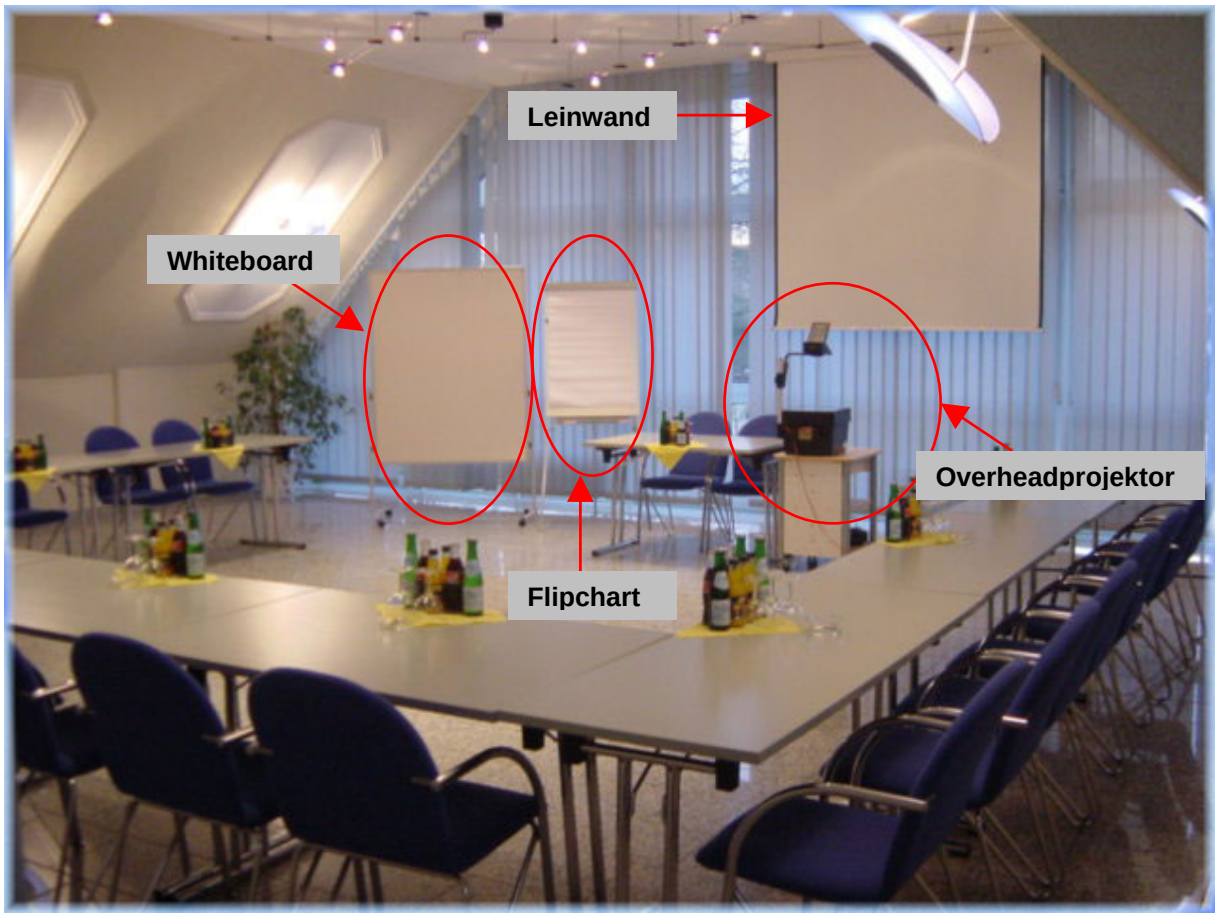

Abbildung 1: Ein typischer Konferenzraum

Die Ausstattung des hier dargestellten Konferenzraumes entspricht nicht dem technischen Stand von heute, ist jedoch noch immer sowohl in Tagungshotels als auch in Firmen präsent. Oft steht anstelle des Overheadprojektors ein Beamer. Dies ist zwar eine Verbesserung, birgt aber kaum weniger Probleme in Hinblick auf effizientes Konferieren. Die Probleme eines in dieser Form ausgestatteten Raumes liegen darin,

dass Dokumente schwer zu bearbeiten, zu verteilen, zu präsentieren und zu archivieren sind. Um diese Problematik zu verdeutlichen, sind folgende Szenarien vorstellbar:

In allen hier beschriebenen Szenarien tagt ein Projektteam bestehend aus den Personen Anne, Bob, Claus, Dora, Eddie und Fred.

Präsentation

Bob möchte auf der Konferenz eine Präsentation durchführen. Hierfür hat er Overheadfolien erstellt. Er legt die erste Folie auf und stellt den Projektor so ein, dass das Bild an der Leinwand scharf ist. Anschließend beginnt er mit dem Vortrag. Dora stellt eine Zwischenfrage, die Bob anhand einer Zeichnung am Flipchart erklärt. Eigentlich hat Bob auf seinem PC ein Dokument, mit dem er die Frage besser beantworten könnte. Dieses kann er hier aber nicht zeigen, da weder ein PC noch ein Beamer im Raum sind. Im Anschluss an die Präsentation findet eine Diskussion über die Präsentation statt. Eddie macht sich während der ganzen Konferenz Notizen und zeichnet die wichtigen Zeichnungen vom Flipchart ab, da er anschließend das Protokoll schreiben soll. Fred möchte gerne eine Kopie der Folien haben. Bob sichert ihm zu, ihm diese zu schicken, sobald er wieder an seinem Arbeitsplatz ist.

Wäre anstelle des Flipcharts ein fest installierter Beamer im Raum, wäre Bob beim Auflegen der digitalen Folien flexibler. Das Vor- und Zurückspringen zwischen den einzelnen Folien ist dann leichter bzw. schneller. Auch hätte er das Dokument, das er Dora gerne gezeigt hätte, präsentieren können.

Weiterhin gestaltet es sich problematisch, die Zeichnungen von den Flipcharts an das Team weiterzugeben. Vorzuziehen wäre ein digitales Board, auf dem die Zeichnung erstellt wird und direkt als Grafik-Datei abgespeichert werden kann. Praktischer wäre es auch, wenn Eddie das Protokoll direkt an einem PC oder Laptop schreibt und es am Ende der Konferenz kurz präsentiert. So kann jeder im Raum Änderungswünsche äußern und Eddie hätte keine weitere Arbeit mit dem Protokoll.

Ein weiterer verbesserungswürdiger Punkt in diesem Szenario ist die Weitergabe der Folien und sonstigen Materialien. Liegen diese in digitaler Form vor, ist die Weitergabe deutlich einfacher. Bob könnte die Datei per Bluetooth an den PDA oder das Mobiltelefon von Fred senden. Noch effektiver wäre es, wenn jeder Teilnehmer seinen eigenen Laptop hätte, diese miteinander vernetzt wären und jeder am Schluss der

Konferenz den kompletten Datensatz auf seinen Laptop gespielt bekäme. Auf diese Weise müsste kein Teilnehmer die Konferenz in Hinblick auf Datenverteilung und Protokollieren nacharbeiten.

Brainstorming

Das Projektteam berät über das Design eines neuen Produkts. Um eine Vielzahl von Ideen zu bekommen, trifft sich das Team zum Brainstorming. Hierzu hat jeder Teilnehmer einen Stapel Karteikarten vor sich liegen. Jede Idee wird auf eine Karte geschrieben. Nachdem alle ihre Ideen notiert haben, werden die Karten mit Nadeln auf Pinnwänden befestigt. In einer anschließenden Diskussion werden einige Ideen aussortiert und andere zu drei Gruppen zusammengefasst. Nun werden die Ideengruppen von jeweils zwei Teilnehmern zu Lösungsansätzen ausgearbeitet. Für die Ausarbeitung der Lösungsansätze nutzen die Teams Papier, Stifte und Flipcharts. Nach einer vorher festgelegten Zeit präsentiert jedes Paar seinen Lösungsansatz mit Hilfe der Flipcharts. Nach den Präsentationen werden die Lösungsansätze diskutiert.
Auch in dieser Situation wäre die Unterstützung durch entsprechende moderne Technik sehr hilfreich. So wäre es denkbar, dass jeder Teilnehmer die Karteikarten an einem PC erstellt. Die Karteikarten werden dann auf ein großes interaktives Display geschoben, auf dem sie danach gruppiert werden. Nach der Zuordnung von Teammitgliedern zu den Karteikartengruppen setzen sich die Paare an jeweils einen PC, um dort ihren Lösungsansatz vorzubereiten. Von dort aus können die Paare wiederum die Präsentation der Ergebnisse auf dem großen Display steuern. Die Ergebnisse werden anschließend zentral gespeichert, so dass das gesamte Team jederzeit darauf zugreifen kann.

Meeting

In der Firma steht die Einführung einer neuen Software an. Um die Mitarbeiter einzuführen, wird ein Berater des Lieferanten eingeladen. Für das Meeting steht dem Team ein Raum zur Verfügung, der mit einem Whiteboard und Flipcharts ausgestattet ist. Der Berater hat einen Laptop dabei, Eddie, Fred und Claus haben Stift und Papier vor sich liegen. Bei diesem Meeting geht es darum, Claus einen Überblick über das Produkt zu verschaffen. Fred und Eddie haben bereits einige Termine mit dem Berater im gleichen Raum gehabt. Der Berater hat eine PowerPoint-Präsentation vorbereitet. Er dreht sein Laptop, damit die anderen die Präsentation

sehen können und berichtet dazu. Anschließend werden Fragen geklärt, zu deren Beantwortung auch die Flipcharts der vorherigen Tage hinzugezogen werden.
In dieser Situation würde sich das Team einen Raum wünschen, in dem für eine Präsentation nicht der Laptop umgedreht werden muss, sondern zumindest ein Beamer vorhanden ist. Zusätzlich wäre es schöner, wenn die Zeichnungen von den Flipcharts auf dem Laptop verfügbar und von allen Teilnehmern veränderbar wären. Auf diese Weise könnte jeder Teilnehmer die Zeichnung direkt verändern und somit den anderen seinen Standpunkt deutlich machen. Sollte die Änderung nicht zutreffend sein, kann diese schnell rückgängig gemacht werden. Auf einem Rechner ist dies sehr einfach, auf den Flipcharts nahezu unmöglich.

Motiviert durch die hier aufgezeigten Problematiken wurde an der Hochschule für angewandte Wissenschaften (HAW) Hamburg das Projekt „kora“ ins Leben gerufen. Kora befasst sich mit der Entwicklung eines Konferenzraumes, der diskussionsunterstützende Techniken in Form von Hard- und Software enthalten wird.
Diese Arbeit befasst sich mit der Entwicklung einer Softwarelösung für die Clients des Konferenzsystems, die die Workflows einer Konferenz unterstützt. Die Clients in dem geplanten Konferenzraum werden von sehr unterschiedlicher Natur sein. Abbildung 2 zeigt einen Raum zur kollaborativen Arbeit mit einer Vielzahl an speziell dafür entworfenen Möbelstücken.

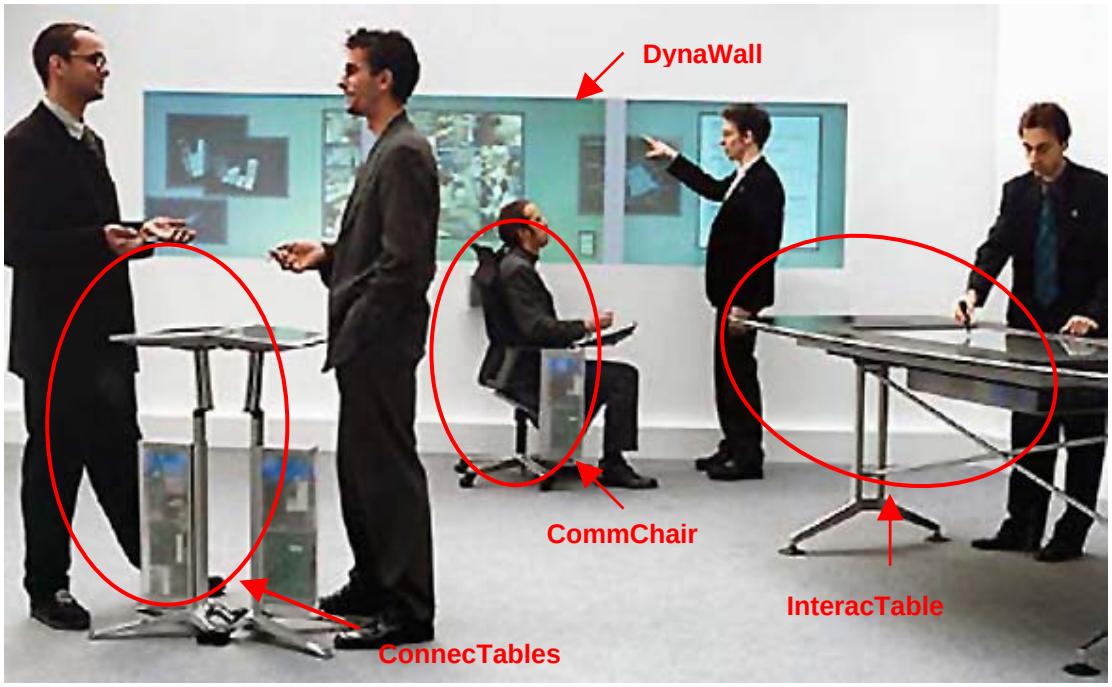

Abbildung 2: Roomware von IPSI Frauenhofer

Ein Client, wie er in dieser Arbeit erarbeitet werden soll, könnte ggf. auf dem „CommChair“ und der „DynaWall“ problemlos laufen. Für die Tische wird der Client wahrscheinlich nicht genug Funktionalität mitbringen, da dort Benutzer von mehreren Seiten aus arbeiten können. Somit müsste der Client drehbar und auf der Arbeitsfläche duplizierbar sein. Eine Analyse dazu soll in diesem Rahmen allerdings nicht weiter durchgeführt werden. Die eben beschriebene Ausstattung stellt lediglich eine Vision dar, wie der Raum nach einigen Entwicklungsstufen aufgebaut sein könnte und was bei der Entwicklung dafür beachtet werden sollte.
Auch soll der Raum später PDAs und Mobiltelefone als Clients akzeptieren. Auf diesen Endgeräten muss demnach die Konferenzsoftware lauffähig sein. In dieser Arbeit wird speziell auf die Gestaltung und die Funktionen der Software auf PCs und Laptops eingegangen. Eventuell könnte der dafür zu entwickelnde Prototyp auch auf einem großen Touchscreen funktionieren – eine Untersuchung dazu ist nicht Thema dieser Arbeit.
Aus den vielen unterschiedlichen Arten von Konferenzen, die im geschäftlichen Alltag stattfinden, werden hier exemplarisch die Konferenzen, die während der Softwareentwicklung vorkommen, betrachtet.
Das Ziel ist eine ergonomische Software, die die Belange des Benutzers stark in den Vordergrund stellt. Für die Entstehung einer solchen Software wird auch ein Entwicklungsprozess benötigt, der stark am Benutzer ausgerichtet ist.

1.2 Zielsetzung

Ein mögliches Vorgehensmodell für die Entstehung benutzerfreundlicher Software beschreibt der WAM-Ansatz (Werkzeug & Material-Ansatz), der am Arbeitsbereich Softwaretechnik der Universität Hamburg entwickelte wurde. Der WAM-Ansatz verwendet klassischer Weise das Leitbild „Arbeitsplatz für eigenverantwortliche Expertentätigkeit“. Bisher manifestierte sich dieser Arbeitsplatz meist am stationären PC, hauptsächlich als Desktopapplikation, aber auch als Webseite im Browser.
Da der WAM-Ansatz anwenderorientiert vorgeht und das Konferenzsystem möglichst intuitiv bedienbar sein sollte, stellt sich die Frage, inwieweit der WAM-Ansatz mit seinen Leitbildern und Metaphern auch für dieses System anwendbar ist und wie er ggf. erweitert oder angepasst werden kann. Es ist zu ergründen, ob das Leitbild des Ar-

beitsplatzes für eigenverantwortliche Expertentätigkeit verwendet werden kann, ein anderes Leitbild bereits existiert oder ein weiteres Leitbild entworfen werden muss. Auch wird ein wesentlicher Bestandteil die Erarbeitung eines Benutzungsmodells[1] sein, da der Anwender sich im Konferenzraum auf die zu bearbeitenden Themen konzentrieren soll und nicht auf die Bedienung der Geräte. Bei einem schwer bedienbaren System würde wertvolle Arbeitszeit verloren gehen und der Konferenzraum wahrscheinlich nur wenig genutzt werden.

Im Rahmen dieser Diplomarbeit wird eine Anforderungsanalyse an das Konferenzsystem erstellt und ein Prototyp entwickelt. Hierbei wird am WAM-Ansatz orientiert vorgegangen. Zusätzlich soll dargestellt werden, welche Anforderungen aus der Sicht der Software-Ergonomie bei der Erstellung der Benutzungsmodelle und Bedienoberflächen der Anwendung zu berücksichtigen sind. Ein besonderes Augenmerk wird darauf geworfen, die Harmonisierung zwischen den unterschiedlichen für die Softwareentwicklung eingesetzten Anwendungen und Artefakten[2] zu verbessern.

1.3 Inhalt, Methodik und Abgrenzung

Zur Erarbeitung der Anforderungsanalyse und eines Prototypen wird zunächst der Ist-Zustand des Anwendungsbereichs analysiert. Um den Ablauf von Softwareprojekten in der Praxis kennen zu lernen und um die Situationen, in denen Meetings stattfinden, zu erarbeiten, wurden zielgerichtete Interviews mit Mitarbeitern der Softwareentwicklungsabteilungen von IBM und Signal Iduna durchgeführt und dort einige Meetings besucht. Aus den dabei gewonnenen Erfahrungen konnten typische Projektsituationen, Szenarien sowie Rollen, die die Mitarbeiter in einem Softwareprojekt verkörpern, festgestellt werden. Hieraus konnten typische Arbeitsabläufe – so genannte Workflows – erarbeitet und Anforderungen an das System abgeleitet werden. In dieser Arbeit wird nach einem zum Konferenzsystem passendem Leitbild gesucht und die Entwicklung eines Benutzungsmodells angestrebt. Auf Basis der vorherigen

[1] Ein Benutzungsmodell ist ein fachlich orientiertes Modell darüber, wie Anwendungssoftware bei der Erledigung der anstehenden Aufgaben im jeweiligen Einsatzkontext benutzt werden kann. Es umfasst eine Vorstellung von der Handhabung und Präsentation der Software aber auch von den fachlichen Gegenständen, Konzepten und Abläufen, die von der Software unterstützt werden (Züllighoven 1998, S. 71).

[2] Ein Artefakt ist ein Element im Komponentendiagramm, das eine physische Informationseinheit darstellt, zum Beispiel ein Modell, eine Quellcode-Datei, eine ausführbare Binärdatei, eine Archivdatei (Balzert 2005, S. 527). Anstelle des Begriffs „Artefakt“ wird in dieser Arbeit auch der Begriff „Dokument“ als Synonym verwendet.

Analyse des Anwendungsbereiches wird das Leitbild festgelegt und die benötigten Werkzeuge und Materialien abgeleitet. Auf dieser Basis wird ein Prototyp entwickelt und ein Benutzungsmodell für diesen geschrieben.
Diese Diplomarbeit stellt nur einen Teil des Projekts zur Entwicklung eines kollaborativen Arbeitsraumes dar und beschäftigt sich hauptsächlich mit dem Konferenzsystem aus Benutzersicht. Eine Marktanalyse, ein Grundkonzept sowie die Erarbeitung der Architektur des Arbeitsraumes sind in der Diplomarbeit von Roman Bartnik (Bartnik 2006) beschrieben.
Auch die Datenverwaltung – also die Persistenzschicht – und die dazugehörigen Überlegungen wie zum Beispiel das Rechtemodell und die Architektur der Persistenzschicht sind in dieser Arbeit nicht enthalten. Informationen über diese Komponente sind in der Diplomarbeit von Horst Mund (Mund 2006) zu finden.
Eine weitere Anforderungsanalyse für ein Konferenzsystem beinhaltet die Diplomarbeit von Lars Burfeindt (Burfeindt 2006), der sich mit Konferenzen und deren typischen Abläufen in Projekten der Stadtplanung befasst und das erarbeitete Konzept von Roman Bartnik weiter verfolgt und in einen Prototyp umsetzt.
Eine Untersuchung in Bezug auf ergonomische Bedienung von großen touchsensitiven Displays wird von Michael Gottwald (Gottwald 2006) durchgeführt. Bei dieser Untersuchung wird zum einen der Fall betrachtet, dass ein oder mehrere der Benutzer direkt vor dem Display stehen, zum anderen aber auch der für den Konferenzraum typischen Fall, dass das große Display von den Clients aus gesteuert bzw. bedient wird.
Die Weiterentwicklung des in dieser Arbeit entstehenden Prototyps wird in der Diplomarbeit von Petra Ehlers (Ehlers 2006) verfolgt.

1.4 Aufbau der Arbeit

Kapitel 2 gibt einen Überblick über den Anwendungsbereich. Hier werden typische Szenarien in der Softwareentwicklung erarbeitet und die Rollen beschrieben, die in einem Softwareprojekt von den Mitarbeitern verkörpert werden können.

Kapitel 3 beschreibt die Weiterentwicklung der in Kapitel 2 erarbeiteten Projektsituationen und Szenarien zu Workflows. Des Weiteren wird sich hier mit der Entwicklung des Prototyps nach dem WAM-Ansatz sowie der Erstellung eines Benutzungsmodells befasst und ein Lösungsansatz entwickelt. Abschließend werden die Anforderungen an das System beschrieben.

Kapitel 4 enthält die Erarbeitung der Werkzeuge und Materialien für die Konferenzanwendung auf Basis der im vorherigen Kapitel herausgestellten Anforderungen sowie die Beschreibung des Benutzungsmodells.

Kapitel 5 schließt die Diplomarbeit mit einer Zusammenfassung, mit der Diskussion der Ergebnisse und mit einem Ausblick auf die mögliche Weiterentwicklung der damit verbundenen Problemstellungen des Prototyps ab.

2 Szenarien für typische Projektabschnitte

Um die Anforderungen, die ein Softwareprojekt an ein Konferenzsystem stellt, herausarbeiten zu können, ist es notwendig, den Ablauf von Softwareprojekten zu durchleuchten und typische regelmäßig wiederkehrende Situationen zu finden. In der Literatur gibt es diverse Ansätze dafür, wie ein Softwareprojekt ablaufen kann. Dies sind so genannte Vorgehensmodelle, die vorschreiben, wie der Softwareentwicklungsprozess laufen soll.
Das erste vorgeschlagene Vorgehensmodell ist das Wasserfallmodell (Boehm 1981, S. 35) – ein Verfahren, das ein Projekt in definierte Phasen unterteilt, die sequentiell abgearbeitet werden (siehe Abbildung 3).

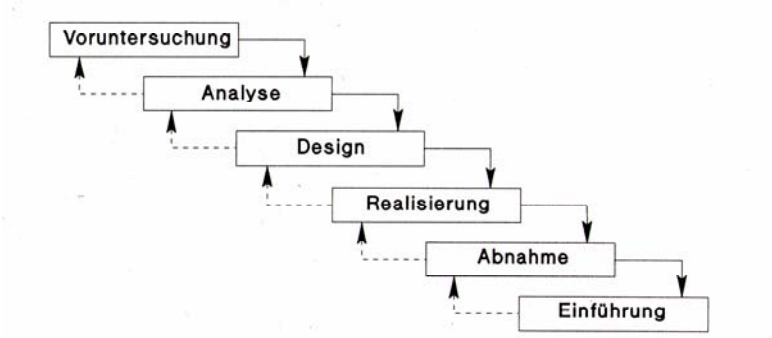

Abbildung 3: Wasserfallmodell

In diesem Verfahren wird die Herstellung von Software als zeitliche Folge von geschlossenen Aktivitäten bzw. Phasen gesehen. Rückgriffe auf vorangegangene Phasen sind zwar erlaubt und in der Praxis die Regel, haben aber Fehlercharakter und sollten somit im optimalen Projektverlauf vermieden werden. Die Testphase steht direkt vor der Auslieferung, wodurch Nutzungsproblematiken und sonstige Schwächen des Systems erst am Projektende auffallen. Dadurch werden Änderungen kompliziert, aufwändig und teuer (Züllighoven 1998, S. 543). Nachdem diese Problematik erkannt worden ist, wurden andere Ansätze zur Softwareentwicklung erdacht. Dazu gehören iterative Verfahren, bei denen die Projektphasen mehrmals durchlaufen werden, um auf geänderte Anforderungen und neue Erkenntnisse eingehen zu können.

Ein Beispiel für iteratives Vorgehen stellt der WAM-Ansatz dar (Züllighoven 1998, S. 539). Er ist selbst kein definiertes Vorgehensmodell, sondern liefert einen Metaprozess für die Softwareentwicklung, welcher eine evolutionäre Vorgehensweise mit Prototyping beinhaltet. Das Vorgehensmodell im WAM-Ansatz muss für jedes Projekt neu erarbeitet werden, weil Softwareprojekte trotz Gemeinsamkeiten unterschiedlich sind. Ein zentraler Punkt des WAM-Ansatzes ist der Autor-Kritiker-Zyklus (siehe Abbildung 4). Die Entwickler sind meist die Autoren. Ihre Arbeitsgegenstände sind die Dokumenttypen des WAM-Ansatzes und Prototypen. Als Kritiker fungieren die potentiellen Nutzer, die das nötige Fachwissen haben. Somit ist der Autor-Kritiker-Zyklus ein permanenter Wechsel zwischen Analysieren, Modellieren und Bewerten und findet in jedem Projektstatus statt (Züllighoven 1998, S. 9).
Da der WAM-Ansatz dazu konzipiert wurde, anwenderfreundliche Software zu produzieren, wird sich die Entwicklung des Prototyps für die Konferenzanwendung am WAM-Ansatz orientieren. Die Konferenzanwendung selber soll später kein Vorgehensmodell unterstützen, sondern dem dort arbeitenden Projektteam die Wahl überlassen, wobei es wünschenswert wäre, wenn nach einem benutzerzentrierten Modell entwickelt werden könnte.

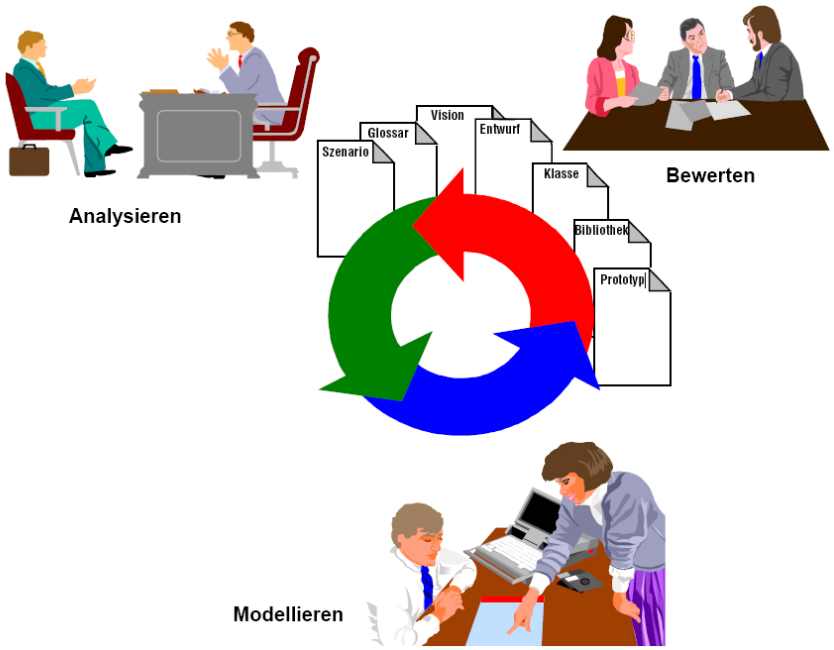

Abbildung 4: Autor-Kritiker-Zyklus

Eine Grundvorrausetzung zur Entwicklung einer Software ist die Einarbeitung in den Anwendungsbereich. Der konkrete Anwendungsbereich ist hier die multimediale Un-

terstützung von Meetings bzw. Konferenzen, die während eines Softwareentwicklungsprojekts stattfinden. Wie bereits erwähnt, wurden Interviews mit potentiellen Nutzern – in diesem Fall Mitarbeiter von IBM und Signal Iduna – des Systems geführt und Meetings bei Signal Iduna beobachtet. Die Vorgehensweise in der Praxis orientiert sich vielfach an der Theorie, wobei das Wasserfallmodell nach wie vor oft zum Einsatz kommt. Das war besonders dem Interview eines Softwareentwicklers bei IBM zu entnehmen. Allerdings spielt die Rückkopplung mit dem späteren Nutzer des Systems zunehmend eine große Rolle, so dass Iterationen in der Entwicklung häufig zu finden sind. Das Konferenzsystem sollte einen anwenderorientierten Ansatz der Softwareentwicklung möglichst intensiv unterstützen und auf diese Weise Softwareprojekte in eine benutzerorientierte Richtung leiten, dem Projektteam aber kein Vorgehen diktieren. Somit sollte u. a. die Rückkopplung zwischen Entwicklerteam und Kunde in jeder Projektphase unterstützt werden.
Bei der Betrachtung von Diskussionsprozessen in Unternehmen – wie in diesem Fall Signal Iduna – ist festzustellen, dass es mehrere Varianten gibt. Einerseits gibt es geplante vorangekündigte oder spontane Meetings in einem Konferenzraum, andererseits können sie aber auch spontan in einem Büro stattfinden, wo Whiteboard, Papier oder sonstige vorhandene Utensilien zu Hilfe genommen werden. Es bleibt hier zu überlegen, wie spontane Meetings in den Büros oder anderen denkbaren Räumlichkeiten unterstützt werden können und ob ggf. eine Anbindung dieser Räume an das Konferenzsystem erfolgen kann.

Folgende typische und in jedem Projekt wesentliche Aufgaben werden exemplarisch untersucht:

- Aufwandsabschätzung/ Angebotserstellung
- Fein-Analyse
- Projektplanung und Entwurf
- Umsetzung und Implementierung
- Auslieferung
- Krisensituationen
    - technische Probleme
    - Missverständnisse und Konflikte im Team und mit dem Anwender

2.1 Rollen

Um die Anforderungen an das Konferenzsystem zu erarbeiten, ist es wichtig, die potentiellen Nutzer – also die Mitarbeiter in einem Softwareprojekt – zu identifizieren und sie den einzelnen Situationen und Szenarien zuzuordnen. In der Literatur, zum Beispiel bei Züllighoven (S. 91 ff.) und Beck (S. 139 ff.), sind dies die so genannten Rollen. Literatur und Praxis machen hier keinen Unterschied. Typischerweise sind folgende Rollen in einem Softwareprojekt zu finden:

Im Systemhaus:

- Vorstand
- Abteilungsleiter
- Projektleiter
- Architekt
- Entwickler

Beim Kunden:

- Vorstand
- Abteilungsleiter
- Organisator
- Benutzer

Beim Lieferanten:

- Vertriebler
- System-Spezialist

Je nach Rolle des Teilnehmers ergeben sich unterschiedliche Anforderungen an die Projektdaten, die dem Nutzer zur Verfügung stehen müssen, da jede Rolle eine andere Position im Projekt verkörpert und somit eine andere Sichtweise auf das Projekt in Hinblick auf Verantwortlichkeiten, Interessen, Aufgaben und Zielsetzungen hat. Die Besetzung bzw. das Vorhandensein der Rollen ist abhängig von der Größe der Firma und der Art des Projekts. So kann zum Beispiel der Besitzer einer kleinen Firma in manchen Fällen mehrere Rollen wie Vorstand und Projektleiter verkörpern. Die Rolle des Abteilungsleiters würde wegfallen oder gewissermaßen auch vom Firmeneigentümer übernommen werden, da ein kleines Unternehmen oft nur eine Abteilung hat bzw. die Abteilungen nur aus jeweils einer Person bestehen.

2.1.1 Rolle Vorstand

Der Vorstand besteht aus mehreren Personen und verwaltet bzw. leitet das Unternehmen. Je nach Größe und Art des Unternehmens sind im Vorstand auch die Besitzer des Unternehmens zu finden. Der Vorstand gibt ein Unternehmensziel vor, das die Mitarbeiter umsetzen sollen, trifft strategische Entscheidungen, macht die Finanzplanung und ist für die Repräsentation zuständig.

2.1.2 Rolle Abteilungsleiter

Beim Abteilungsleiter liegen die Verantwortlichkeiten über seine Abteilung und die Projekte, die dort laufen. Seine Mitarbeiter sind diejenigen, die Verträge aushandeln, das Budget planen und kontrollieren. Er trägt die Verantwortung für die Handlungen seiner Mitarbeiter. Am Tagesgeschäft hat er nur dann Anteil, wenn Ziele nicht erreicht werden und Mitarbeiter Unterstützung benötigen. Zusätzlich ist er für die Personalführung und -entwicklung in seiner Abteilung zuständig. Er stellt den zentralen Ansprechpartner für interne und externe Gesprächspartner dar, verfolgt die Projekte unter Berücksichtigung des vereinbarten Budgets und der Ressourcen gemäß interner Prozessvorgaben und sucht, findet und realisiert Kostenreduzierungspotenziale.

2.1.3 Rolle Organisator

Der Organisator ist eine mit den Organisationsaufgaben beauftragte Person. Je nach Größe des Unternehmens ist er für einzelne Bereiche oder das ganze Unternehmen zuständig. Er führt gemeinsam mit den Fachabteilungen Überprüfungen von Arbeitsabläufen und der Organisation durch, analysiert, beschreibt und bewertet bestehende Organisationskonzepte in betrieblichen Aufgabenbereichen. Er ermittelt die Ist-Zustände im Hinblick auf Datenhaltung und Informationsfluss, auf Prozesse, Abläufe, Strukturen, Schwachstellen und Störungen. Ausgehend von der Ist-Analyse konzipiert er organisatorisch relevante Sollvorstellungen zur Optimierung wirtschaftlicher Abläufe unter Berücksichtigung der Belange der Betroffenen. Er ermittelt die Anforderungen der Nutzer, plant Veränderungen in den Arbeitsabläufen in Zusammenarbeit mit den Nutzern und wählt die zweckmäßigsten Verfahren für den Anwendungskontext aus. Er erstellt in Zusammenarbeit mit den Fachabteilungen neue Arbeitsabläufe, verhandelt ggf. mit Nutzern und wirbt um Verständnis für geplante Veränderungen. Er begleitet und betreut Organisations-Entwicklungs-Projekte bis hin zur Umsetzung, einschließlich der Einweisung in neue Organisationsstrukturen, -mittel und -abläufe und der Schulung der betroffenen Mitarbeiter. Dabei achtet er auf positive oder negative Auswirkungen der Veränderungen auf die betrieblichen Abläufe (BfA 2006, EDV-Organisator).

2.1.4 Rolle Projektleiter

Der Projektleiter ist für das Projekt verantwortlich. Innerhalb der von der Geschäftsleitung festgelegten Vorgaben bearbeitet er selbstständig Projekte von der Konzeption

bis zur technischen und organisatorischen Ausführung. Er definiert die einzelnen Projektschritte, führt und motiviert die Mitarbeiter des Projektteams, weist ihnen Aufgaben zu, koordiniert und kontrolliert die Arbeiten. Daneben verwaltet er das Budget seines Projekts. Er sorgt für die Einhaltung von Terminen und Deadlines und koordiniert die Arbeit seines Teams mit der anderer Teams oder mit vor- bzw. nachgelagerten Abteilungen. Er muss sicherstellen, dass die Qualität seines Produkts stimmt, denn oberstes Gebot ist immer die Kundenzufriedenheit. Darüber hinaus sind einschlägige Rechtsvorschriften zu beachten sowie technische und betriebswirtschaftliche Zusammenhänge zu berücksichtigen. Projektleiter führen ggf. auch Verhandlungen mit staatlichen Stellen oder Aufsichtsbehörden (BfA 2006, Projektleiter).

2.1.5 Rolle Architekt

Zu den Aufgaben des Softwarearchitekten gehört die Konzeption, Entwicklung und Dokumentation von Systemarchitekturen. Er bildet die Schnittstelle zwischen Analyse, Entwurf, Implementierung, Management und Betrieb von Software (Starke 2002, S, 14). Er hat eine verantwortungsvolle Schlüsselrolle, da er für die Erfüllung der Anforderungen (funktional und nichtfunktional) verantwortlich ist. Bei ihm liegt die Funktion des Vermittlers zwischen dem Kunden und dem Entwicklerteam. Somit gehört es zu seinen Aufgaben, stabile Grundgerüste bereitzustellen, die eine Umsetzung neuer und geänderter Anforderungen zulassen (Starke 2002, S, 26 ff).

2.1.6 Rolle Entwickler

Die Entwickler programmieren das System und sind somit für die technische Umsetzung der Projektanforderungen zuständig. Je nach Komplexität wird das System in mehrere Module unterteilt, an denen jeweils einer oder mehrere Entwickler arbeiten. Die Entwickler erstellen das Werkzeug und die Materialien, mit denen die Benutzer später arbeiten sollen. Wichtig während der Entwicklung ist eine fortlaufende Dokumentation der einzelnen Module, die Dokumentation von Besonderheiten sowie Tests mit dem Nutzer. Eine weitere Aufgabe der Entwickler kann es sein, neue Lösungen und Technologien zu evaluieren.

2.1.7 Rolle Benutzer

Die Benutzer sind nach Züllighoven (1998, S. 91) die kompetenten Akteure. Sie erklären den Entwicklern ihre Tätigkeiten bzw. Arbeitsabläufe und sollten für Besprechungen und Usability-Tests zur Verfügung stehen. Durch Teilnahme an Usability-Tests am Prototyp können sie stark auf Gestaltungsentscheidungen einwirken und tragen dafür eine gewisse Verantwortung.

2.1.8 Rolle Vertriebsmitarbeiter

Der Vertriebsmitarbeiter ist dafür angestellt, die Produkte seiner Firma zu verkaufen. Er macht Angebote, handelt Verträge aus und gibt Einführungen und Beratung zu den Produkten. Die Angebote erstellt er auf Basis einer Analyse der Kundenanforderungen. Er prüft die Machbarkeit und die Produktrentabilität in Bezug auf Kundenwünsche, führt professionelle Kundenpräsentationen durch, übergibt nach Vertragsabschluss die Projekte an den zuständigen Systemspezialisten und betreibt permanente Kundenpflege, indem er seine Kunden stetig über Neuerungen beim Produkt informiert.

2.1.9 Rolle Systemspezialist

Der Systemspezialist ist der Ansprechpartner für technische Details eines Systems. Im Kontext dieser Arbeit arbeitet er beim Lieferanten, was bedeutet, dass das System, das der Systemspezialist betreut, für das Projektteam ein Fremdsystem ist. Hat beispielsweise ein Entwickler oder der Architekt Fragen zur Einbindung des Fremdsystems, ist der Systemspezialist der Ansprechpartner. Er kennt die sämtliche Schnittstellen, technische Gegebenheiten und Besonderheiten des Systems.

Im nächsten Kapitel geht es darum, die Situationen genauer zu schildern, die Rollen den entsprechenden Situationen zuzuordnen und typische Szenarien für die Situationen zu beschreiben.

2.2 Szenarien

Die im Folgenden aufgeführten Projektsituationen könnten den Anschein erwecken, Phasen wie im Wasserfallmodell zu sein. Sie sind aber lediglich als Situationen zu verstehen, wie sie in den meisten Softwareprojekten vorkommen und sind hier zumeist episodisch und exemplarisch dargestellt. Jede dieser Situationen kann iterativ sein und öfters durchlaufen werden. Somit unterliegen sie keiner fest vorgeschriebenen Reihenfolge und keinen Regeln innerhalb der Vorgehensmodelle. Sollte sich im Projektverlauf ergeben, dass eine Situation erneut durchlaufen werden muss, ist dies ganz im Sinne einer benutzerorientierten Entwicklung. Die im Folgenden niedergeschriebenen Situationen und Szenarien sind auf Basis von Interviews von Entwicklern in den Softwareentwicklungsabteilungen von IBM und Signal Iduna und Teilnahme an Meetings bei Signal Iduna erarbeitet worden.

2.2.1 Aufwandsabschätzung und Angebotserstellung

Dieser Situation geht vorweg, dass eine Anfrage bei einer Firma für Softwareentwicklung eingegangen ist. Über diese wird nun beraten. Entscheidend ist hierbei, ob das Projekt machbar ist und wo die Risiken liegen. Der Kalkulation des Preises und der Abrechnungsmethode (Festpreis oder Zeit + Material) liegen u. a. die Anforderungen des Kunden wie zum Beispiel die Arbeitsabläufe im Unternehmen, die das System unterstützen soll, und die Frage, ob Altsysteme migriert werden sollen, zu Grunde. Auf dieser Basis werden auch erste Use-Cases, eine grobe Architektur und ein operationales Systemmodell für Hardware-Anforderungen und -Kosten entworfen. Außerdem wird der Zeitrahmen mit so genannten Meilensteinen[1] festgelegt. Bei den Meilensteinen findet jeweils eine Abnahme durch den Kunden statt. Im Angebot werden bereits auch die Verantwortlichkeiten beider Parteien festgelegt. Speziell die Aufwandsschätzung zieht sich durch das komplette Softwareprojekt, zum Beispiel
damit der Projektleiter ermitteln kann, ob das Projekt im Rahmen des Budgets verläuft.

Teilnehmer in dieser ersten Runde sind ein Projektleiter, ein Architekt oder ein Architektenteam und evtl. Entwickler, die Erfahrung aus ähnlichen Projekten mitbringen. Das Resultat ist ein Angebot, das nach Unterzeichnung ein Vertrag mit Verantwort-

[1] definierte Zustände, die das System zu bestimmten Zeitpunkten erreicht haben muss

lichkeiten darstellt. Dieses wird vor der Zustellung an den Kunden vom Abteilungsleiter oder einer anderen dafür befugten Person abgezeichnet.

Szenario 1: Einblick in die Arbeitsabläufe beim Kunden

Vorbedingung:

Ein potentieller Kunde möchte vom Systemhaus ein Angebot über ein neues Softwaresystem erstellt bekommen. Hierfür findet ein Beratungsgespräch statt, in dem die Anforderungen an das System besprochen werden.

Szenario:

Um einen Überblick über die Anforderungen des Systems zu bekommen, muss das Team die Arbeitsabläufe beim Kunden sowie das Altsystem kennen lernen. Somit bespricht ein Vertreter des Projektteams mit einer oder mehreren dafür befugten Personen beim Kunden die Abläufe im Unternehmen. Die befugten Personen beim Kunden können zum Beispiel der Organisator, ein Sachbearbeiter oder auch der Systemadministrator sein. Die Arbeitsabläufe sowie die relevanten Daten über das Altsystem werden schriftlich erfasst.

Nachbedingung:

Das im Gespräch entstandene Dokument wird an alle Teammitglieder verteilt. Aus den Arbeitsabläufen werden Szenarios formuliert, die u. a. als Basis für die Systementwicklung dienen. Mit Hilfe dieser Dokumente wird eine grobe Systemarchitektur entworfen.

Szenario 2: Entwurf einer Architektur

Vorbedingung:

Auf Basis von Szenario 1 hat der Architekt erste Vorschläge für eine Systemarchitektur entworfen.

Szenario:

Der Architekt präsentiert dem Entwicklerteam seine Entwürfe. Während der darauf folgenden Diskussion entstehen Zeichnungen am Whiteboard. Teilweise werden diese Zeichnungen wieder verworfen, andere sind aber sehr wich-

tig und werden von einer Person, die auch das Protokoll schreiben wird, abgezeichnet. Ein Teammitglied erinnert an ein voriges Projekt, bei dem die Problemstellung ähnlich war und möchte die Architektur in die Diskussion einbringen. Um die alte Architektur zu holen, muss das Teammitglied den Raum verlassen, was eine meist unerwünschte Unterbrechung des Meetings zur Folge hat.

Nachbedingung:

Der Architekt stellt die Systemarchitektur wie im Meeting besprochen fertig. Ein anderer Teilnehmer hat das Protokoll geschrieben und die relevanten Zeichnungen vom Whiteboard digitalisiert. Die Dokumente werden nun an das Projektteam verschickt.

2.2.2 Fein-Analyse

Während der Feinanalyse wird auch eine Aufwandsschätzung durchgeführt, allerdings fließen hier präzisere Daten als Grundlage ein, da das Team inzwischen mehr Informationen zum Projekt bekommen hat. Auch werden soweit möglich markante Punkte und Ähnlichkeiten aus anderen Projekten zu Hilfe genommen. Die Use-Cases werden verfeinert und Leitfäden für Gespräche mit dem Enduser ausgearbeitet. Durch diese Gespräche können die Anforderungen an das zukünftige System ermittelt werden. Auch wird das Altsystem in Hinblick auf seine Strukturen und Daten genauer betrachtet, da diese in der Regel übernommen werden müssen. Des Weiteren sollte der Kunde dem Entwicklerteam Testdaten zur Verfügung stellen.
Die Architekten und Entwickler müssen die Arbeitsverteilung regeln und ihre Zeitschienen untereinander abstimmen.
Als Resultat dieser Aktivitäten gibt es das Konzept, den Projekt- und Einsatzplan und ggf. erste explorative oder experimentelle Prototypen.

Szenario 3: präzise Zeit- und Ressourcenplanung

Vorbedingung:

Der Projektleiter hat für die Erstellung des Angebots eine vorläufige Aufwandsschätzung gemacht, die er nun mit den Entwicklern bespricht.

Dieses Szenario tritt während eines Projekts öfter und in unterschiedlichen Situationen auf. Auf diese Weise kann der Projektleiter zeitliche Engpässe frühzeitig erkennen.

Szenario:

Aus der Festlegung über die zu verwendenden Technologien gibt sich die konkrete Zeit- und Ressourcenplanung, da die Entwickler möglichst ihren Kompetenzen nach eingesetzt werden sollen, um lange Einarbeitungszeiten zu vermeiden. Zu diesem Thema findet ein Meeting statt, in der die Teilnehmer zusammen sitzen und über den zeitlichen Rahmen und die benötigten Ressourcen des Projekts beraten. Hierfür liegen ihnen der Einsatzplan der Entwickler, an dem sie freie Ressourcen erkennen können, sowie ein Überblick über die Spezialgebiete der Entwickler vor.

Nachbedingung:

Eine Person schreibt und verteilt das Protokoll, in dem die Ergebnisse des Meetings festgehalten sind. Der Projektleiter pflegt den erarbeiteten Einsatzplan in ein Projektplanungstool ein.

Szenario 4: Verfeinerung der Use-Cases

Vorbedingung:

Im bisherigen Projektverlauf sind auf Basis der Szenarios erste Use-Cases entstanden.

Szenario:

Die bereits erstellten Use-Cases müssen überarbeitet bzw. verfeinert werden. Die verantwortlichen Teammitglieder treffen sich dafür im Konferenzraum und besprechen einen Use-Case nach dem anderen. Hierbei entstehen auch neue Use-Cases. Aus der Diskussion über die Use-Cases entstehen parallel Leitfäden bzw. Fragenkataloge für Gespräche mit den künftigen Nutzern des Systems. Das gesamte Material dient als Unterstützung der Testleiter bei den Usability-Tests.

Nachbedingung:

Das entstandenen Material wird schriftlich festgehalten und an allen Teammitglieder zur Verfügung gestellt, die es je nach Aufgabenbereich entsprechend nutzen.

2.2.3 Entwurf und Projektplanung

Diese Projektsituation beinhaltet die Planung des Projekts in Hinblick auf Zeit und Technologie. Es werden Pläne wie zum Beispiel ein Projektstrukturplan, ein Terminplan und ein Kostenplan erarbeitet. Des Weiteren wird die methodische Vorgehensweise besprochen, die einzusetzenden Technologien sowie die Programmiersprache festgelegt. Auch werden die Use-Cases komplettiert, d. h. sie werden mit allen Alternativen ausgearbeitet, die Gesamtarchitektur wird fertig gestellt (wobei diese im Entwicklungsprozess ggf. angepasst werden muss), Komponentenmodelle und operationale Systemmodelle sowie Daten- und Klassenmodelle werden ausgearbeitet. Zusätzlich werden Testfälle definiert und einige Screens vorbereitet. An diesem Projektabschnitt arbeiten hauptsächlich Architekten und Entwickler. Als Resultat entstehen Design-Dokumente.

Szenario 5: Speicherung von Testdaten

Vorbedingung:

Der Kunde liefert Testdaten.

Szenario:

Zum besseren Verständnis des Kontextes und damit das System realistisch getestet werden kann, benötigt das Entwicklungsteam Testdaten vom Kunden. Die vom Kunden gelieferten Daten werden vorerst anonymisiert, d. h. sie bleiben lediglich in ihrer Struktur erhalten. Der Inhalt wird verfälscht. Trotzdem sollten nur die Teammitglieder, die direkt mit den Daten arbeiten müssen, auf diese zugreifen können.

Nachbedingung:

Die anonymisierten Testdaten werden von einem ausgewählten Personenkreis – dem Entwicklerteam – zum Testen der Komponenten bzw. des Systems verwendet.

Szenario 6: Einigung über Technologie

Vorbedingung:

Die Mitglieder des Projektteams machen sich auf Basis der bisher erstellten Artefakte Gedanken über die Realisierbarkeit des Systems und die dafür benutzbaren Technologien.

Szenario:

Bei einem Meeting soll darüber beratschlagt werden, welche Technologien in dem Projekt verwendet werden sollen. Hierzu findet ein Brainstorming statt, in dem jeder die seiner Meinung nach möglichen Technologien aufführt. Anschließend werden Vor- und Nachteile für die jeweilige Technologie gesammelt. Jeder Teilnehmer, der eine neue Technologie vorgeschlagen hat, stellt diese vor. Die Entscheidung für oder gegen eine neue Technologie hängt u. a. von der Verteilung der Fachkompetenzen im Team und vom Projektbudget ab. Die Argumentation für eine bereits bekannte Technologie wird auf die Dokumentation aus vorherigen Projekten gestützt.

Dieses Szenario kann in dieser Form nur stattfinden, wenn der Kunde oder auch Projektleitung nicht auf die Nutzung bestimmter Technologien besteht.

Nachbedingung:

Ein Protokoll mit den Ergebnissen des Meetings wird erstellt und verteilt. Es folgt eine Einarbeitungsphase in die ausgewählte Technologie.

2.2.4 Implementierung

Diese Situation beschreibt die tatsächliche Entstehung des Systems. Während der Programmierung muss ausreichend dokumentiert werden, um später u. a. nachvollziehen zu können, wo evtl. etwas schief gelaufen ist und ob neue Technologien geholfen haben. Hier müssen beispielsweise die Meilensteindokumente abgearbeitet und vervollständigt werden. Im Mittelpunkt der Implementierung - wie auch in allen anderen Projektssituationen - sollte der Autor-Kritiker-Zyklus stehen. Die Kritiker werden in möglichst kurzen Abständen mit einem Prototyp konfrontiert, den sie testen und anschließend bewerten. Die Autoren analysieren daraufhin die Testergeb-

nisse und Bewertungen der Kritiker und modellieren den nächsten Prototyp. Auf diese Weise können ungünstige Entwicklungen sofort erkannt und verändert werden.

Szenario 7: Präsentation eines Prototyps

Vorbedingung:

Ein Prototyp wurde erstellt und Benutzer zum Testen sind eingeladen worden. Für den Nutzertest wurden Gesprächsleitfäden und Fragebögen entworfen.

Szenario:

Um möglichst den Anforderungen des Kunden zu entsprechen, werden ein oder mehrere Benutzer zur Präsentation eines Prototyps eingeladen. Hier kann der Kunde den Prototyp testen, während ein Mitglied des Projektteams die Anmerkungen des Kunden notiert. Nach dem Test führt der Testleiter ein Interview mit der Testperson durch. Hierzu benutzt er den Gesprächsleitfaden. Je nachdem, ob der Fragebogen ausschließlich Vorabinformationen über die Testperson und ihre Kenntnisse einfordert oder ob er Fragen zum Test und zum Prototypen beinhaltet, füllt die Testperson den Fragebogen vor oder nach dem Test aus.

Nachbedingung:

Die Kritik des Nutzers wird festgehalten und ausgewertet. Die Auswertung der Fragebögen und die Ergebnisse der Tests fließen bei der Weiterentwicklung des Prototyps ein. Neue Tests werden vorbereitet.

Szenario 8: Zwischenstand

Vorbedingung:

Das Projekt hat einen bei der Projektplanung definierten Stand erreicht bzw. ein bei der Planung festgelegter Zeitpunkt ist gekommen.

Szenario:

Es findet ein Meeting über den Zwischenstand des Projekts statt. Die Meilensteindokumente werden besprochen, sowie eventuelle Schwierigkeiten und Engpässe. Hierbei findet ein Brainstorming statt, durch das erarbeitet werden

soll, was bei der Projektarbeit gut läuft und was künftig verbessert werden sollte.

Nachbedingung:
Ein Protokoll über die Ergebnisse des Meetings wird verfasst und an das Projektteam verteilt. Die Ergebnisse aus dem Brainstorming sollten in den weiteren Verlauf des Projekts einfließen.

2.2.5 Auslieferung

Diese Situation beschreibt, wie das fertige System dem Kunden übergeben wird. Die Auslieferung muss sorgfältig geplant und vorbereitet werden. Die zukünftigen Nutzer des Systems müssen in das neue System eingeführt werden und ggf. eine Schulung erhalten. Sowohl die Einführung als auch die Schulung müssen vorbereitet werden. Somit muss sich die Person, die für diese Arbeitsschritte zuständig ist, rechtzeitig mit dem Entwicklungsteam in Verbindung setzen, um alle relevanten Informationen über das System zu erlangen und entsprechende Unterlagen und Präsentationen zu erstellen. Auch muss die Übergabe für das Rechenzentrum des Kunden gründlich mit Dokumentation und allen anderen wichtigen Dokumenten vorbereitet werden. Im Optimalfall findet anschließend noch eine Abschlussbesprechung mit allen Beteiligten statt.

Szenario 9: Vorbereitung für Schulung und Einführung

Vorbedingung:
Es steht genügend Dokumentationsmaterial zur Verfügung und die Funktionen sowie das User-Interface stehen soweit fest, dass eine Person Schulungsunterlagen und Dokumentation für den Benutzer erstellen kann.

Szenario:
Das System steht kurz vor der Auslieferung. Das bedeutet, dass das Team die Übergabe organisieren bzw. planen muss. Die Mitarbeiter beim Kunden müssen in die Benutzung des Systems eingeführt oder geschult werden. Die dazu nötigen Veranstaltungen müssen sorgfältig vorbereitet werden. Es wird ein Mitarbeiter auserkoren, der die Einführung bzw. Schulung durchführen soll. Diesem Mitarbeiter stellt das Team oder ein Teil des Teams das System vor.

Parallel machen sich die Teilnehmer daran, Punkte zu notieren, die beim Kunden erwähnt werden müssen. Dieses geschieht auch mit Hilfe der Dokumentation, die während der Projektlaufzeit entstanden ist. Auf dieser Basis stellt der Mitarbeiter seine Einführung und Schulungsunterlagen zusammen.

Nachbedingung:
Ein Protokoll über die Ergebnisse des Meetings wird verfasst und an das Projektteam verteilt. Schulungsunterlagen und eine Dokumentation des Systems werden erstellt.

Szenario 10: Abschlusspräsentation

Vorbedingung:
Das Projekt ist abgeschlossen und das System ausgeliefert.

Szenario:
Am Ende eines Softwareprojekts findet üblicherweise eine Abschlusspräsentation statt. Das gesamte Projektteam trifft sich im Konferenzraum, wo zunächst eine Präsentation über den Projektverlauf gehalten wird. Anschließend startet eine Diskussion über positive und negative Erfahrungen im Projekt.

Nachbedingung:
Ein Protokoll über die Ergebnisse des Meetings wird verfasst und an das Projektteam verteilt. Die Ergebnisse der Diskussion sollten in die künftige Arbeit des Teams einfließen.

2.2.6 Krisensituationen

Es gibt viele Gründe, weshalb ein Projekt scheitern kann. Von daher sollen an dieser Stelle auch Krisensituationen und deren Lösung betrachtet werden. Häufige Quellen von Krisensituationen sind u. a. zwischenmenschliche und technische Probleme. Wichtig ist, die Anzeichen für eine drohende Krise rechtzeitig zu erkennen und ihr somit vorbeugen zu können.

Viele Probleme lassen sich nicht während eines Meetings im Konferenzraum lösen, da die Atmosphäre oft zu förmlich wirken kann. Auch kommt es bei solchen Gesprächen oft vor, dass ein Schuldiger für die entstandene Krise gesucht wird. Da diese Schuld keiner auf sich nehmen möchte, versucht jeder, Beweise für seine Unschuld vorzubringen. Eine solche Diskussion verläuft in den meisten Fällen ergebnislos oder unterstützt sogar das Scheitern eines Projekts. Aus diesem Grunde werden diese eher zwischenmenschlichen Probleme gerne fernab vom Arbeitsplatz bei einem Geschäftsessen geklärt.
Technische Probleme hingegen sollten am besten früh erkannt und im Team gelöst werden. Tritt ein solches Problem auf, gilt es zu erörtern, wo und wie es entstanden ist. Hierbei ist beispielsweise zu recherchieren, ob die Komplexität eines Moduls nicht erkannt wurde oder ob es architektonische oder technische Fehler gab.

Szenario 11: Drohender Verzug

Vorbedingung:

Das System befindet sich in der Entwicklung. Bei einem Modul treten Problematiken auf, die vorher nicht bedacht worden sind.

Szenario:

Ein Entwickler wird mit seinem Modul nicht rechtzeitig fertig. Das Problem scheint ein Fehler im Programm zu sein, der noch nicht lokalisiert werden konnte. Gemeinsam sitzen die Entwickler mit dem Projektleiter, dem Architekten oder einem hinzugezogenen Experten und betrachten den Quellcode, um den oder die Fehler zu finden.

Nachbedingung:

An dieser Stelle gibt es mehrere Möglichkeiten. Zum einen kann der Fehler gefunden werden und das Projekt wird pünktlich fertig. Zum anderen kann der Fehler aber auch so gravierend sein, dass das Projekt tatsächlich in Verzug gerät. Bei diesem Fall muss über das weitere Vorgehen beratschlagt werden. Dieses geschieht ggf. zusammen mit einem Vertreter des Kunden.

2.3 Ergebnis

In diesem Kapitel wurden Situationen mit dazugehörigen Szenarien sowie die Rollen der Mitwirkenden erarbeitet, die laut Erhebung typischerweise in einem Softwareprojekt vorkommen. Bei Betrachtung der Situationen sind von der Thematik her deutliche Unterschiede zu erkennen. Dies ist eine logische Konsequenz daraus, dass das Projekt immer weiter fortschreitet und sich die Diskussionsgrundlage ändert. Anhand der Szenarios ist dies nachvollziehbar.

Für die Herausarbeitung der Anforderungen an ein Konferenzsystem ist es wichtig, die Gemeinsamkeiten der Situationen und Szenarien herauszuarbeiten. Werden die Gemeinsamkeiten erkannt, birgt das die Möglichkeit, einen Werkzeugkasten zu erstellen, mit dem beliebige Dokumente – egal in welcher Situation sie vorkommen – veröffentlicht und gemeinsam bearbeitet werden können. Folglich sind die einzelnen Projektphasen, die in den Vorgehensmodellen gerne hervorgehoben werden, und die einzelnen Situationen und Szenarien für diese Analyse nicht relevant.

Somit beschäftigt sich das folgende Kapitel mit der Herausarbeitung der Gemeinsamkeiten und der daraus resultierenden Anforderungen sowie mit der Erarbeitung eines Benutzungsmodells zur einfachen, intuitiven Bedienbarkeit des Konferenzsystems.

3 Anforderungen an arbeitsunterstützende Methoden

In diesem Teil der Arbeit werden die in Kapitel 2 erarbeiteten Projektsituationen und Szenarios genauer betrachtet. Aus diesen Szenarios werden nun einige beispielhafte Workflows herausgearbeitet, die typischer Weise in einem Softwareprojekt vorkommen. Zusätzlich werden Ansätze zur Ausarbeitung eines Benutzungsmodells vorgestellt und zum Entwurf des Konferenzsystems diskutiert. Aus diesen Grundlagen können im Anschluss wichtige Anforderungen an das Konferenzsystem abgeleitet werden.

3.1 Workflows

Ein Workflow oder auch Geschäftsprozess ist die Bezeichnung für eine Abfolge von Tätigkeiten in einer definierten Reihenfolge, die zur Schaffung eines Produkts oder auch Arbeitsergebnisses dienen und in einem direkten Zusammenhang stehen. Diese erfassen ein Spektrum, das von einfachen Prozessen bis zu komplexen, organisationsweiten bzw. organisationsübergreifenden Vorgängen reicht (WfMC 2006). Workflow-Modelle sollen dabei helfen, die optimale Einbindung verschiedenster Applikationen - wie beispielsweise bei einer Versicherung eine Anwendung zum Erstellen von Versicherungsverträgen - in die jeweiligen Arbeitsabläufe sicherzustellen.
Übertragen können auch die kooperativen Prozesse innerhalb der Anwendungsentwicklung als Workflows angesehen und entsprechend beschrieben werden. Dieses ist erforderlich, um eine Werkzeugunterstützung planen zu können, die über die typische und bereits vorhandene Unterstützung einzelner Schritte hinausgeht. Beispielhaft werden hier, angelehnt an die Szenarien aus Kapitel 2.2, folgende Arbeitsabläufe betrachtet:

- Aufwandsschätzung
- Entwurf der Architektur
- Use-Cases definieren und verfeinern
- Entwurf und Projektplanung
- Implementierung
- Erstellen von Schulungsunterlagen

Zur Analyse der hier aufgelisteten Szenarien wird deren Ablauf kurz beschrieben, um darauf basierend Workflow-Modelle zu den Szenarien passend zu entwickeln.

Aufwandsschätzung

Als Grundlage für eine erste Schätzung werden die Daten aus den vorherigen Projekten hinzugezogen sowie die in der Grobanalyse entstandenen Dokumente. Daraus wird eine erste Schätzung erstellt. Das Projekt wird auf Basis der ersten Schätzung umgesetzt, wobei Artefakte[1] entstehen. Während des Lebenszyklusses des Projekts wird die Aufwandsschätzung immer wieder überarbeitet und es wird dabei überprüft, ob die Schätzung mit dem aktuellen Stand des Projekts harmoniert. Bei jeder neuen Aufwandsschätzung fließt die vorige Version sowie die Artefakte bzw. der aktuelle Projektstand ein. Da die Aufwandsschätzung in Iterationen während des Projekts durchgeführt wird, ist sie mit Abschluss des Projekts beendet und wird als fertiges Dokument abgelegt. Ggf. kann es Nachkalkulationen geben.

Entwurf der Architektur

Jedem System liegt eine Architektur zu Grunde. Die Anforderungen, Gesprächsprotokolle, Daten über das Altsystem und die Arbeitsabläufe dienen als Grundlage für die Architektur. Daten aus vorherigen Projekten können zusätzlich als Muster bzw. Vorlage für ein ähnliches System bzw. ähnliche Komponenten genutzt werden. Der Entwurf der Architektur ist ein iterativer Vorgang. Nach einem ersten Entwurf wird permanent reflektiert, ob die Architektur zum System passt oder ob sie verändert werden muss. So fließen bei jeder Iteration der alte Entwurf und die Überlegungen zu Änderungen in den nächsten Architekturentwurf ein.

Use-Cases definieren und verfeinern

Ein Use-Case zeigt das externe Verhalten eines Systems aus der Sicht der Benutzer, indem es die Benutzer, die Use-Cases und deren Beziehungen zueinander darstellt. Ein Benutzer kann eine Person, aber auch ein Nachbarsystem sein. Use-Cases bilden die Reaktion des Systems auf Ereignisse seiner Umwelt ab und fassen dabei Teile der Systemdienstleistung zusammen (Jeckle 2003, S. 175).

[1] Ein Artefakt ist ein Element im Komponentendiagramm, das eine physische Informationseinheit darstellt, zum Beispiel ein Modell, eine Quellcode-Datei, eine ausführbare Binärdatei, eine Archivdatei (Balzert 2005, S. 527).

Das Definieren und Verfeinern von Use-Cases ist ein ständig wiederkehrender Ablauf in Softwareprojekten. Als Grundlage für den Entwurf von Use-Cases dienen hauptsächlich Protokolle aus Gesprächen mit Anwendern, da diese die Arbeitsabläufe im Unternehmen am besten wiedergeben und die Erwartungen, die an das neue System gestellt werden, beschreiben können. Als Vorlage für ähnliche Systeme können auch die Use-Cases aus vorherigen Projekten dienen. Use-Cases unterliegen in ihrer Entstehung einem Zyklus, da sich beispielsweise zwischendurch herausstellen kann, dass ein Use-Case zu ungenau ist und in mehrere Use-Cases unterteilt werden sollte. So werden vorerst einige Use-Cases erstellt, die im Laufe des Projekts verfeinert werden. Der Verfeinerung liegen zusätzlich zu den Gesprächsprotokollen, von denen es während des Projekts auch immer wieder neue geben wird, die vorherigen Versionen der Use-Cases zu Grunde.

Entwurf und Projektplanung

Diese Projektsituation beinhaltet die Planung des Projekts in Hinblick auf Zeit und Technologie. Es werden Pläne wie zum Beispiel ein Projektstrukturplan, ein Terminplan und ein Kostenplan erarbeitet. Des Weiteren wird die methodische Vorgehensweise besprochen, die einzusetzenden Technologien sowie die Programmiersprache festgelegt. Auch werden die Use-Cases komplettiert, d. h. sie werden mit allen Alternativen ausgearbeitet, die Gesamtarchitektur wird fertig gestellt (wobei diese im Entwicklungsprozess ggf. angepasst werden muss), Komponentenmodelle und operationale Systemmodelle sowie Daten- und Klassenmodelle werden ausgearbeitet. Zusätzlich werden Testfälle definiert und einige Screens vorbereitet.

Implementierung

Die Implementierung ist die Umsetzung der Planung. Hier wird das System programmiert. Die Implementierung erfolgt also auf Basis aller bisher erzeugten Dokumente. Als weitere Basis können auch Daten aus vorherigen Projekten dienen, da zum Beispiel manche Systemkomponenten oder Funktionen wieder verwendet werden können. Die Implementierung findet idealer Weise auch zyklisch statt. Nach jeder Iteration entsteht ein neuer Prototyp, der möglichst unter Einbeziehung potentieller Benutzer getestet wird. Die aus den Tests resultierenden Änderungsvorschläge sowie der letzte Prototyp dienen als Basis für den nächsten Entwicklungszyklus.

Schulungsunterlagen erstellen

Bevor ein System ausgeliefert wird, sollten Unterlagen für eine Schulung der künftigen Benutzer sowie eine Anleitung für die Systempfleger entstehen. Als Basis für die Schulungsunterlagen dienen die gesamte Projektdokumentation sowie einige Daten aus vorherigen Projekten, die auch hier als Vorlagen dienen können. Die Ausarbeitung der Unterlagen ist wie die anderen Szenarien zyklisch, da nach dem Entwurf der ersten Version über die Dokumente diskutiert wird. In den Diskussionen entstehen Änderungsvorschläge, die zusammen mit der vorherigen Version als Basis für die nächste Version der Unterlagen genutzt werden.

Aus den Beschreibungen der Szenarien ist zu entnehmen, dass sie alle einen ähnlich aufgebauten Ablauf haben, der in Abbildung 5 schematisch dargestellt wird.

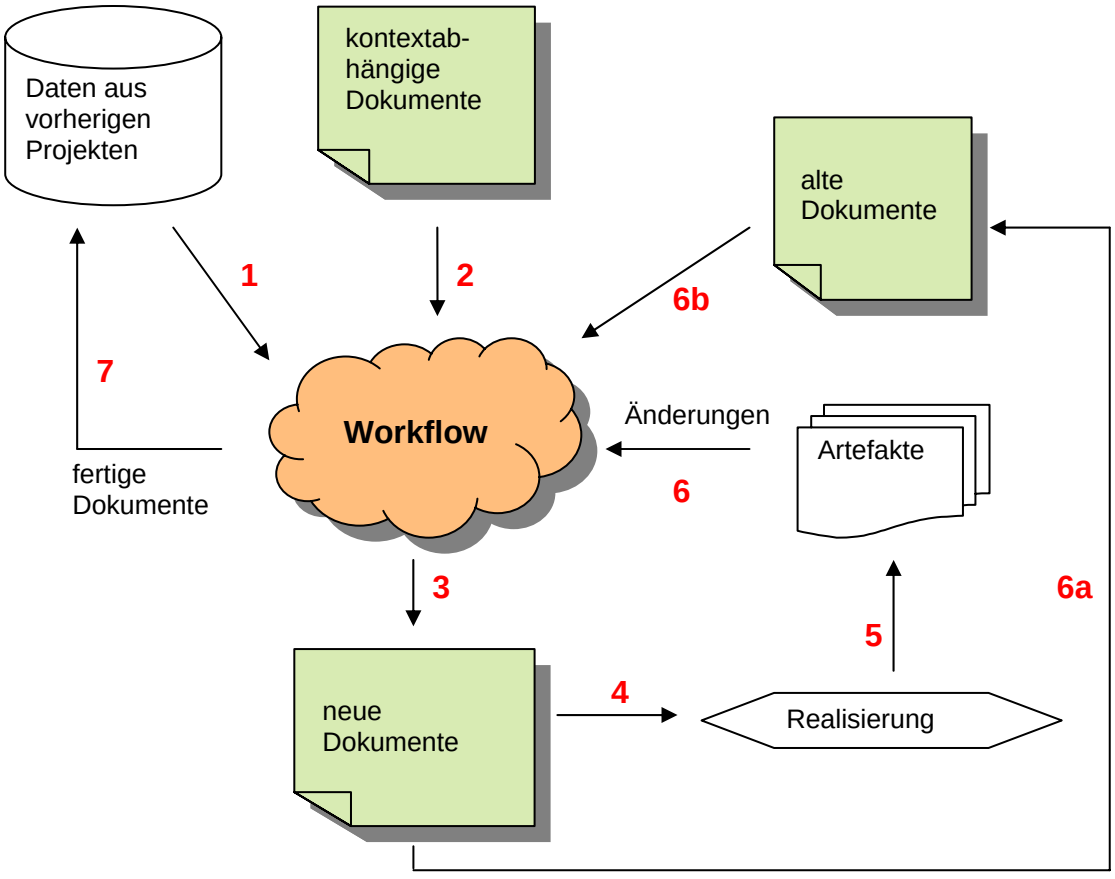

Abbildung 5: Schematischer Aufbau der Workflows

So werden in allen Situationen als Basis die „Daten aus vorherigen Projekten“(1) sowie die kontextabhängigen Dokumente (2) genutzt. Zu den kontextabhängigen Dokumenten gehören zum Beispiel bei dem Entwurf der Architektur die Anforderungen

an das System, Gesprächsprotokolle und Daten über das Altsystem. Nachdem auf dieser Basis neue Dokumente (3) entstehen, werden diese umgesetzt bzw. verarbeitet (4), woraus Artefakte resultieren (5). Ein Artefakt ist ein Dokument, das bearbeitet wurde wie zum Beispiel ein Use-Case. Durchläuft das Artefakt einen weiteren Zyklus (6), fließt es zusammen mit seiner vorherigen Version als Ausgangsbasis (6a und 6b) in den erneuten Durchlauf ein. Ist der Vorgang abgeschlossen und das Dokument in finaler Version erstellt, wird es als fertiges Dokument abgelegt (7). Der Zeitpunkt der finalen Version kann mit Projektende, aber auch während des Projekts erreicht sein. Da der gesamte Entwicklungsvorgang zyklisch ist und die Dokumente voneinander abhängig sind, ist die wirklich endgültige Version eines Dokuments erst am Projektende erreicht.

3.1.1 Ergebnis

In allen hier aufgeführten Workflows ist ein einheitlicher Aufbau mit drei Zyklen erkennbar – die Planung, die Durchführung und die Iteration. Die Planung umfasst den Zugriff auf die vorherigen Projekte, die eine Basis für ein neues Projekt bieten können. Nach Abschluss des Projekts wird es zu einem vorherigen Projekt. Die Durchführung ist die Umsetzung des jeweiligen Workflows und enthält die Abfolge der Nummern 3 bis 6 in Abbildung 5 und wird in der Iteration so oft durchlaufen, bis ein vom Projektteam akzeptierter Zustand erreicht ist.

Hier wurden nur einige Beispiele für Workflows des Entwicklungsprozesses betrachtet. Die dabei gewonnenen Beobachtungen sind so grundlegend, dass sie auch für die ausgelassenen Workflows formgebend sein dürften. So ist hier beispielsweise kein expliziter Workflow für die Qualitätssicherung aufgeführt, wobei diese u. a. in den Nutzertests während der Implementierung zu finden ist. Typisch für den Entwicklungsprozess ist weiterhin: die Bearbeitung der Workflows kennt keine auf verschiedene Personen aufgeteilte Rollen. Es gibt auch kein verbindliches Modell für die kooperative Bearbeitung. Die Workflows könnten zum Beispiel bei einem sehr kleinen Projekt im Extremfall durchaus von einer Person allein ausgeführt werden. In größeren Projekten ist jedoch die gemeinschaftliche, interdisziplinäre Zusammenarbeit vorzuziehen. Im Konferenzraum ist diese durch stetig wechselnde Initiative der Beteiligten gekennzeichnet. Eine Ausnahme bildet das Brainstorming, bei dem alle Teilnehmer parallel bzw. nebenläufig ihre Gedanken und Assoziationen betrachten.

Aus diesen Anhaltspunkten kann für jeden Workflow ein einheitlicher Rahmenplan für die Bildschirmnutzung (siehe Abbildung 6) abgeleitet werden.

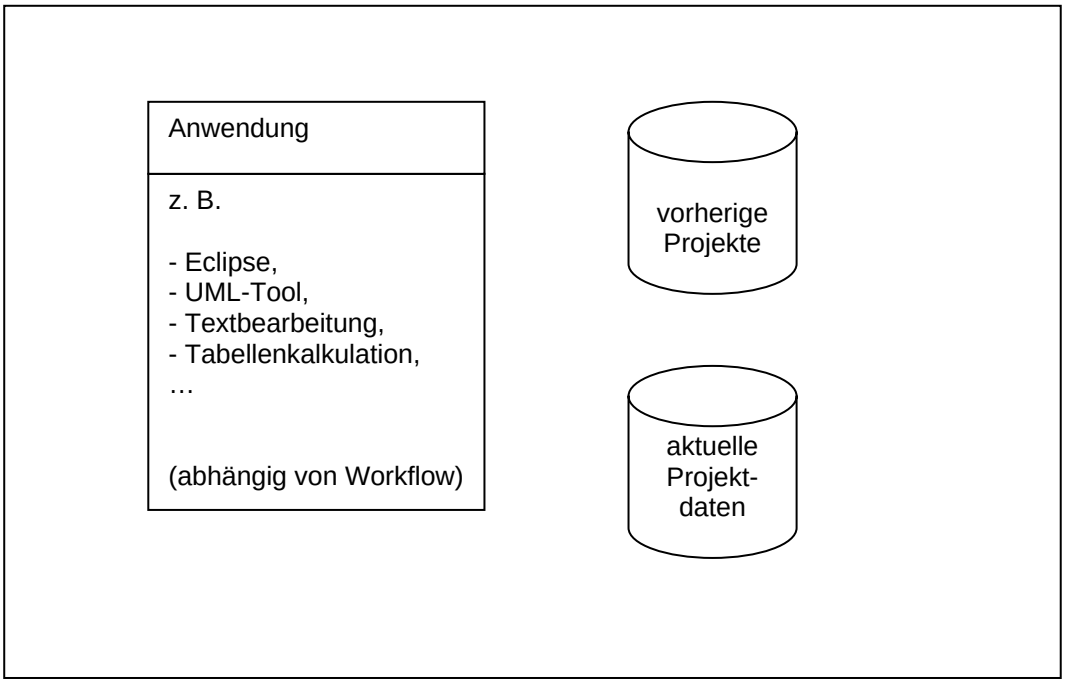

Abbildung 6: Wandtafel (grobes Layout)

Es wird momentan jeweils eine Anwendung benötigt, mit der der entsprechende Workflow realisiert werden kann. Im Fall der Implementierung könnte die Anwendung beispielsweise Eclipse oder ein anderes Tool zur Programmierung sein. Für die Erstellung von Use-Cases wird wiederum ein UML-Tool benötigt. Diese Tools kooperieren untereinander nicht. So ist es derzeit schwer möglich, den Zusammenhang zwischen einer Kette von aufeinander aufbauenden Artefakten darzustellen. Solche Ketten entstehen im Laufe eines Projekts. Zu Beginn werden zum Beispiel die Arbeitsabläufe beim Kunden aufgenommen, daraus werden Szenarios entwickelt, woraus dann wiederum Use-Cases und Workflows abgeleitet werden. Auf der Basis wird anschließend programmiert und es entsteht Quellcode. Für die Erstellung der jeweiligen Artefakte werden unterschiedliche Anwendungen benötigt, die untereinander nicht kooperieren.

Trotz der Unterschiede, die in der Anwendung liegen, werden nach Abbildung 5 für jeden Workflow die gleichen Arbeitsschritte durchgeführt. Zusätzlich könnten für jedes Artefakt die gleichen Methoden und Werkzeuge zur Bearbeitung genutzt werden. Schließlich sind in jeder Einzelanwendung bis auf einige Sonderfunktionen die gleichen Funktionen implementiert, nur abgestimmt auf einen bestimmten Dokumenttyp.

Auch die Informationsbasis, in der die Artefakte liegen, bestehend aus Historie, Kontext und Evolution, ist unabhängig von der Anwendung gleich.
Im weiteren Verlauf der Arbeit ist es nun wichtig, den in Abbildung 6 dargestellten Ansatz aufzugreifen und dahingehend zu entwickeln, dass die unterschiedlichen Werkzeuge kooperieren bzw. ein Modell geschaffen wird, indem die Beziehungen der einzelnen Artefakte erkennbar werden. Auch sollte eine gemeinsame Datenbasis geschaffen werden, so dass die Nutzer jederzeit auf sämtliche Projektdaten zugreifen können und nicht nur auf einen durch ein Tool eingeschränkten Datenbestand.
Da das System einfach und intuitiv bedienbar sein soll, ist dabei nach einem anwendungsorientierten Ansatz vorzugehen. Ein solcher Entwicklungsansatz ist der WAM-Ansatz, der für die gute Bedienbarkeit die Entwicklung eines Benutzungsmodells vorsieht. Das folgende Kapitel befasst sich deshalb mit der Entwicklung eines solchen Modells.

3.2 Benutzungsmodell

Aus der Betrachtung der Workflows ist zu entnehmen, dass sie einen recht ähnlichen Aufbau haben und sich hauptsächlich darin unterscheiden, welche Daten direkt zur Verfügung stehen müssen und welche Tools zur Bearbeitung der Thematik benutzt werden.
Die Gemeinsamkeiten der Szenarien liegen darin, dass…

- …Beratung und Diskussion über ein Thema stattfindet.
- …Präsentation gehalten werden.
- …Brainstorming stattfindet.
- …Protokolle geschrieben werden.
- …Zugriffsbeschränkungen bestehen sollten.
- …Zeichnungen und andere Dokumente erstellt werden.

Somit muss das System Werkzeuge zur Verfügung stellen, die die benötigten Tätigkeiten unterstützen und je nach Workflow die entsprechenden Daten direkt zugreifbar machen. Es muss natürlich auch möglich sein, beispielsweise während der Verfeine-

rung der Use-Cases auch auf sämtliche andere Projektdokumente zuzugreifen, jedoch sollten die wichtigsten Daten direkt bzw. offensichtlich erreichbar sein.
Um diese Aspekte in ein System einfließen zu lassen, empfiehlt sich nach Züllighoven (1998, S. 71) die Entwicklung eines Benutzungsmodells. Es soll verdeutlichen, mit welchen Mitteln und auf welche Weise die anstehenden Aufgaben unterstützt werden.
Die Informatik ist im Vergleich zu anderen Wissenschaften wie zum Beispiel Fahrzeugtechnik oder der Entwicklung von Werkzeugen eine vergleichsweise junge Wissenschaft. Die hier genannten Forschungsgebiete greifen in ihrer Entwicklung auf teilweise Jahrhunderte lange Erfahrung in der Handhabung der produzierten Gegenstände zurück. Dementsprechend sind die Produkte sehr gut auf die Anatomie des Benutzers zugeschnitten. Auf solch fundierte Erfahrung im Bereich der Ergonomie kann die Informatik noch nicht zurückgreifen, wodurch es manchen Softwareprodukten an Gebrauchstauglichkeit fehlt. Es gibt diverse Ansätze, wie z. B. die Ergonomierichtlinien und Gestaltungsgrundsätze der ISO-Normen 13407 und 9241-10 (CEN 1995) und der Ansatz des User Centered Design (Neumann 2005), die den Entwicklern Möglichkeiten aufzeigen, wie gebrauchstaugliche Software entstehen kann und welche Kriterien gebrauchstaugliche Software erfüllen sollte.

3.2.1 Leitbild und Entwurfsmetaphern

Züllighoven beschreibt in seinem Buch über den WAM-Ansatz eine Möglichkeit, anwendungsorientiert Software zu erstellen. Hierbei legt er sehr großen Wert auf das Benutzungsmodell, da es unerlässlich für die Erstellung ergonomischer Systeme sei. Eine direkte Anleitung gibt auch er nicht, da jedes System in einem anderen Kontext steht und die Nutzer sehr unterschiedliche Erwartungen haben. Somit ist es schwer möglich, ein generelles Benutzungsmodell festzulegen. Züllighoven gibt den Anreiz, es mit Hilfe der im WAM-Ansatz beschriebenen Leitbildern und Entwurfsmetaphern zu erarbeiten. Im Folgenden soll in Anlehnung an den WAM-Ansatz ein Benutzungsmodell für das Konferenzsystem entwickelt werden.
Für diesen Zweck muss vorerst ein Leitbild[1] gefunden werden, das zu dem System passt.

[1] Ein Leitbild in der Softwareentwicklung gibt im Entwicklungsprozess und für den Einsatz einen gemeinsamen Orientierungsrahmen für die beteiligten Gruppen. Es unterstützt den Entwurf, die Verwendung und die Bewertung von Software und basiert auf Wertvorstellungen und Zielsetzungen. Ein Leitbild kann konstruktiv oder analytisch verwendet werden (Züllighoven 1998, S. 73).

Im Falle des Konferenzsystems kommt das klassische WAM-Leitbild vom Arbeitsplatz für eigenverantwortliche Expertentätigkeit in Frage, da an jedem Rechner im Konferenzraum Experten sitzen, die ihre Materialien selbständig mit den zur Verfügung stehenden Werkzeugen bearbeiten wollen. Dabei wird auch der spezielle kooperative Charakter von Entwicklungsprozessen berücksichtigt, der im Diskussionsprozess durch ständig wechselnde Initiativen gekennzeichnet ist. Das hier gewählte Leitbild hat eine unterstützende Sichtweise auf die Arbeit mit dem System. Diese Sichtweise hat folgende Merkmale (Züllighoven 1998, S. 81):

- Experten erledigen häufig wechselnde Aufgaben von hoher Qualität.
- Die Arbeit erfordert einen hohen Grad an Qualifikation, Kenntnissen und Erfahrung, der nicht formalisiert werden kann.
- Die Erledigung von Aufgaben orientiert sich zielgerichtet an vorhandenen Arbeitsmitteln und -gegenständen und nicht an vorgegebenen Routinen.
- Die situative Auswahl von Arbeitsschritten wird durch die vorhandenen Mittel unterstützt und führt zu jeweils adäquaten Ergebnissen.
- Pläne werden zur Orientierung und nicht als ausführbare Handlungsvorschrift betrachtet.
- Die Kontrolle über die Handhabung verbleibt beim Menschen.

Dieses Leitbild unterstützt einen Experten, der an einem Arbeitsplatz seine Tätigkeit ausführt. Im Konferenzraum tritt allerdings auch die Situation ein, dass mehrere Experten ein Dokument bearbeiten. Möglicherweise haben diese Experten auch unterschiedliche Ziele, die in der Diskussion vereint werden müssen. Diese Situation könnte zur Folge haben, dass das Leitbild überarbeitet werden muss. Hier wird aber vorerst angenommen, dass das Leitbild zutreffend ist und sich die gemeinsame Arbeit lediglich auf die Gestaltung des Werkzeugs auswirkt.
Da nach Züllighoven das Leitbild und Entwurfsmetaphern[1] verwendet werden sollten, um ein Benutzungsmodell zu erarbeiten, bleibt an dieser Stelle zu erklären, was Entwurfmetaphern sind und wozu sie genutzt werden.

[1] Eine Entwurfsmetapher ist eine bildhafte, gegenständliche Vorstellung, die ein Leitbild fachlich und konstruktiv „ausgestaltet“. Sie strukturiert die Wahrnehmung, trägt zur Begriffsbildung bei und leitet die Vorstellung und Kommunikation über das, was fachlich analysiert, modelliert und technisch realisiert werden soll. Sie dient der Gestaltung von Softwaresystemen, indem sie Handhabung und Funktionalität für die Beteiligten verständlicher macht (Züllighoven 1998, S. 79)

Metaphern im Allgemeinen sind sprachliche Ausdrucksmittel, in denen ein Wort nicht in seinem eigentlichen Sinn sondern in einer übertragenen Bedeutung gebraucht wird.
Bei einem Windows-System sind diverse Entwurfsmetaphern zu finden. So gibt es zum Beispiel Fenster, den Papierkorb, Ordner, das Briefsymbol zum Schreiben von Mails und das Papier mit Linien für Textdokumente.
Das Leitbild soll durch plastische Entwurfsmetaphern konkretisiert werden. Somit müssen Leitbild und Entwurfmetaphern bruchlos zueinander passen. Sie müssen die Analyse des Anwendungsbereichs, den Entwurf und die Verwendung des Systems durchgängig unterstützen. Zusätzlich müssen die Entwurfsmetaphern neben ihrer fachlichen auch eine technische Interpretation haben (Züllighoven 1998, S. 80).
Das Leitbild vom Arbeitsplatz für eigenverantwortliche Expertentätigkeit wird durch die Entwurfsmetaphern *Werkzeug* und *Material* konkretisiert. So fallen beispielsweise Mappen, Formulare oder auch Programmtexte unter den Begriff Material. Debugger und Schreibgerät hingegen sind Werkzeuge.
Was Werkzeug und was Material darstellt, ist erst in einer Arbeitssituation erkennbar, da es durchaus Gegenstände gibt, die in der einen Situation als Werkzeug gebraucht werden und in der anderen zum Material werden. In einer bestimmten Situation muss es aber eindeutig zu erkennen sein, welcher Gegenstand welche Rolle einnimmt.
Eine weitere Entwurfsmetapher ist die Arbeitsumgebung. Sie ist der Ort, an dem Werkzeuge, Materialien und andere Gegenstände der Arbeit ihren Platz haben. Hierbei wird zwischen dem Arbeitsplatz und der Arbeitsumgebung unterschieden. Am Arbeitsplatz findet die Arbeit selbst statt, die Arbeitsumgebung beinhaltet auch noch die Orte, die um den Arbeitsplatz herum zugänglich sind.
Da sich nicht alle Hilfsmittel, die in einer Arbeitsumgebung bei der Erledigung von Aufgaben benötigt werden, in Werkzeug und Material aufteilen lassen, wurde die Entwurfsmetapher des Automaten entwickelt. Der Automat soll dem Anwender Routinetätigkeiten abnehmen. In dem hier gewählten Leitbild muss der Automat nach dem Start im Hintergrund laufen und seine Routine ausführen. Dadurch dass der Anwender bestimmt, wann der Automat gestartet wird, passt er in das Leitbild der eigenverantwortlichen Expertentätigkeit und ist einem Werkzeug recht ähnlich. Die Erstellung einer Statistik könnte zum Beispiel durch einen Automaten realisiert werden.

Eine weitere wichtige Entwurfsmetapher ist der Behälter. Er dient als Sammelstelle für Materialien. In einem Behälter lassen sich beispielsweise Dokumente gruppieren. Beim Konferenzsystem könnte zum Beispiel ein Behälter für die Use-Cases vorhanden sein und ein weiterer für die alten Projektdaten. Der Behälter mit den alten Projektdaten würde wiederum weitere Behälter mit den Daten der einzelnen Projekte enthalten. Auch sollte jeder Benutzer einen privaten für die anderen Nutzer nicht zugänglichen Behälter haben, in dem er seine vorbereiteten Materialien zur Konferenz mitbringt bzw. ablegen kann.

Züllighoven beschreibt in seinem Ansatz zwar, mit welchen Mitteln ein Benutzungsmodell erarbeitet werden kann und was der Begriff in etwa bedeutet, eine richtige Anleitung und weitere Zusammenhänge stellt er jedoch nicht vor. Es gibt allerdings noch weitere Dinge, die bei der Erstellung eines Benutzungsmodells bedacht werden sollten. Im Wesentlichen geht es um die Beantwortung der folgenden drei Fragen:

1. Wie wird die Anwendungssoftware benutzt?
   - Aufgabe?
   - Einsatzkontext?

2. Was ist wichtig für die Benutzbarkeit?
   - Handhabung und Präsentation, Ergonomie
   - die fachlichen Gegenstände

3. Bild des Arbeitsplatzes
   - Leitbilder
   - (Entwurfs-) Metaphern.

Bei der Erstellung eines Benutzungsmodells sollte zunächst bedacht werden, dass es zwei Sichtweisen gibt: die des Benutzers und die des Entwicklers bzw. der Anwendung (Applikation). Diese Sichtweisen sind in der Regel unterschiedlich und sollten in Einklang gebracht werden. Abbildung 7 soll diese Zusammenhänge deutlich machen.

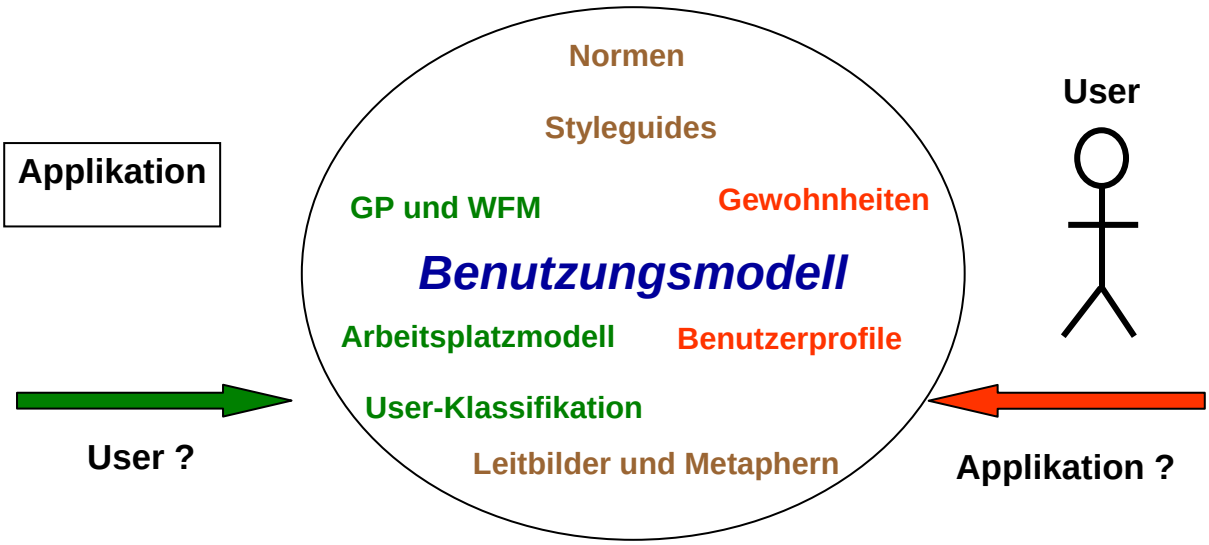

Abbildung 7: Benutzungsmodell (Raasch u. a. 2006)

Der Applikation bzw. den Entwicklern ist der Benutzer (User) unbekannt und dem Benutzer ist die Applikation vorerst unbekannt. Der Applikation liegen die im Entwicklungsprozess erkannten Geschäftsprozesse (GP), das Workflowmanagement (WMF), das Arbeitsplatzmodell und die User-Klassifikation zu Grunde. Diese Grundlagen hat das Entwicklerteam während des Entwicklungsprozesses – wie in den Szenarien beschrieben – erarbeitet. Auf der anderen Seite steht der Benutzer, der gewisse Gewohnheiten hat und ein Benutzerprofil verkörpert. Das Benutzerprofil kann zum Beispiel Anfänger, Fortgeschrittener oder Experte sein. Diese beiden Einflussgrößen sind individuell und von Benutzer zu Benutzer unterschiedlich. So bevorzugt der eine die Arbeit mit der Maus, dem andern geht dies zu langsam, er benutzt lieber die Tastatur zur Steuerung des Systems. Ein anderer wiederum nutzt eine Kombination aus beiden Geräten. Somit kann die Seite des Benutzers nur grob kategorisiert werden in beispielsweise Anfänger, Fortgeschrittener, Power-User und Gelegenheitsnutzer. An dieser Stelle stehen User-Klassifikation und Benutzerprofil im Zusammenhang. Es sollte für jedes Benutzermodell eine User-Klassifikation geben und umgekehrt. Neben diesen Einflussgrößen aus der Praxis gibt es Vorgaben bzw. Richtlinien aus der Theorie. Dazu gehören Normen, Styleguides und Leitbilder und Metaphern (Neumann 2005). Als eine weitere Einflussgröße, die auf Abbildung 7 nicht zu sehen ist, sollten die Ergebnisse von Usability-Tests und Benutzerbefragungen ins Benutzungsmodell einwirken. Wie in den vorher betrachteten Workflows deutlich wird, sollen diese Erfahrungen als Basismaterial im Entwicklungsprozess vorliegen.

Auf der Basis der in diesem Abschnitt dargestellten Fakten gilt es, im weiteren Verlauf der Arbeit, ein Benutzungsmodell für das Konferenzsystem zu entwickeln. Dazu gehört u. a. auch das äußere Erscheinungsbild – also die GUI – der Anwendung. Um eine leichte Bedienung zu ermöglichen und den Nutzern eine möglichst vertraute Umgebung zu schaffen, sollte das Konferenzsystem auf einem Fenstermanagementsystem aufsetzen.

3.2.2 WIMP[1]

Die WIMP-Metapher hat sich in nahezu allen Systemen als klassische 2D-Benutzungsoberfläche durchgesetzt. So unterscheiden sich die Oberflächen von Windows, Macintosh und Unix-Systemen nur im Detail. Auch andere elektronische Geräte wie Mobiltelefone und PDAs halten sich an diesen Gestaltungsansatz.
Ein Programm nach WIMP besteht aus einem rechteckigen Fenster, das weitere Steuerelemente enthält. Dazu zählen globale und kontextsensitive Menüs, Symbole (Icons) und verschiedene Formen von Eingabefeldern.
Die Menüs werden in globale und kontextsensitive Menüs unterteilt. Globale Menüs bieten den Benutzern eine hierarchisch gegliederte Auflistung aller im Programm verfügbaren Funktionen, die die Benutzer mit der Maus auswählen können. Im Gegensatz dazu bieten kontextsensitive Menüs nur eine auf die Menge der derzeit möglichen Aktionen reduzierte Liste an. Aktionen sind dabei Kommandos, die an die Applikation übermittelt werden.
Bei den Symbolen handelt es sich um kleine Piktogramme, die mit einer Programmfunktion verknüpft sind. Das Bild eines Druckers könnte dem Benutzer die Möglichkeit geben, das gerade aktive Dokument zu drucken.
Eingabefelder erlauben es dem Benutzer, einzelne Werte oder ganze Zeichenketten einzugeben.
Als Eingabegeräte stehen dem Benutzer meist Maus und Tastatur zur Verfügung. Die Tastatur dient zum Eingeben von Buchstabenfolgen oder dem Aktivieren von speziellen Programmroutinen. Bei letzterem handelt es sich meist um so genannte Shortcuts, die als Alternative zur Auswahl eines Menüeintrags oder Icons mit der Maus verwendet werden kann. Die Maus wird verwendet, um einen auf dem Bildschirm eingeblendeten Zeiger zu steuern und mit einer der Tasten Steuerelemente auszuwählen (Brauner 2004, S. 4).

[1] WIMP steht für Windows, Icons, Menus, Pionting device.

Demnach sind die meisten Programme, die auf einer grafischen Benutzungsschnittstelle mit einem Fenstermanagementsystem basieren, nach der WIMP-Metapher erstellt. So ist die WIMP-Metapher nicht ausschließlich beim klassischen Windows-PC zu finden sondern zum Beispiel auch beim Macintosh und der KDE von Linux. Bei Verwendung von WIMP-Programmen würde dem Benutzer des Konferenzsystems also eine recht vertraute Umgebung erscheinen. Allerdings bringen WIMP-Anwendungen auch einige Nachteile mit sich (vgl. Israel 1999, S.4ff):

1. Überfrachtung des Arbeitsplatzes mit Windows, Icons, Buttons etc.
2. Platzmangel bei wachsender Komplexität der Aufgabe
3. überlastete Menüs oder zu tiefe und somit unübersichtliche Schachtelungen
4. komplizierte Verwaltung mehrerer offener Fenster
5. WIMP-Anwendungen sind in ihren Funktionen und Struktur oft auf den Computer abgestimmt, worunter die Benutzerfreundlichkeit leiden kann
6. Verringerung der Handlungseffizienz durch uniform aussehende oder stereotype Mauspositionierungs- und Klickhandlungen
7. Menüleiste häufig mit Kategorien, die in keinem logischen Zusammenhang stehen
8. Programme unterschiedlicher Hersteller haben nicht zwingend eine einheitliche Namensgebung der Menüpunkte (Widerspruch zur Erwartungskonformität)

Das Hauptproblem von WIMP-Anwendungen ist die visuelle Überfrachtung der Benutzungsschnittstelle bzw. des Arbeitsplatzes (Punkte 1-4). Um die Arbeitsgeschwindigkeit erträglich zu halten, richten sich Benutzer ihre Oberflächen (bewusst oder unbewusst) häufig so ein, dass alle während des Arbeitens nötigen Fenster und Kommandos sofort zugänglich sind. Deshalb sind WIMP-Schnittstellen nach kurzer Arbeitszeit in der Regel mit Windows, Icons, Buttons etc. überfrachtet und es kann zu einer Informationsüberflutung kommen. Mit der wachsenden Komplexität einer zu lösenden Aufgabe wächst auch die Anzahl der nötigen Fenster. Der Benutzer kann

sie maximiert, überlappend oder in Kacheldarstellung (siehe Abbildung 8) anordnen, wobei der Wechsel zwischen den Fenstern teils recht unpraktisch und störend sein kann, da der Benutzer beispielsweise den Überblick über die Zusammenhänge der offenen Fenster verlieren oder ein bestimmtes Fenster nicht sofort finden kann. Bei disziplinierter und konzentrierter Arbeit sollte dieser Zustand vom Benutzer allerdings trotzdem zu bewältigen sein. Dem Benutzer fehlt oft der Platz, seine Arbeit vor sich ausbreiten zu können. Erschwerend kommt hierbei hinzu, dass die Komplexität der Aufgabe meist stark mit der Anzahl der offenen Fenster zusammenhängt – je komplexer die Aufgabe, desto mehr Fenster sind offen. Daraus kann wiederum die Folge gezogen werden, dass der Benutzer bei der Bearbeitung einer komplexen Aufgabe bereits durch diese sehr beansprucht ist und zusätzlich noch durch die Navigation zwischen den Fenstern und deren Inhalten belastet wird. An dieser Stelle wäre es wünschenswert, wenn sich der Benutzer allein auf seine Aufgabe konzentrieren könnte.

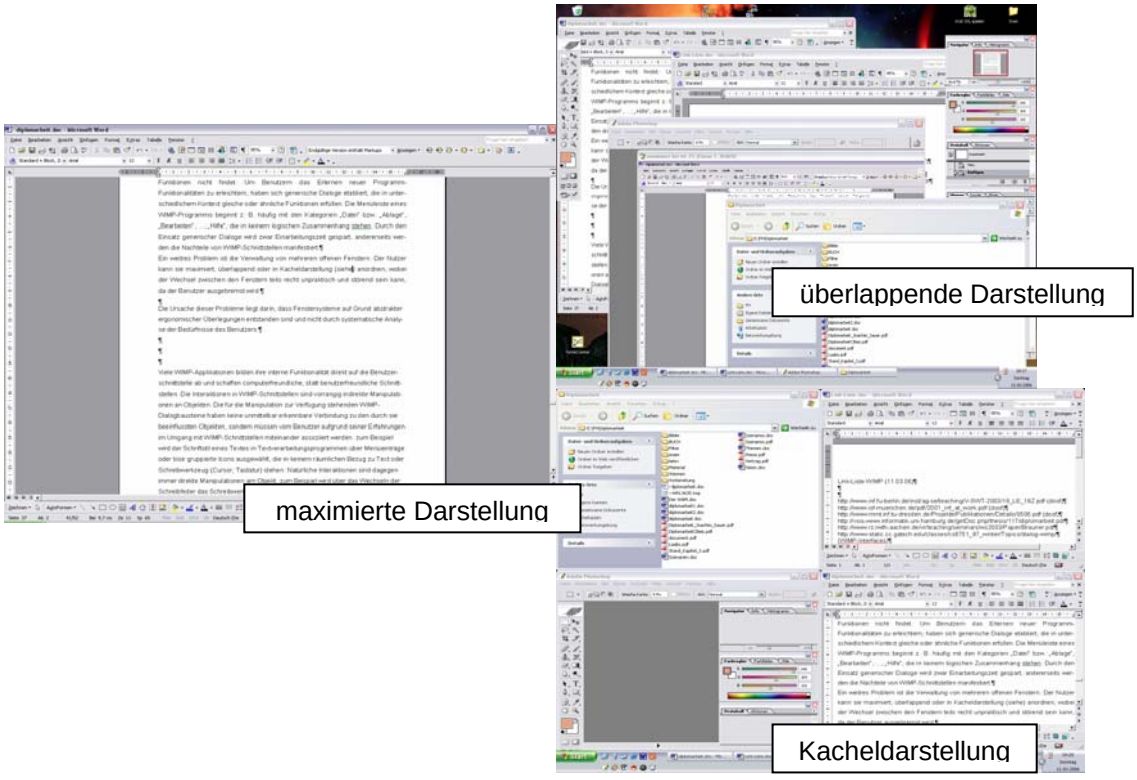

Abbildung 8: Mögliche Fensteranordnungen in WIMP-Benutzerschnittstellen

Wo auf der einen Seite der Nutzer selbst seinen Arbeitsplatz unübersichtlich gestaltet, bieten andererseits diverse Programme so viele Funktionen an, dass die Menüs überlastet oder die Schachtelungen zu tief sind und somit auch unübersichtlich werden (Punkt 3). Das hat zur Folge, dass der Benutzer manche Funktionen nicht findet.

Um Benutzern das Erlernen neuer Programm-Funktionalitäten zu erleichtern, haben sich generische Dialoge etabliert, die in unterschiedlichem Kontext gleiche oder ähnliche Funktionen erfüllen. Die Menüleiste eines WIMP-Programms beginnt z. B. häufig mit den Kategorien „Datei“ bzw. „Ablage“, „Bearbeiten“, ..., „Hilfe“, die in keinem logischen Zusammenhang stehen (Punkt 7). Durch den Einsatz generischer Dialoge wird zwar Einarbeitungszeit gespart, andererseits werden die Nachteile von WIMP-Schnittstellen manifestiert.

Eine weitere Problematik findet sich (siehe Punkt 5) in der Gebrauchstauglichkeit von WIMP-Anwendungen. Nach Israel (S. 11) bilden viele dieser Anwendungen ihre interne Funktionalität direkt auf die Benutzerschnittstelle ab und schaffen computerfreundliche, statt benutzerfreundliche Schnittstellen. Die Interaktionen in WIMP-Schnittstellen sind vorrangig indirekte Manipulationen an Objekten. Die für die Manipulation zur Verfügung stehenden WIMP-Dialogbausteine haben keine unmittelbar erkennbare Verbindung zu den durch sie beeinflussten Objekten, sondern müssen vom Benutzer aufgrund seiner Erfahrungen im Umgang mit WIMP-Schnittstellen miteinander assoziiert werden. Zum Beispiel wird der Schriftstil eines Textes in Textverarbeitungsprogrammen über Menüeinträge oder lose gruppierte Icons ausgewählt, die in keinem räumlichen Bezug zu Text oder Schreibwerkzeug (Cursor, Tastatur) stehen. Natürliche Interaktionen sind dagegen immer direkte Manipulationen am Objekt, zum Beispiel wird über das Wechseln der Schreibfeder das Schreibwerkzeug verändert. Der Benutzer von WIMP-Schnittstellen kann also sein erlerntes Interaktionsverhalten nicht anwenden und wird gezwungen, neue Interaktionssprachen zu erlernen.

Die Ursache dieser Probleme liegt darin, dass Fenstersysteme auf Grund abstrakter ergonomischer Überlegungen entstanden sind und nicht durch systematische Analyse der Bedürfnisse des Benutzers.

Ein genereller Nachteil dieser Programme ist, dass sie durch uniform aussehende Menüsysteme oder stereotype Mauspositionierungs- und Klickhandlungen Handlungseffizienz verschenken, die im Umgang mit realen Werkzeugen erworben wurden (Punkt 6). Auch haben Programme unterschiedlicher Hersteller nicht zwingend eine einheitliche Namensgebung der Menüpunkte, was beim Benutzer zu Verwirrung führen kann (Punkt 8).

Zusammenfassend kann festgestellt werden, dass bei der Benutzung von WIMP-Interfaces aus der Vielzahl der motorischen und sensorischen Fähigkeiten des Men-

schen nur der visuelle Sinn und die Motorik der Hand nennenswert zum Einsatz kommen. Die abstrakte Funktions- und Kommandostruktur der Maschine wird häufig in der WIMP-Schnittstelle sichtbar, die kognitiven Fähigkeiten des Menschen werden selten bewusst eingeplant, effiziente sensomotorische Prozesse sind kaum möglich (Israel 1999, S. 10 ff).

Neben den Nachteilen für die Benutzer haben WIMP-Programme noch einen weiteren großen Nachteil für das Konferenzsystem:
Sie sind für die Verwendung an einem PC gedacht. Ein Benutzer sitzt vor dem PC und bedient die Anwendung. Dabei ist es natürlich möglich, dass der PC an ein Netzwerk angebunden ist und auch auf andere Rechner wie zum Beispiel einen Server zwecks Filetransfer zugreift oder auch dass der Desktop über einen Beamer auf eine Leinwand projiziert wird. Aber die Bearbeitung eines Dokuments von mehreren Benutzern gleichzeitig von mehreren PCs aus ist nicht vorgesehen. Für das Konferenzsystem ist diese Funktionalität allerdings eine Kernfunktion, die jede Anwendung, die eingebunden wird, unterstützen sollte, sonst können die Vorteile des Konferenzraumes kaum genutzt werden.

Um diese Kernfunktion zu gewährleisten, gibt es die Möglichkeit, einen Wrapper zu schreiben, der der Anwendung die erforderlichen Funktionen zur Kommunikation mit dem Konferenzsystem zur Verfügung stellt und einen Multi-User-Betrieb erlaubt. Bedingung dafür ist, dass die Anwendung eine Schnittstelle besitzen muss, an der ein solcher Wrapper angesetzt werden kann. Der Nachteil dieser Lösung liegt darin, dass die Problematiken der WIMP-Programme bestehen bleiben und dass diese isolierten Desktopanwendungen weiterhin nur eine begrenzte Integrationsfähigkeit haben. Die Isolation der Desktopanwendungen entsteht dadurch, dass eine Datei durch ihre Endung fest mit einer Anwendung verknüpft wird. Der Bezug, den die Dateien untereinander haben können, geht verloren bzw. wird gar nicht berücksichtigt. Im Fall des Konferenzsystems wird es allerdings vorkommen, dass Dateien von unterschiedlichem Typ miteinander in Beziehung stehen. So liegt zum Beispiel einer Programmkomponente ein Use-Case zugrunde. Die Java-Komponente ist dokumentiert, wodurch eine Beziehung zur Dokumentation entsteht. Wiederum gab es vorher Diskussionen zur Realisierung der Komponente, in der Entwurfsentscheidungen getroffen worden sind. Zu diesen Diskussionen gibt es Protokolle und ggf. andere Dokumente.

Möchte ein Teammitglied nun alle Dokumente, die mit der Java-Komponente zu tun haben, angezeigt bekommen, bietet ihm ein herkömmliches WIMP-System diese Funktion nicht.
Um die meisten Probleme der WIMP-Programme zu umgehen und die Isolation aufzuheben, scheint es erforderlich, sämtliche Anwendungen, die in dem Konferenzraum zur Verfügung stehen sollen, selbst zu entwerfen und zu schreiben. Dabei sollten die Entwickler der Anwendungen einen Schwerpunkt auf die Gebrauchstauglichkeit legen und die Fehler, die bei existierenden Systemen vorliegen, nicht wiederholen. Auch sollte ein Ansatz gefunden werden, der die Artefakte in den Vordergrund stellt und nicht die Anwendungen, so dass es eine Möglichkeit gibt, die Beziehungen der Artefakte darzustellen und angezeigt zu bekommen. Im Idealfall sollten die Anwendungen nicht mehr unabhängig voneinander stehen, sondern als eine Anwendung zusammengefasst sein, die bei Bedarf um weitere Funktionen ergänzt werden kann.
Um all diese Forderungen umzusetzen, sollten die Daten zusätzliche Informationen erhalten. Bei einem herkömmlichen Dateisystem ist die Datei durch ihre Endung mit einer Anwendung verknüpft. Die Datei selber enthält kaum weitere Informationen außer dem Erstellungsdatum, dem Autor und ggf. einer Zugriffsbeschränkung. Hätten die Dateien zusätzliche Informationen, zum Beispiel mit welchen Dokumenten sie in Beziehung stehen, mit welchen Anwendungen sie bearbeitet werden können und wer auf sie zugreifen darf, könnten die Dateien in ihrem jeweiligen Kontext angezeigt werden. Dieses würde eine aufwändige Suche und ein unübersichtliches Fenstermanagement – für jedes Dokument öffnet sich ein eigenes Fenster – ersparen. Im Folgenden wird eine Lösung für die beschriebene Situation erläutert.

3.2.3 Grapheneditor

Die Idee ist hierbei, die Daten als Netz darzustellen. Ein Netz besteht aus Knoten, die über Kanten verbunden sind. Die Daten werden in diesem Fall zu Knoten und die Beziehungen zwischen den Daten sind die Kanten, die durch Links realisiert werden. Die Knoten enthalten sämtliche Informationen darüber, wie sie zu behandeln sind und was mit ihnen gemacht werden darf. Somit ist ein Knoten also ein Dokument bzw. Artefakt, das zusätzliche Informationen (Metadaten) enthält – wie am Ende des vorigen Kapitels gefordert. Die Knoten haben durchweg eine einheitliche Struktur, die

in Abbildung 9 dargestellt ist. Die Attribute variieren je nach Knotentyp (vgl. Tabelle 2 im Anhang).

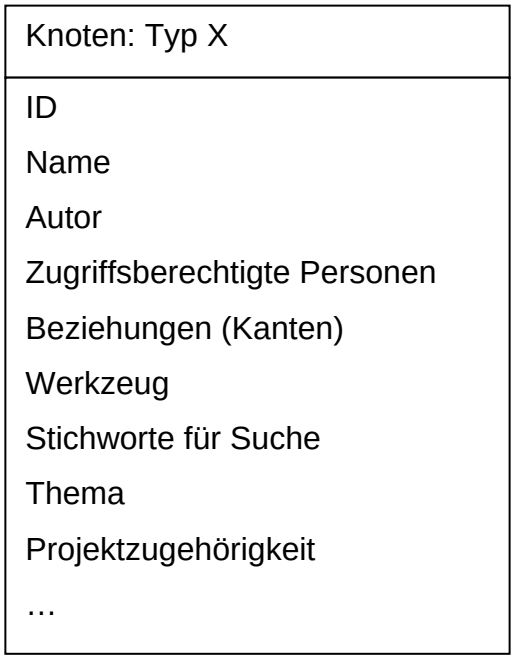

Abbildung 9: Struktur eines Knotens

Dennoch unterscheiden sich die Knoten anhand des zugrunde liegenden Datentyps. So wird zum Beispiel ein Textdokument, ein UML-Diagramm oder auch eine Anwendung ein Knoten sein. Da diese Dokumente nicht einheitlich aufgebaut sind, müssen die Knoten an die jeweiligen Dokumenttypen angepasst werden. Also muss für jeden Dokumenttyp auch ein Knotentyp bereitgestellt werden.

Jeder Knoten enthält auch die Information, welche Personen auf ihn zugreifen dürfen, wodurch die Zugriffsrechte geregelt werden können. Auf diese Weise entsteht auch eine Schichtung des Netzes in Zugriffsebenen. So kann beispielsweise der potentielle Benutzer nur auf für ihn bestimmte Dokumente zugreifen und die Entwickler wiederum haben einen erweiterten Zugriffsraum. Auf diese Weise wird der Zugriff auf die Dokumente abhängig davon geregelt, welche Rolle die Person im Team einnimmt. An den Grenzen der Zugriffsebenen entstehen Autor-Kritiker-Zyklen. Diese Zyklen sind nach diesem Schema unumgänglich, da die Mitglieder der einen Benutzergruppe die Daten der anderen Gruppe evtl. einsehen aber nicht verändern können. Also müssen Änderungswünsche, Lob und Kritik diskutiert, aufgenommen und ins System eingepflegt werden. Abbildung 10 visualisiert die Idee des Grapheneditors in vereinfachter Form mit den Zugriffsebenen. Zusätzlich zu den durch die Struk-

tur erzeugten Autor-Kritiker-Zyklen sollten innerhalb eines Zugriffsbereichs – also im Team – auch Gespräche und Rückkopplungen stattfinden.

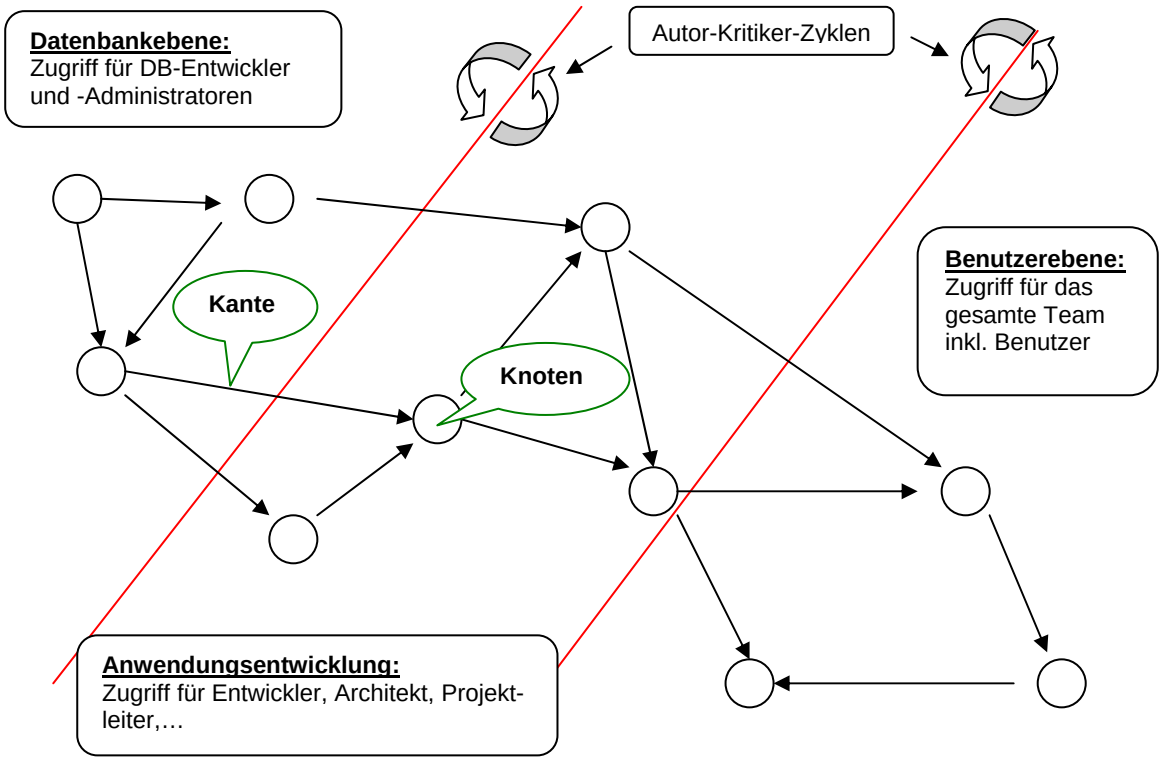

Abbildung 10: Netz mit Zugriffsebenen

Da diese Art der Dateiverwaltung von den bestehenden WIMP-Anwendungen nicht zur Verfügung gestellt wird, müsste entweder eine eigene Anwendung für den Grapheneditor oder eine Umgebung geschrieben werden, in die bestehende WIMP-Anwendungen eingebettet werden können. Letztendlich wird es Ziel dieses Projekts sein, eine eigene Konferenzanwendung zu schaffen. Das bedeutet, es soll eine einzige WIMP-Anwendung geben, die sämtliche Werkzeuge zur Bearbeitung der jeweiligen Knoten zur Verfügung stellt und die auf dem hier beschriebenen Dateisystem basiert. Die grobe Struktur dieser Anwendung ist in Abbildung 11 dargestellt.

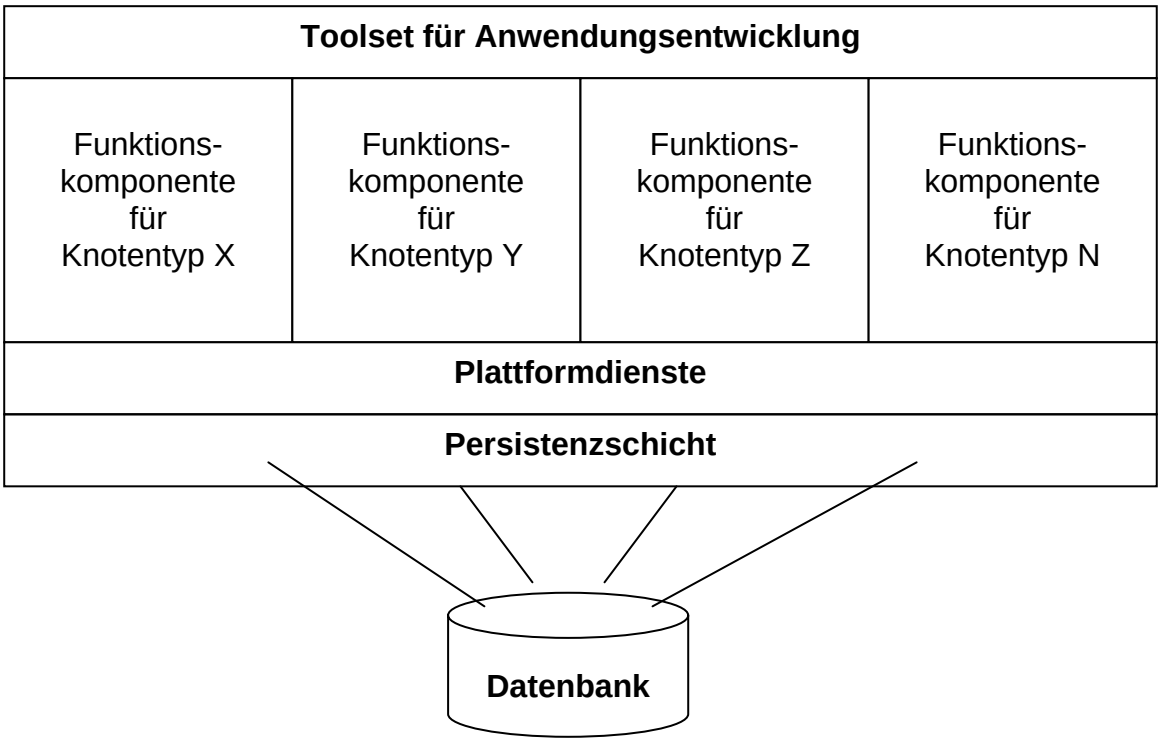

Abbildung 11: Vision der Konferenzanwendung

Die Funktionskomponenten dienen der Bearbeitung und Betrachtung der jeweiligen Knotentypen bzw. Dokumenttypen, was in diesem Fall äquivalente Bezeichnungen sind. Die Plattformdienste bieten Funktionen, die unabhängig von den Werkzeugen Dienste (wie zum Beispiel das Veröffentlichen von Artefakten am großen Display oder der Im- und Export von Artefakten) zur Verfügung stellen. In der Persistenzschicht ist hinterlegt, wie und wo die Dokumente in der Datenbank abgelegt werden.

Der in Abbildung 11 dargestellte Aufbau der Anwendung ähnelt der von Shaw und Garlan beschriebenen Repository-Architecture (Shaw u. Garlan 1996, S. 69 ff.). Die Idee bei dieser Architektur ist, die Daten in einer oder mehreren Datenbanken bereit zu halten, auf die die Tools, die zur Bearbeitung der Daten bereitgestellt werden, zugreifen können. Dabei gibt es unterschiedliche Ansätze. Zum einen greifen voneinander unabhängige Tools auf eine gemeinsame Datenbasis zu, wobei die Tools vom Benutzer aus gesteuert und ausgewählt werden. Zum anderen gibt es den Entwurf, dass ein Datenmanager die Auswahl für den Benutzer trifft. Wenn der Benutzer ein Dokument bearbeiten möchte, kann er dies tun, ohne sich um die Auswahl des Tools zu kümmern. Er bekommt automatisch die richtigen Funktionen für den Dokumenttyp vom System zur Verfügung gestellt.

Übertragen auf den Grapheneditor liegen die Knoten in einer gemeinsamen Datenbank, auf die der Benutzer über eine Benutzungsschnittstelle zugreifen kann. In einer ersten Entwicklungsstufe des Konferenzsystems werden bestehende WIMP-Anwendungen für die kollaborative Arbeit angepasst bzw. in eine Anwendung eingebettet. Diese Lösung entspricht in etwas dem Modell, dass es unterschiedliche Tools gibt, die für die Bearbeitung der Daten zur Verfügung stehen. Diese Tools werden unter einer Oberfläche vereint und an die unterschiedlichen Knotentypen gebunden. Auf diese Weise besitzt der Knoten die Information, mit welcher Anwendung er zu bearbeiten ist. Der Knoten wird dadurch automatisch in der richtigen Anwendung geöffnet, wenn der Benutzer dies wünscht.
Diese erste Entwicklungsstufe hat den Nachteil, dass die Grenzen der Anwendungen nach wie vor erhalten bleiben, was eigentlich nicht gewünscht ist. Ziel der Konferenzanwendung ist es, jeden Knoten unter der gleichen Oberfläche zu bearbeiten d. h. mit der gleichen Anwendung. Die Bearbeitung der unterschiedlichen Knoten geschieht mit augenscheinlich den gleichen Werkzeugen. Welche Implementierung des Werkzeugs aber benutzt wird, ist vom Knotentyp abhängig und wird vom System automatisch gewählt. Diese Vision entspricht der zweiten vorgestellten Variante der Repository-Architecture und soll in weiteren Entwicklungsstufen des Grapheneditors, der im weiteren Verlauf den Namen „kora nodes“ bekommt, nach und nach umgesetzt werden.
Aufbauend auf diesem Stufenkonzept wird in dieser Arbeit hauptsächlich die erste Entwicklungsstufe ausgebaut und prototypisch realisiert werden. Im weiteren Verlauf werden die Knotentypen und Werkzeuge zu deren Bearbeitung bestimmt und ein Prototyp erstellt, der die Vision für die erste Entwicklungsstufe darstellt. Auf den Ergebnissen basierend können in weiteren Arbeiten Funktionskomponenten bzw. Werkzeugkästen zur Bearbeitung der Knoten entstehen.
Um auf diesem Wege weiterzuarbeiten, müssen zunächst die Anforderungen an diese Anwendungen konkretisiert und aufgeschrieben werden.

3.3 Anforderungen

Das Konferenzsystem soll Diskussionsprozesse unterstützen und sie effizienter machen. Daraus ergibt sich die allgemeine Anforderung, dass das System sehr einfach

und intuitiv zu bedienen sein muss, ohne dass für die Benutzer lange Einarbeitungszeiten entstehen. Somit sind eine einfache und übersichtliche Benutzungsschnittstelle und die Entwicklung eines entsprechenden Benutzungsmodells von großer Bedeutung. In diesem Zusammenhang sollte der Benutzer eine möglichst vertraute Umgebung vorfinden, weswegen auch dieses System eine WIMP-Anwendung, die auf einem Fenstermanagementsystem aufsetzt, sein sollte. Allerdings sollen die Nachteile, die WIMP-Anwendungen meistens mit sich bringen (wie in Kapitel 3.2.2 beschrieben), nach Möglichkeit weitestgehend vermieden werden.

Nach der Analyse der Arbeiten, die in dem Konferenzraum durchgeführt werden sollen, braucht das System auf jede Projektsituation bzw. für jeden Workflow abgestimmte Funktionalitäten und Ansichten auf die Daten. So sollte es später möglich sein, dass ein Themengebiet, wie zum Beispiel der Entwurf der Architektur, ausgewählt wird und als Folge der dafür benötigte Kontext offensichtlich angezeigt wird. Natürlich muss der Nutzer auch dort die Möglichkeit haben, ohne großen Aufwand auf sämtliche für ihn zugängliche Daten zurückgreifen zu können. Folglich sollte während der Entwicklung der unterschiedlichen Ansichten darüber reflektiert werden, welche Daten zu welchem Zeitpunkt benötigt werden.

Zusätzlich sollte die Anwendung über einen Browser und eine Internetanbindung verfügen, damit – falls nötig – Zusatzinformationen aus dem Internet besorgt werden können.

Ein weiterer wichtiger Aspekt ist die Suche nach Dokumenten. Es ist durchaus vorstellbar, dass ein Benutzer weiß, dass er zu einem bestimmten Zeitpunkt mit ihm bekannten Personen ein Dokument zu einem bestimmten Thema erstellt hat, aber den Namen des Dokuments und den Speicherort vergessen hat. Gäbe es eine Suchfunktion, bei der der Benutzer die ihm bekannten Kriterien – in diesem Fall Erstellungsdatum, Autoren und Stichwort – eingibt, würde ihm das gewiss eine Zeitersparnis bringen und eine aufwendige Suche ersparen. Somit braucht das Konferenzsystem eine ausgefeilte kriterienbasierte Suchfunktion.

Ergänzend oder auch an Stelle der Suchfunktion kann ein Dokumentenfilter implementiert werden. Der Filter hat den Vorteil, dass die Struktur der Daten bleibt. Es werden lediglich die Artefakte ausgeblendet, auf die der Filter nicht zutrifft. Als Ergebnis einer Suche hingegen wird dem Benutzer eine Liste mit „Treffern“ angezeigt, bei der die Struktur der Daten verloren geht.

Zur Ausstattung des Konferenzraums (siehe Abbildung 12) gehören mehrere feste PCs und Dockingstations, an denen Notebooks angeschlossen werden können. Mobiltelefone und PDAs sollen auch mit dem System kommunizieren können. Einen weiteren zentralen Bestandteil stellen ein oder mehrere große Displays dar, hinter denen jeweils ein eigener PC steht. Diese Displays sollen touchsensitiv sein. Das ganze System ist miteinander vernetzt. Die Anforderung an das System besteht darin, dass ein Dokument auf allen Displays – sowohl auf den großen als auch auf den anderen Geräten – angezeigt werden soll. Auf den großen Displays soll eine gemeinsame Bearbeitung von allen Arbeitsplätzen aus möglich sein. Auch soll jeder Benutzer einem oder mehreren Benutzern Dokumente per „drag & drop“ auf ein Gerät seiner Wahl oder auch auf alle Geräte zukommen lassen können.

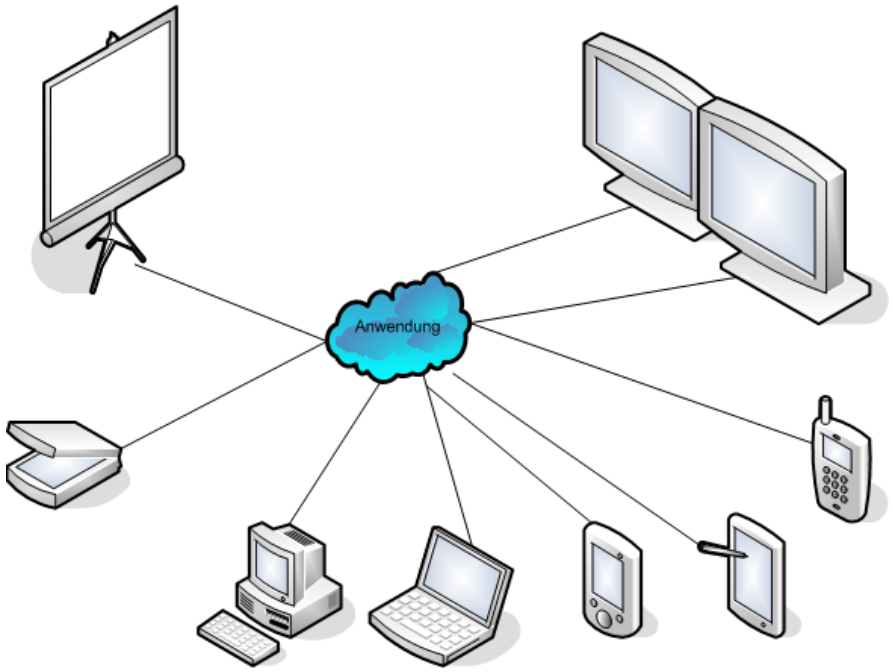

Abbildung 12: Ausstattung des Konferenzraumes

An dieser Stelle bleibt zu überlegen, ob es sinnvoll wäre, dass sämtlich Dokumente, egal von welchem Typ sie sind, auf den Geräten im Konferenzraum angezeigt werden können. Die Tatsache, dass ein Dokument angezeigt wird, könnte beim Benutzer den Eindruck erwecken, dass auch diese Dokumente zu bearbeiten seien. Wenn das nicht funktioniert, könnte der Benutzer in seinen Erwartungen enttäuscht werden, was gegen den Grundsatz der Erwartungskonformität verstößt und eine Unregelmäßigkeit im Benutzungsmodell wäre. Andererseits könnte es während einer Diskussion hilfreich sein, ein Dokument anzuzeigen, auch wenn es nicht zu bearbeiten ist, um

eine Situation o. ä. zu verdeutlichen. Werden also jegliche Dokumente zur Anzeige zugelassen, sollte der Nutzer darauf hingewiesen werden, dass das angezeigte Dokument nicht manipulierbar ist.

Der Wunsch vieler Benutzer wird sein, die erarbeiteten Dokumente aus dem System zu exportieren, um sie später außerhalb des Konferenzraums zu nutzen und vorbereitete Dokumente zu importieren, um diese während einer Sitzung zu diskutieren. Hierfür müssen geeignete Funktionen zur Verfügung stehen. Wird ein bereits bestehendes Dokument wieder importiert, wird es zum Konflikt kommen, da das vorhandene Dokument von dem neuen überschrieben werden könnte. Um diese Art von Konflikten zu vermeiden, benötigt das System eine Versionskontrolle, mit deren Hilfe der Benutzer auf den Konflikt aufmerksam und auf die Unterschiede zwischen den Dokumenten hingewiesen wird.

Auch sollte das System über eine Möglichkeit verfügen, Dokumente auszudrucken und einzuscannen, da in manchen Fällen die Ansicht auf Papier angenehmer ist oder eine Zeichnung, die auf Papier existiert, ins System übernommen werden soll.

Zusätzlich soll dieser Konferenzraum auch von anderen Benutzergruppen als Softwareentwicklern genutzt werden. In diesem Zusammenhang schreibt Lars Burfeindt eine Anforderungsanalyse für Stadtplanung (Burfeindt 2006). Für eine solche Benutzergruppe müssen andere Funktionskomponenten als für Softwareentwickler zur Verfügung stehen. Somit braucht das System eine Funktion, die für die entsprechende Benutzergruppe die richtigen Funktionskomponenten zur Verfügung stellt.

Die bis jetzt beschriebenen Anforderungen sind wahrscheinlich für alle Benutzergruppen gültig, da sie die Basisfunktionen, die das System erfüllen muss, darstellen. Außerdem ist zu überlegen, welche Funktionskomponenten für die Gruppe der Softwareentwickler benötigt werden.

Wie bereits in Kapitel 3.2 erwähnt, liegen die Gemeinsamkeiten der Szenarien darin, dass…

- …Beratung und Diskussion über ein Thema bzw. Artefakt stattfindet.
- …Präsentation gehalten werden.
- …Brainstorming stattfindet.
- …Protokolle geschrieben werden.
- …Zugriffsbeschränkungen bestehen sollten.
- …Zeichnungen und andere Artefakte erstellt werden.

Hinzu kommen die situationsspezifischen Abläufe wie

- die Aufwandsschätzung
- der Entwurf der Architektur
- der Entwurf von Use-Cases
- die Projektplanung
- die Implementierung
- und das Erstellen von Schulungsunterlagen.

Für die hier aufgelisteten Abläufe sollten Funktionskomponenten bzw. Werkzeuge und Materialien bereitgestellt werden, die die allgemeinen vorher beschriebenen Anforderungen erfüllen. So wird zum Beispiel zum Präsentieren ein Programm wie zum Beispiel PowerPoint und zur Implementierung ein Programm wie Eclipse benötigt. Zum Entwerfen der Architektur und Use-Cases kann eine UML-Anwendung dienen. Zum Erstellen der Schulungsunterlagen werden ggf. mehrere Funktionen benötigt wie zum Beispiel PowerPoint für Präsentationen, Word für die Erstellung eines Benutzerhandbuchs, ein Konvertierungstool von Word zu PDF zum gleichen Zweck und eine Komponente zur Erstellung eines Hilfesystems zur Anwendung. Die hier aufgeführten oder andere Programme mit gleicher Funktionalität müssen also ins Konferenzsystem als Funktionskomponenten eingebunden werden. Hierbei sollten gängige Programme bzw. Formate benutzt werden, damit die Daten sowohl im Raum als auch außerhalb des Raumes uneingeschränkt nutzbar sind.
Der Workflow Projektplanung fordert eine weitere Überlegung, da die Projektplanung auch die Zeitplanung des Projekts umfasst. Diese wird nicht ausschließlich im Konferenzraum gemacht. So bleibt zu überlegen, wenn das Konferenzsystem im Konferenzraum einer Firma installiert ist, wie der Zugriff auf das System von außerhalb des Raumes geregelt werden kann und sollte. Es macht keinen Sinn, zwei Projektplanungstools zugleich zu haben. Also muss das Planungstool der Firma in das Konferenzsystem integriert werden oder umgekehrt. Die Daten müssen sowohl im Konferenzraum als auch am PC des Projektleiters oder einer anderen dafür verantwortlichen Person zur Verfügung stehen. Ein weiteres damit zusammenhängendes Problem könnte sein, dass die Firma bereits ein Projektplanungstool nutzt, das dann ins Konferenzsystem integriert werden muss. Dieses kann auch für andere Funktionskomponenten gelten wie zum Beispiel die UML-Anwendung, mit der die Entwickler oder der Architekt Vorentwürfe am PC im Büro machen. In diesem Fall ist das Prob-

lem der Synchronisation nicht gegeben, aber die Daten müssen von der Struktur her zum Dateimanagement des Konferenzsystems passen. Dieses gilt natürlich für sämtliche Dokumente bzw. Funktionskomponenten, die außerhalb des Raumes benutzt werden. Die Lösung dieses nicht unrelevanten Problems sollte nicht vernachlässigt werden und bedarf im weiteren Projektverlauf einiger Überlegung.
Eine weitere wichtige Anforderung ist die der Durchgängigkeit. Sie bedeutet, dass die Modelle, die in einem Entwicklungsabschnitt entstehen, in den folgenden weiterhin genutzt werden können. Die Modelle verändern sich allerdings während der Entwicklungsabschnitte, da einige Informationen im weiteren Verlauf nicht mehr relevant sind, andere Informationen kommen dafür hinzu. Diese Modelle müssen zueinander kompatibel gehalten werden. Direkt gekoppelt mit dem Verlangen nach Durchgängigkeit ist Forderung nach Harmonisierung der einzelnen Anwendungen. Es wäre vorstellbar, dass ein Modell in den unterschiedlichen Entwicklungsabschnitten in einer anderen Form dargestellt wird. Diese andere Darstellung wird mit einer anderen Anwendung erzeugt. Wenn die unterschiedlichen Artefakte nicht entsprechend markiert oder abgespeichert werden, könnte der Zusammenhang zwischen den Artefakten verloren gehen. Das wiederum würde die Durchgängigkeit stören.
Als Zusammenfassung und zur Veranschaulichung folgt hier eine Auflistung der Anforderungen (vgl. Tabelle 1: Auflistung der Anforderungen im Anhang):

1. hohe Gebrauchstauglichkeit
2. Datenansicht abhängig vom Szenario
3. kriterienbasierte Suche bzw. Filterfunktion
4. gemeinsame Bearbeitung von Dokumenten auf den großen Displays
5. Austausch von Dateien mit sämtlichen Geräten im Raum
6. Funktionskomponenten abhängig von Benutzergruppe
7. Bereitstellung der benötigten Funktionskomponenten
8. Einbindung von externen Anwendungen (Beispiel Projektplanung)
9. Anmeldung/ Registrierung der Nutzer
10. Im- und Export von Daten ins bzw. aus dem System
11. aktueller Projektstand im System
12. Durchgängigkeit
13. Harmonisierung der Anwendungen

3.4 Ergebnis

In diesem Kapitel wurden die Grundvoraussetzungen geschaffen, um mit der Erstellung der benötigten Werkzeuge für das Konferenzsystem zu beginnen. Hierzu werden zunächst die Knoten und ihre Abhängigkeiten erarbeitet. Im Anschluss werden die Architektur der WIMP-Anwendung mit den benötigten Funktionskomponenten sowie ein Konzept zu deren Erstellung entworfen. Das Resultat des nächsten Kapitels soll ein Prototyp bzw. eine GUI sein, die den Aufbau und die Struktur der Daten erkennen lässt.

4 kora nodes

In diesem Teil der Arbeit werden die Werkzeuge und Materialien gemäß den vorher erarbeiteten Anforderungen, Projektsituationen und Szenarien erarbeitet und es wird ein Prototyp von kora nodes entstehen, der die erste Entwicklungsstufe widerspiegelt. Um zu erkennen, welche Werkzeuge und Materialien tatsächlich benötigt werden, müssen die Artefakte von ihrer Art und ihrem Typ bzw. Verwendungszweck ermittelt werden. Die Informationen hierzu stammen aus Interviews von Mitarbeitern von Signal Iduna und IBM. Die Bestimmung der Knotentypen für den Graphen editor orientiert sich u. a. an den in einem Softwareprojekt vorkommenden Artefakten. Die Knotentypen werden zum einen Materialien wie zum Beispiel Formulare oder Protokolle sein. Zum anderen werden auch die Werkzeuge, mit denen die Materialien bearbeitet werden können, Knotentypen darstellen. Des Weiteren gilt es zu analysieren, wie die Knotentypen miteinander in Beziehung stehen. Die Beziehungen werden in kora nodes die Kanten sein, mit denen die Knoten verbunden sind.

4.1 Knotentypen

Knotentypen sollten alle Objekte werden, die mit Artefakten in Beziehung stehen. Auf diese Weise kann der Graph, der kora nodes zugrunde liegen wird, vollständig dargestellt werden. Somit sollten auf jeden Fall alle Artefakte, die Anwendungen bzw. Werkzeuge und die Projektmitarbeiter als Knoten dargestellt werden.
Die Mitarbeiter als Knoten darzustellen, macht insofern Sinn, dass jeder Mitarbeiter mit allen Artefakten verlinkt sein wird, die er bearbeitet bzw. erstellt hat. Das könnte die Suche nach Artefakten im späteren Einsatz vereinfachen. Auch kann dadurch ein Rechtemodell unterstützt werden, da jeder Mitarbeiter eine Rolle hat (vgl. Kapitel 2.1 Rollen). Auf diese Weise ist festgelegt, auf welche Bereiche im Projekt der Mitarbeiter zugreifen darf und auf welche nicht. Somit wird es auf jeden Fall einen Knotentyp „Mitarbeiter“ geben, der ein Attribut „Rolle“ und Links auf die von ihm erstellten Artefakte hat.
Für die Artefakte müssen je nach Typ auch unterschiedliche Knotentypen entstehen, da sie unterschiedliche Dateitypen sind und somit vorerst mit unterschiedlichen An-

wendungen bearbeitet werden. Auch bei einem Universal-Werkzeug, das alle Artefakte bearbeiten kann, muss der Typ des Artefakts an das Werkzeug übergeben werden, damit die entsprechenden Methoden zur Bearbeitung angesprochen werden können. Deshalb wird im Folgenden Kapitel erarbeitet, welche Artefakte typischer Weise in einem Softwareprojekt vorkommen.

Zusätzlich sollten Knoten für Behälter geschaffen werden. Diese dienen der Ablage von Artefakten zum Beispiel nach Knotentyp oder Thematik sortiert. Auch kann es weitere geben – zum Beispiel private Artefakte, die die Mitarbeiter als Grundlage zu einer Konferenz mitbringen.

Abbildung 13 stellt einen Auszug von denkbaren Knotentypen dar. Die Abhängigkeiten sind hier als Vererbung zu betrachten, d. h. es gibt Knoten, die spezialisiert sind in Material (Artefakt), Werkzeug, Mitarbeiter und Behälter. Jeder dieser Knotentypen hat bestimmte Eigenschaften, die er speziellere Ausprägungen seiner Art vererbt. So ist eine Architektur zum Beispiel eine Spezialisierung eines UML-Knotens.

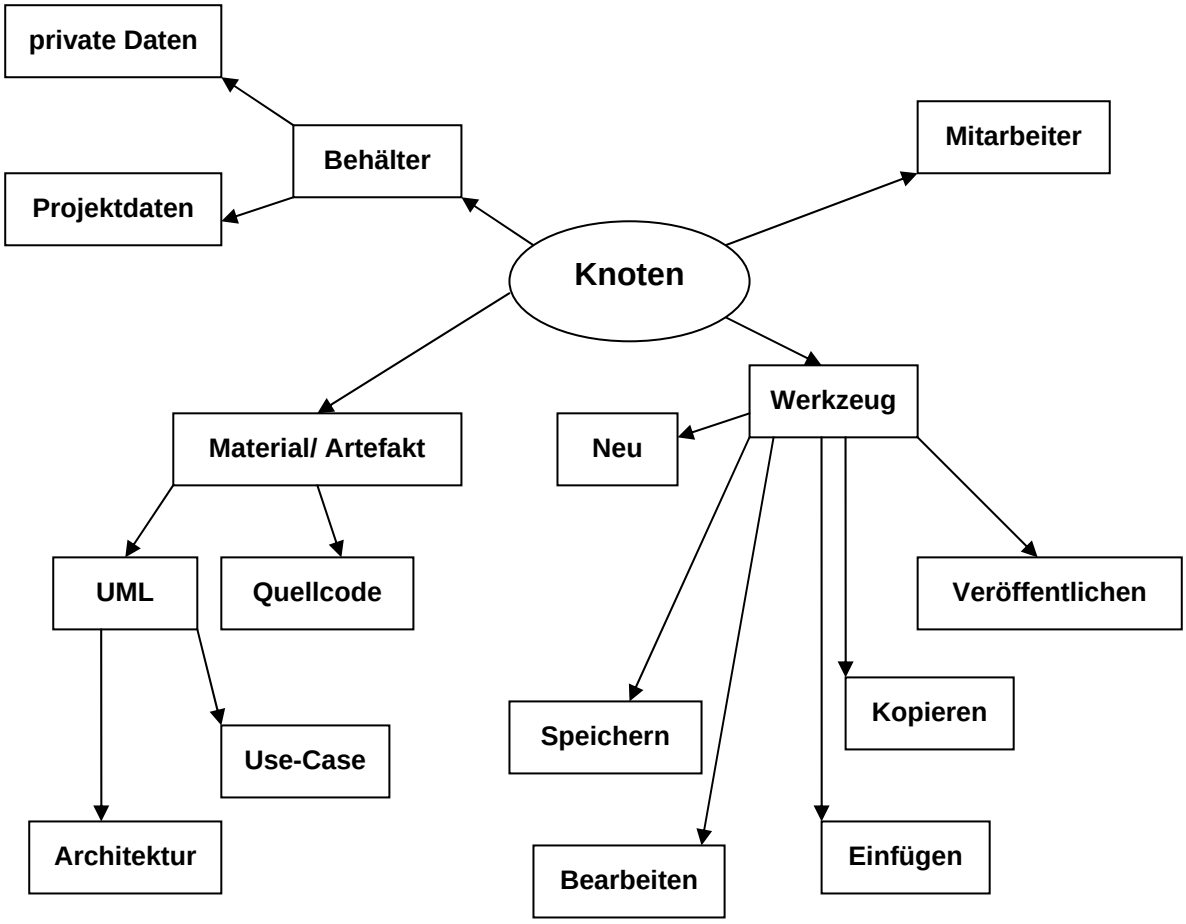

Abbildung 13: Knotentypen

Das Modell in Abbildung 13 stellt lediglich die Basis für die Implementierung dar, nicht aber die Abhängigkeiten zwischen den einzelnen Knotentypen wie beispielsweise zwischen Mitarbeiter und Materialien. Diese Abhängigkeiten könnten auch in Abbildung 13 eingezeichnet werden, würden die Grafik allerdings unübersichtlich machen.

4.1.1 Artefakte

Ein Artefakt ist ein Element im Komponentendiagramm, das eine physische Informationseinheit darstellt, zum Beispiel ein Modell, eine Quellcode-Datei, eine ausführbare Binärdatei, eine Archivdatei (Balzert 2005, S. 527). Artefakte entstehen während jeder Projektsituation und jedem Szenario. Der Typ des Artefakts ist meist unabhängig von der Projektsituation, wobei bestimmte Typen in einigen Situationen häufiger auftauchen. So wird es zum Beispiel während der Implementierung viele Quellcode-Dateien geben und während der Projektplanung vermehrt Komponentenmodelle und Use-Cases. Somit sollten die Artefakte zunächst unabhängig von den Projektsituationen betrachtet werden, da eine genaue Zuordnung kaum möglich ist.
Nach oder während den meisten Meetings werden Protokolle geschrieben. Dies sind in der Regel Textdokumente. Sie werden nicht nur für Protokolle verwendet, sondern auch für die Erstellung von Fragebögen, Interviews, Gesprächsleitfäden oder zum Notieren von Arbeitsabläufen und Szenarios. Damit ist die Verwendung von Textdokumenten noch nicht ausgeschöpft. Es wird allerdings verdeutlicht, dass der Knotentyp für Textdokumente erforderlich ist. Abgeleitet von dem Knotentyp „Text“ sollte es Knotentypen für die Spezialisierungen geben, um die Textdokumente später nach ihrem Inhalt einordnen zu können.
Ein weiterer wichtiger Bestandteil in einem Softwareprojekt sind UML-Diagramme. Mit UML können Use-Cases, die Architektur, Komponentenmodelle, Daten- und Klassenmodelle und viele weitere Diagramme und Schaubilder zur Projektplanung erstellt werden. Folglich wird ein Knotentyp für UML-Dokumente benötigt. Diese Dokumente sollten für die bessere Übersicht und Struktur in ihre Themengebiete unterteilt werden. Eine denkbare Aufteilung wäre Knotentypen für die Bereiche Architektur, Use-Cases, Workflows und sonstige UML-Diagramme.
Bei der Implementierung entsteht Quellcode. Dieser kann in einem Texteditor angezeigt werden. Die meisten Texteditoren unterstützen allerdings kein Syntax-Highlighting, wodurch der Code schwieriger zu lesen ist. Das Lesen des Quellcodes

wird in dem Moment relevant, wenn er während einer Konferenz betrachtet werden soll, was bei einer gemeinsamen komplexen Fehlersuche zum Beispiel in einer Krisensituation der Fall sein könnte. Ansonsten finden Betrachtungen von Code eher am Arbeitsplatz im Büro statt. Zusätzlich muss der Quellcode als Knotentyp vorhanden sein, da er u. a. in Beziehung mit seiner Dokumentation, diversen Use-Cases und Komponentendiagrammen steht.
Ein häufiger Bestandteil in Konferenzen sind Präsentationen. Sei es, dass dem Kunden ein Konzept präsentiert wird, ein Projektleiter über den Stand des Projekts berichtet und dazu eine Präsentation vorbereitet hat oder ein Vertriebler eines Lieferanten sein Produkt vorstellt. Diese Präsentationen müssen in kora nodes eingebunden werden und brauchen somit auch einen Knotentyp.
Da in dem Konferenzraum auch ein digitales Whiteboard vorhanden sein soll, an dem auch Zeichnungen erstellt werden können, müssen Bilder eingebunden werden können. Auch soll der Raum über einen Scanner verfügen, um Graphiken zu digitalisieren. Auf diese Weise kann eine Abbildung aus einem Buch oder einer Zeitschrift allen Konferenzteilnehmern auf einfachem Wege gezeigt werden. Somit wird ein Knotentyp für Bilder benötigt, der die gängigen Bildformate unterstützt.
Des Weiteren könnte es nützlich sein, einen Knotentyp für Tabellenkalkulationen einzubinden, da diese einen Dokumenttyp darstellen, der oft benutzt wird.
Da Situationen entstehen können, in ein Artefakt angezeigt werden soll, dessen Typ kora nodes unbekannt ist, sollte dafür ein Knotentyp „Sonstige“ eingebunden werden. Dieser hat die Eigenschaft, die Dokumente anzuzeigen, bietet aber keine Möglichkeit zur Bearbeitung der Knoten.
Nunmehr stehen die wichtigsten Knotentypen für Artefakte fest. Da in einem Unternehmen allerdings auch andere Artefakte auftreten können, sollte die Möglichkeit bestehen, neue Knotentypen zu erzeugen, die dann von kora nodes interpretiert werden können. Auch sollte im weiteren Verlauf des Projekts beobachtet werden, ob weitere Spezialisierungen der Knotentypen, so wie hier schon in den Fällen der UML-Diagramme und Textdokumente beschrieben und in Abbildung 13 dargestellt, notwendig sind.

Nach den bisherigen Überlegungen sind die Knotentypen stark an den Dokumenten bzw. Artefakten ausgerichtet. Das soll in der groben Struktur auch nicht verändert werden. Es kann allerdings durchaus sinnvoll sein, die Granularität der Knotentypen

in einigen Fällen feiner zu gestalten. Das würde bedeuten, dass auch Teile von Artefakten Knoten sein können. Wird beispielsweise ein Textdokument in mehrere Knoten aufgeteilt, kann das für die gemeinsame Arbeit am Dokument hilfreich sein. So könnten unterschiedliche Abschnitte gleichzeitig von unterschiedlichen Personen bearbeitet werden, ohne dass es zu Kollisionen beim Zugriff oder bei der Speicherung kommt.
Auch könnten die Quellcodedateien verfeinert werden, indem – im Falle von objektorientierten Sprachen – jede Klasse in ihrer Gesamtheit ein Knoten ist, die einzelnen Methoden aber noch wieder in eigenen Knoten abgespeichert werden.
Bei welchen Knotentypen die Verfeinerung vom Benutzer als notwendig oder überflüssig erachtet wird, wird sich im weiteren Verlauf des Projekts herausstellen, wenn ein funktionierender Prototyp existiert und Nutzertests durchgeführt werden können. Von der Implementierung her stellt die Verfeinerung kein Problem dar, da es eine Menge an Basisknotentypen gibt, von denen mittels Vererbung bei Bedarf weitere Typen abgeleitet werden können.

4.1.2 Werkzeuge

Die Werkzeuge dienen dazu, die Artefakte bzw. Materialien zu bearbeiten. Bei handelsüblichen Systemen wie zum Beispiel Windows, Linux oder Macintosh ist es üblich, dass es für jedes Material bzw. jeden Dateityp ein eigenes Werkzeug bzw. eine eigene Anwendung zur Bearbeitung gibt. So ist zum Beispiel Photoshop ein klassisches Werkzeug zur Bildbearbeitung, Excel dient der Tabellenkalkulation und mit Together können UML-Diagramme erstellt und bearbeitet werden. Somit sind die Werkzeuge völlig unabhängig voneinander und nur für ihre Spezialfälle einsetzbar. Diese Metapher soll in kora nodes aufgebrochen werden. Die ANwendung soll einen Werkzeugkasten beinhalten, mit dem sämtliche Dokumenttypen bzw. hier Knotentypen manipulierbar sind. Es wird also Werkzeuge geben, die für alle Artefakte benutzt werden können, aber auch einige, die nur bei speziellen Artefakten sinnvoll einzusetzen sind. Die Werkzeuge in Abbildung 13 sind zum Beispiel auf alle Materialien anwendbar, wobei bei der Funktion „bearbeiten“ zwischen den Artefakttypen differenziert werden muss. Dieses sollte das kora nodes aber selbst erkennen und im Modus „bearbeiten“ den Artefakttypen angepasste Werkzeuge zur Verfügung stellen.
In der ersten Entwicklungsstufe von kora nodes werden diese Baukästen über externe Programme eingebunden werden, da es zunächst nicht sinnvoll erscheint, Werk-

zeuge wie Eclipse oder PowerPoint neu zu schreiben. Somit sollte ein Weg zur Integration dieser Anwendungen gefunden werden. Allerdings sollen im weiteren Verlauf des Projekts eigene Funktionskomponenten entstehen, die entsprechende Werkzeugkästen zur Verfügung stellen. Auf diese Weise können die Grenzen der Spezialanwendungen, die in der ersten Version bestehen, aufgebrochen werden. Die Funktionskomponenten werden einen geringeren Umfang haben, als die in diesen Prototyp eingebundenen Fremdanwendungen, da die Artefakte in einer Konferenz zwar manipulierbar sein sollen, Detailarbeiten allerdings eher am Arbeitsplatz erledigt werden und nicht im Konferenzraum. Es sollte aber bei der Benutzung des Raumes beobachtet werden, ob eine Reduzierung des Funktionsumfangs wirklich umgesetzt werden sollte, da es auch sein könnte, dass der Raum nicht nur zum Konferieren, sondern auch als Gruppenarbeitsraum genutzt werden wird. In diesem Fall würde eine Reduzierung des Funktionsumfangs eine Reduzierung der Nutzbarkeit bedeuten.

Für diese erste Entwicklungsstufe wird die Zuordnung von Material und Werkzeug nach wie vor eher anwendungs- bzw. dokumentenbezogen sein, woraus sich eine Zuordnung von Material und Werkzeug, wie in Abbildung 14 dargestellt, abgeleitet werden kann.

Material (Artefaktknoten)	**Werkzeug (Anwendungsknoten)**
Textknoten	Textbearbeitung
UML-Knoten[1]	UML-Tool
Quellcode-Knoten	Entwicklungsumgebung/ Texteditor
Bildknoten	Bildbearbeitung
Präsentationsknoten	Präsentationswerkzeug
Tabellenkalkulationsknoten	Kalkulationswerkzeug
sonstige	anzeigen

Abbildung 14: Zuordnung von Material und Werkzeug in erster Stufe

Zusätzlich zu den hier aufgeführten Werkzeugen sollte es Werkzeuge geben, die auf jedes Material angewendet werden können. Eines dieser Werkzeuge hat die Funktionalität zum Anzeigen von Artefakten am großen Display. Die Funktion dieses Werk-

[1] UML-Knoten können in mehrere Typen unterteilt werden, werden aber alle mit dem gleichen Werkzeug bearbeitet. Von daher ist der Typ „UML-Knoten“ bei der Werkzeug-Zuordnung vorerst ausreichend.

zeugs ist in keiner Anwendung vorhanden, die als Werkzeug eingebunden werden kann, da Anwendungen wie zum Beispiel Word, Together und Eclipse für den Einzelplatz konzipiert sind und nicht für einen Multi-User-Arbeitsraum, wie dem hier beschriebenen, ausgelegt sind. Es ist durchaus denkbar, dass während der weiteren Entwicklung von kora nodes nach weiteren Spezialwerkzeugen, die in Kapitel 3.2.3 und in Abbildung 11 als Plattformdienste beschrieben wurden, verlangt wird. Dazu kann u. a. die Filterung, die Suche und der Im- und Export von Artefakten gehören. Auch eine Versionskontrolle der Artefakte sollte zu diesen Funktionen gehören.
Ein weiteres konferenzunterstützendes Werkzeug, dem kein offensichtliches Material zugrunde liegt, ist die Unterstützung von Brainstorming. Charakteristisch für Brainstorming ist, dass aus einer Diskussion heraus Ideen bzw. Stichpunkte oder Schlagworte gesammelt werden. Somit entstehen während des Brainstormings Artefakte. Ein denkbares Werkzeug für Brainstorming ist eine Anwendung, mit der Mindmaps erstellt werden. Abbildung 15 zeigt zur Veranschaulichung eine Mindmap. Im Mittelpunkt steht das Thema (hier „Brainstorming“), abgeleitet davon sind die unterschiedlichen Ideen und deren Unterpunkte. Auf diese Weise kann das klassische Brainstorming, das häufig mit Karteikarten, die anschließend gruppiert werden, abgebildet werden.

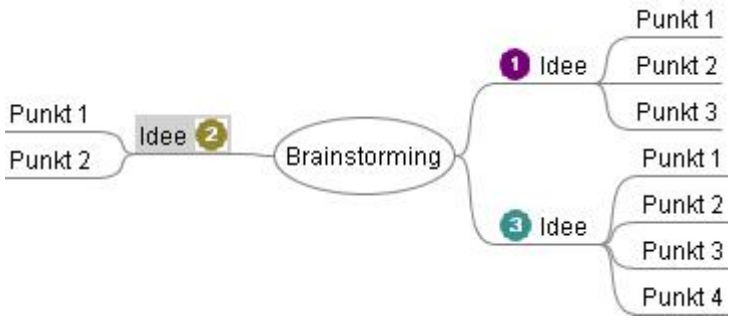

Abbildung 15: Mindmap für Brainstorming

Auch unabhängig von den Artefakten ist der Browser, der in kora nodes eingebunden werden soll. Er ist in dem Sinne ein Werkzeug, dass der Recherche im Internet dient. Im Anhang ist eine Tabelle zu finden, in der die Knotentypen mit ihren Attributen aufgelistet sind.

Die hier erarbeitete Zuordnung von Werkzeug und Material und deren Ausprägung ist für die erste Entwicklungsstufe gedacht. Diese Stufe bindet Fremdanwendungen

in das System ein, wodurch die Kopplung zwischen Dokument und Anwendung nach wie vor bestehen bleibt.

4.1.3 Behälter

Behälter dienen der Aufbewahrung von Materialien. In kora nodes werden einige Behälter zur Gruppierung und sinnvollen Strukturierung der Artefakte benötigt. Die Behälter sollten möglichst den Zusammenhang der Artefakte widerspiegeln. So wäre es zum Beispiel ein Ansatz die Artefakte nach ihrer Thematik in Behälter abzulegen. Bei einer solchen Strukturierung und in Anlehnung an die in Kapitel 2 erarbeiteten Projektsituationen können folgende Behälter abgeleitet werden:

- Organisatorisches
- Anforderungen
- Arbeitsabläufe/ Szenarios
- Material für Kundengespräche und Usability-Tests
- Glossar
- Testdaten
- Architektur
- Use-Cases
- Komponentenmodelle
- Workflows
- Quellcode
- Dokumentation
- Schulungsunterlagen
- Behälter für jeden Mitarbeiter
    - für jedes angemeldete Gerät wie PC, Laptop, Mobiltelefon oder PDA
    - für private Daten

Diese Liste ist beispielhaft zu sehen und kann beliebig bzw. nach Bedarf erweitert und gekürzt werden. Welche Behälter wirklich gebraucht werden und welche ggf. durch andere Behälter ergänzt werden müssen und welche vielleicht gar nicht benötigt werden, wird sich im späteren Verlauf des Projekts bzw. bei der Benutzung des

Raumes ergeben. In den Behältern liegen die Artefakte, die zu der der Thematik passen. Sie müssen nicht zwingend vom gleichen Typ sein.

4.2 Das User Interface

Nachdem die Grundlage von kora nodes erörtert wurde, bleibt zu überlegen, wie die Artfakte und Werkzeuge bzw. die Knotentypen dargestellt werden können. Die Benutzungsoberfläche bzw. das User Interface soll dem Nutzer möglichst vertraut vorkommen und die Artefakte müssen gut strukturiert angezeigt werden, da ein Softwareprojekt meistens sehr viele Artefakte hat. Eine gute Struktur der Daten erleichtert dem Benutzer den Zugriff und das Auffinden bestimmter Artefakte. Es gilt also eine Lösung zu finden, das Netz aus Artefakten, das kora nodes als Datenbasis hat, für den Benutzer so übersichtlich wie möglich darzustellen. Ein Problem ist, das ein Netz in sich birgt, dass es Zyklen enthalten kann. Diese sollten dem Benutzer allerdings nicht als solche präsentiert werden.

4.2.1 Darstellung der Artefakte

Um die Artefakte ordnen zu können und Abhängigkeiten aufzuzeigen, scheint die Baumstruktur als geeignet. Eine einfache Liste oder eine schlichter Behälter, in denen die Artefakte nur abgelegt werden, machen wenig Sinn, da die Übersicht über die Projektdaten verloren gehen würde. Auch in einem Baum gibt es unterschiedliche Ansätze, die Artefakte anzuordnen, wobei darauf geachtet werden muss, dass eventuell vorhandenen Zyklen nicht als endlose Äste dargestellt werden.
Alle Artefakte, die zu einem Projekt gehören, könnten untereinander angeordnet sein. Beim Doppelklick auf ein Artefakt, öffnet sich der Ast des Baumes. Hier werden die Artefakte angezeigt, die mit dem ersten Artefakt in Beziehung stehen. In Abbildung 16 ist kora nodes mit der hier beschriebenen Struktur der Artefakte dargestellt.

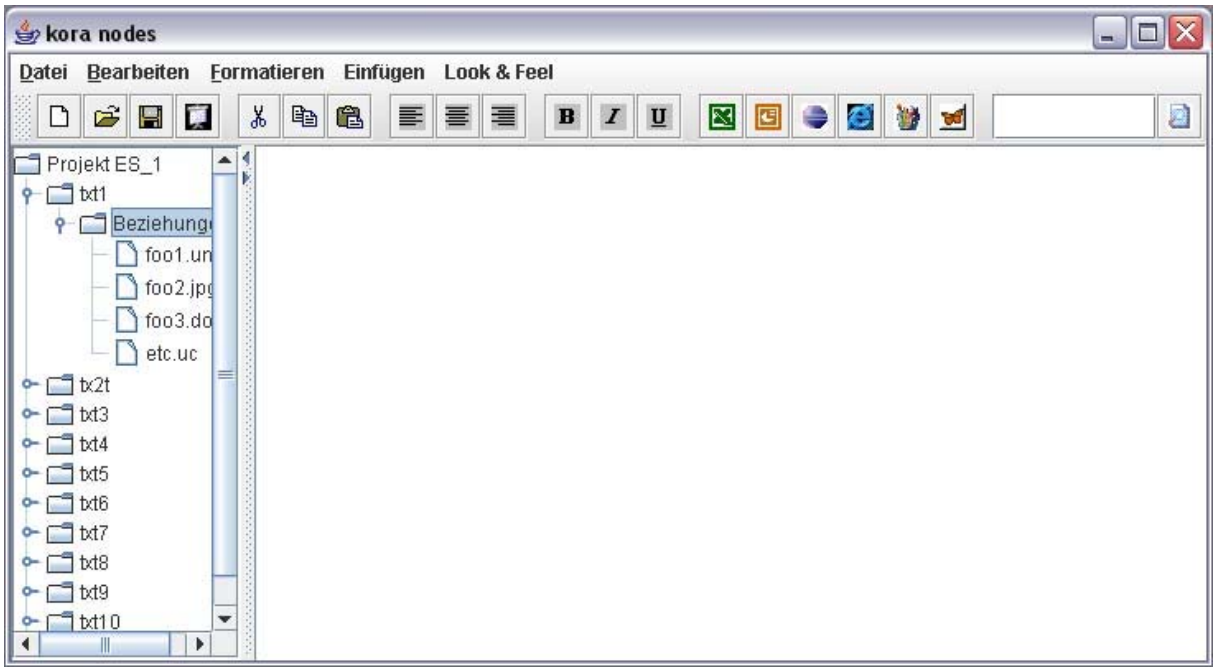

Abbildung 16: kora nodes unsortiert

Die Struktur in Abbildung 16 bringt einige Nachteile mit sich, die dem Benutzer die Arbeit mit der Anwendung erschweren. In einem Projekt werden deutlich mehr Artefakte erstellt, als in dieser Abbildung vorhanden sind. Somit wird der Baum immer länger und die Chance, das gesuchte Artefakt zu finden, wird immer geringer. Die Suche nach Artefakten würde dem Benutzer sehr viel Zeit kosten oder er müsste regelmäßig die Such- oder Filterfunktion nutzen.

Zusätzlich kritisch zu betrachten sind die Artefakte, die in den Ordnern „Beziehungen“ liegen. Wenn alle Artefakte bereits auf der ersten Ebene aufgeführt sind, würden sie in den Ordern „Beziehungen“ noch mal im Baum auftauchen. Da diese Artefakte wiederum mit anderen in Beziehung stehen, würden extrem viele Artefakte mehrfach im Baum angezeigt werden. Damit würde die Struktur noch unübersichtlicher werden. In Abbildung 16 endet die Baumstruktur mit den in den Ordnern „Beziehungen“ stehenden Artefakten. Hätte nun jedes Artefakt aus diesem Ordner wiederum einen Ordner „Beziehungen“, würden die Bäume endlos tief werden, da der Graph von einem Artefakt über andere wieder auf sich selbst zeigen könnte. Auf diese Weise entstehen endlose Schleifen bzw. Zyklen, die im Baum nicht dargestellt werden können. Von daher birgt diese Lösung zu viele Schwierigkeiten, sie soll deswegen nicht weiter verfolgt werden.

Da die scheinbar einfache Lösung, die Daten unsortiert zu lassen, sich als unbrauchbar herausgestellt hat, muss eine Lösung erarbeitet werden, bei der die Artefakte sortiert sind. Deshalb sollte die Behälteraufteilung wie in Kapitel 4.1.3 angedacht im Prototyp umgesetzt und ausprobiert werden. Abbildung 17 zeigt kora nodes mit entsprechend geordneten Artefakten. Die Behälteraufteilung ist exemplarisch und kann bei Bedarf verändert werden.

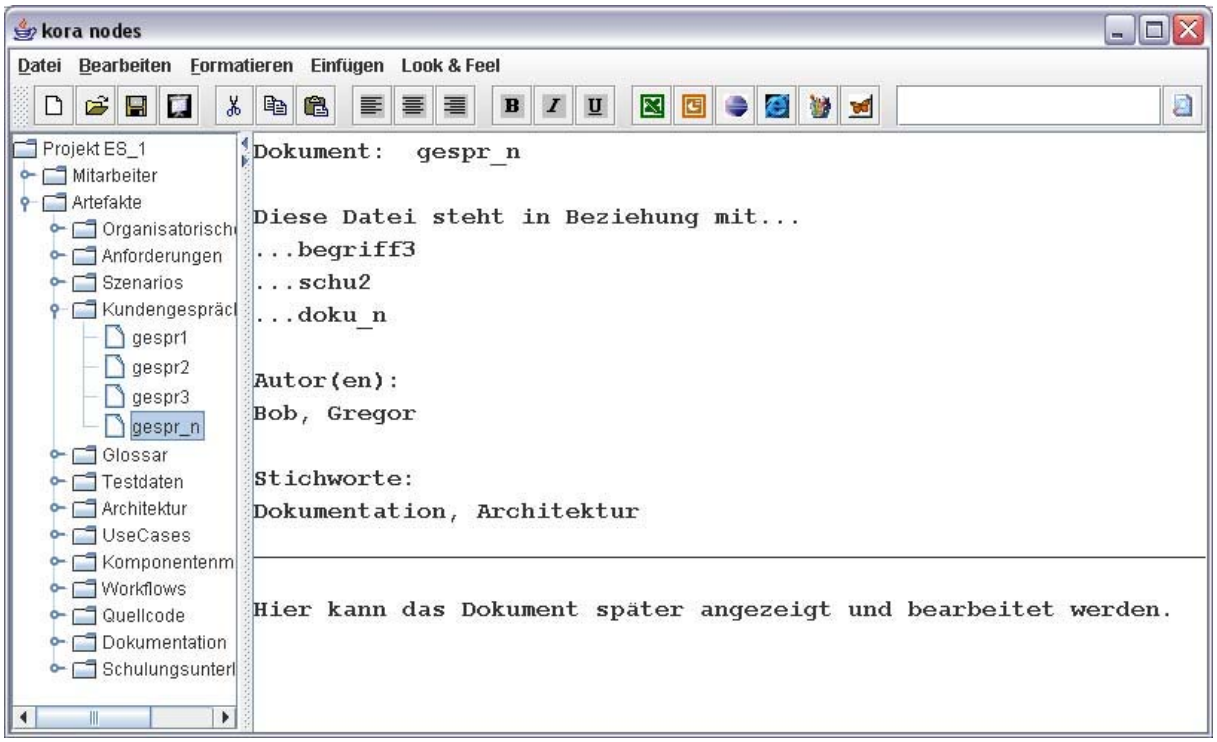

Abbildung 17: kora nodes sortiert nach Themen

Diese Darstellung der Dokumente hat den Vorteil, dass der Benutzer nicht unter sämtlichen Artefakten ein bestimmtes heraussuchen muss. Da davon ausgegangen werden kann, dass der Benutzer weiß, zu welchem Thema er ein Dokument sucht, ist die Anzahl der Dokumente, unter denen er suchen muss, deutlich reduziert. Der Ast des Baumes ist mit Auflistung der Artefakte beendet. Es sind keine weiteren Verschachtelungen vorgesehen. Um die Verflechtung des Netzes darzustellen, gibt es im rechten Feld der Anwendung eine Liste mit Links zu den Knoten, die mit dem gerade betrachteten zusammenhängen. Klickt der Nutzer auf einen dieser Links, öffnet sich der entsprechende Knoten bzw. das Artefakt.

Im Textfeld auf der rechten Seite werden die für den Benutzer relevanten Informationen, die in den Attributen der Knoten enthalten sind, angezeigt. Ganz oben steht der Name des Artefakts, darunter sind die Beziehungen aufgelistet. Diese werden später

als Links realisiert, so dass der Benutzer nur auf den Link klicken muss, um eine dieser Dateien zu öffnen. Auf diese Weise werden keine Dateien doppelt im Baum aufgeführt und der Benutzer ist relativ leicht in der Lage, die von ihm gesuchten Artefakte zu finden. Für den Fall, dass der Benutzer den Namen oder Typ des Artefakts nicht mehr weiß, gibt es eine Such- bzw. Filterfunktion. Diese ist durch das Textfeld in der Symbolleiste und dem Button mit der Lupe dargestellt. Die im Prototyp erstellte Funktion ist sehr einfach. Später sollte sie durch einen Dialog ersetzt werden, der u. a. eine Filterung der Artefakte nach ihren Attributen wie „Datum“ oder „Autor“ unterstützt.

Die beiden bis jetzt präsentierten Lösungen erinnern sehr stark an den Windows-Explorer und spiegeln das Netz, das kora nodes zu Grunde liegt, nur schwer erkennbar wider. Dennoch bietet die zweite dieser Lösungen die Möglichkeit, in dem Netz zu navigieren. Ändert ein Benutzer ein Artefakt, kann er anhand der Linkliste sehen, welche Artefakte er ggf. anpassen muss und hat eine direkte Möglichkeit diese zu öffnen. Auch könnte das System die Links einfärben, die auf Artefakte weisen, auf die sich die Bearbeitung eines Artefakts ggf. auswirken könnte. Auf diese Weise würde der Benutzer auf die Tragweite der Veränderung aufmerksam gemacht und die Durchgängigkeit der Artefakte bliebe bewahrt.

Eine weitere Idee zur Darstellung der Artefakte wäre, sie im dreidimensionalen Raum als Netz anzuordnen. Die Knoten würden mit Beschriftung und ihren Kanten als Verbindungslinien angezeigt werden. Abbildung 18 zeigt eine solche Darstellung eines Netzes.
Die Navigation in einem solchen Netz könnte wie folgt aussehen:
Der Benutzer klickt mit der Maus auf einen Knoten, woraufhin dieser heranzoomt und somit größer im Fokus dargestellt wird. Der Rest des Netzes wird etwas verkleinert im Hintergrund angezeigt, so dass der Knoten mit seinen Beziehungen betrachtet werden kann. Die Bearbeitung bzw. die Anzeige des Inhalts des Knotens – also des Artefakts – könnte der Benutzer durch einen Doppelklick auf den Knoten erreichen. Die Kanten zu den anderen Knoten könnten sich nach einer Veränderung des Artefakts rot färben, damit der Benutzer weiß, welche Knoten er ggf. anpassen muss, um die Durchgängigkeit in der Entwicklung zu bewahren.

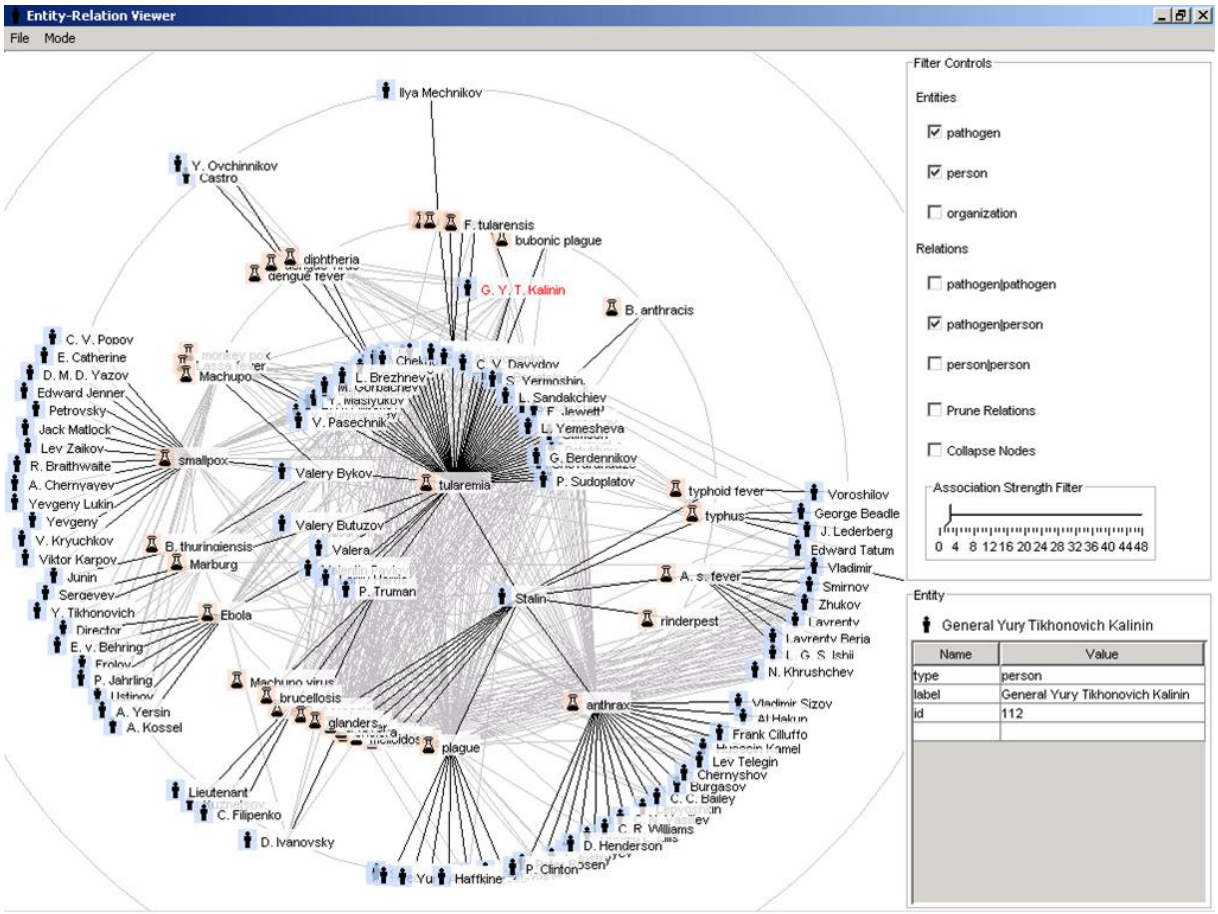

Abbildung 18: Darstellung im 3D-Netz (Fromherz 2006)

Diese Form der Darstellung spiegelt das Netz am besten wieder, könnte auf den Benutzer aber ungewohnt und komplex wirken. Abbildung 18 zeigt ein Netz mit sehr vielen Knoten und Kanten ähnlich wie es in einem Entwicklungsprojekt sein könnte.

Die im Prototyp von kora nodes gewählte Baumstruktur wird auf den Nutzer wahrscheinlich vertrauter und durch ihre Struktur übersichtlicher wirken, weswegen der Ansatz in dieser Arbeit auch weiter verfolgt wird. Eine Gegenüberstellung der beiden Lösungen, um diese Annahme zu bestätigen oder zu entkräften, wird in dieser Arbeit nicht durchgeführt, wäre aber in der Zukunft des Projekts durchaus sinnvoll.

4.2.2 Darstellung der Werkzeuge

Nachdem im vorigen Kapitel eine Darstellung für die Artefakte gefunden wurde, bleibt nun die Darstellung der Werkzeuge zu klären. Die Funktionen und Werkzeuge von kora nodes werden in einer WIMP-Anwendung eingebettet. Somit ist die Oberfläche von kora nodes der von Anwendungen wie zum Beispiel Notepad, Excel oder Word

recht ähnlich. So hat die Menü-Leiste den vertrauten Aufbau von „Datei“, „Bearbeiten“, „Formatieren“ und „Einfügen“. Zusätzlich gibt es ein Menü „Look & Fell“, mit dem der Benutzer eine Oberfläche seiner Wahl einstellen kann. Die drei Möglichkeiten – Metal, Motif und Windows – sind in Abbildung 19 dargestellt. Auf diese Weise hat der Benutzer die Möglichkeit, kora nodes zu personalisieren bzw. die Oberfläche zu wählen, die ihm am angenehmsten ist.

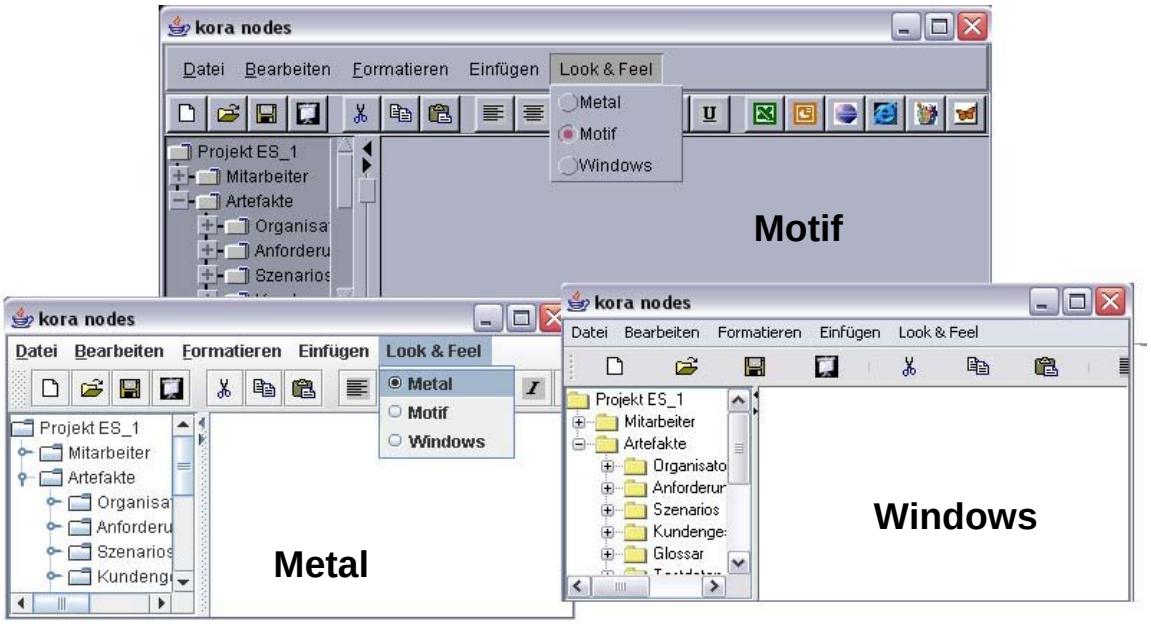

Abbildung 19: Look & Feel

Im Datei-Menü ist zusätzlich zu den gewohnten Funktionen wie „Neu“, „Öffnen“, „Speichern“ und „Speichern unter“ die Funktion „Veröffentlichen“ zu finden (vgl. Abbildung 20). Mit diesem Werkzeug kann jedes Artefakt am großen Display angezeigt werden. Das Werkzeug kann in drei Varianten aufgerufen werden: über das Menü, über einen Button, der in Abbildung 20 verdeckt ist, und über die Tastenkombination „Strg + D“. Auf diese Art können auch die anderen Plattformdienste wie zum Beispiel der Im- und Export von Artefakten eingebunden werden. Eine Versionskontrolle der Artefakte könnte als Automat im Hintergrund laufen.

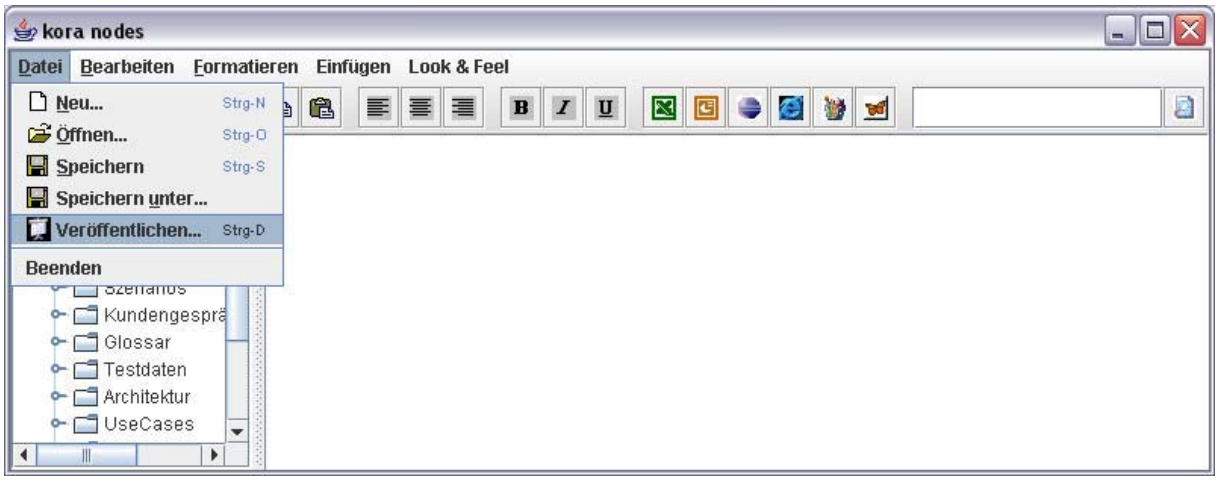

Abbildung 20: Werkzeuge in kora nodes

Des Weiteren enthält der Prototyp Werkzeuge zur Textbearbeitung, die dazu dienen, Textdateien direkt in der Arbeitsfläche von kora nodes zu bearbeiten (siehe Abbildung 17). Ein Ziel für spätere Entwicklungsstufen wäre, diese Werkzeuge für alle Artefakttypen zu verwenden. Da das in dieser Entwicklungsstufe noch nicht geht, werden die Werkzeuge zur Bearbeitung anderer Artefakttypen als Buttons dargestellt. So sind unter der Menüleiste in Abbildung 20 u. a. Buttons mit den Icons von Excel, PowerPoint, Eclipse, Internet Explorer, Paint und Freemind zu finden. Bei einem Mausklick auf einen der Buttons startet die entsprechende Anwendung. Das Artefakt, das bearbeitet werden soll, kann anschließend per drag & drop in das neu aufgegangene Fenster gezogen werden, woraufhin es dort angezeigt und bearbeitet werden kann. Auf diese Weise können eine Vielzahl an Anwendungen in kora nodes eingebunden und von dort aus mit all ihren Funktionalitäten genutzt werden. Das hat den Vorteil, dass der Nutzer nicht erst nach der Verknüpfung zu einem dieser Programme auf dem Rechner suchen muss, sondern direkt aus der gerade geöffneten Maske von kora nodes die Anwendungen starten kann.

4.2.3 Darstellung der Mitarbeiter

Auch die Mitarbeiter sind im Datenmodell von kora nodes Knoten und werden in der Benutzungsoberfläche angezeigt. Bei einem Projekt sind immer nur die am Projekt beteiligten Mitarbeiter aufgeführt, da in einer Firma nicht zwingend alle Mitarbeiter an einem Projekt arbeiten. Somit enthält der Projektordner einen Unterordner mit den Mitarbeitern, die wiederum Unterordner haben (vgl. Abbildung 21).

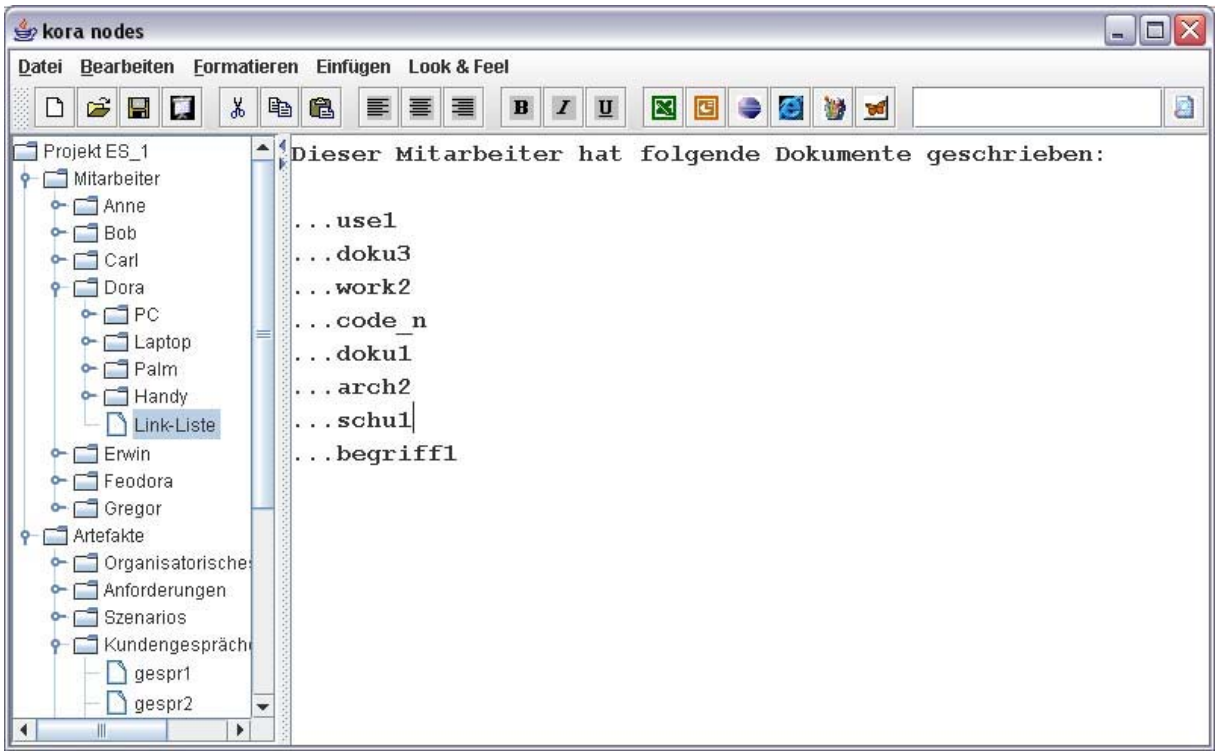

Abbildung 21: Darstellung der Mitarbeiter

Je nachdem mit welchen Geräten ein Mitarbeiter angemeldet ist, wird für jedes Gerät ein Ordner angezeigt. Zusätzlich ist im Ordner eines jeden Mitarbeiter eine vom System erstellte Link-Liste, die Links zu allen Artefakten enthält, die der Mitarbeiter erstellt hat oder zumindest an der Erstellung mitgewirkt hat. Die Liste wird von dem Attribut „Autor“, das jeder Artefaktknoten hat, abgeleitet. Ist ein Mitarbeiter nicht angemeldet, wird er ausgegraut dargestellt. Auf die Link-Liste kann bei Bedarf aber immer zugegriffen werden.

Auf den privaten Geräten eines Mitarbeiters können auch private Daten liegen, die dem Projektteam verborgen bleiben sollen. Von daher werden die Daten von privaten Geräten nur angezeigt, wenn der Eigentümer das explizit wünscht. Allerdings gibt es jederzeit die Möglichkeit, dem Mitarbeiter ein Dokument auf sein Gerät zu schieben bzw. zu kopieren. Ob er das wünscht oder nicht, wird in diesem Fall über soziale Kontrolle geregelt. Im späteren Umgang mit dem Raum wird sich zeigen, ob die soziale Kontrolle hier ausreicht oder Restriktionen im System implementiert werden müssen. Das Kopieren der Dateien wird über drag & drop realisiert.

4.3 Ausblick

In einer weiteren Entwicklungsstufe, in der keine Fremdanwendungen in das System eingebunden werden, sieht die Zuordnung von Werkzeug und Material anders aus. Die Knotentypen können in der erarbeiteten Unterteilung verwendet werden, wobei sie nun unabhängig von der Anwendung in einem großen Behälter – einem gemeinsamen Repository – gespeichert werden. Dieses Repository enthält unstrukturierten Inhalt, der durch kora nodes dem Benutzer strukturiert zur Verfügung gestellt wird. Die Struktur der Daten kann, wie im Prototyp in Behältern zum Beispiel nach Thematik geordnet, dargestellt werden. Denkbar wäre aber auch eine beliebige andere sinnvolle Anordnung wie zum Beispiel die Netzdarstellung. Soweit haben die erste und zweite Entwicklungsstufe die gleiche Basis. Der Unterschied wird in den Funktionskomponenten – also Werkzeugen – liegen. Da in dieser Entwicklungsstufe keine Fremdanwendungen als Werkzeuge für bestimmte Dokumenttypen eingesetzt werden sollen, sondern eigene Funktionskomponenten in kora nodes integriert werden, wird die starre Zuordnung von Knotentyp zu Spezialwerkzeug aufgehoben. Das Ziel ist hierbei, die Grenzen der Anwendungen zu sprengen und die Dokumente untereinander besser zu verbinden. Es soll nach Möglichkeit mit einem Werkzeugkasten gearbeitet werden, der für alle Knotentypen benutzbar ist. Zusätzlich könnten bei Bedarf für jeden Knotentyp einige Spezialwerkzeuge benötigt werden. Der Werkzeugkasten könnte wie folgt aussehen:

- neuen Knoten anlegen (mit Angabe von Knotentyp und Abhängigkeiten)
- speichern
- anzeigen/ veröffentlichen
- bearbeiten
- schrittweise undo-/ redo-Funktion
- Reset auf Ausgangszustand
- Im-/Export von Artfakten bzw. Knoten
- verschieben

Das Werkzeug „bearbeiten“ wird je nach Knotentyp spezielle Werkzeugkästen enthalten müssen, da die Artfakte von ihrer Grundstruktur sehr unterschiedlich sind. So wird zum Beispiel UML-Modell mit anderen Methoden und Zeichensätzen be- bzw. erarbeitet als ein Textdokument oder Quellcode. Diese speziellen Werkzeuge sollte

der Nutzer aber nicht selber im System suchen müssen, sondern vom System angezeigt bekommen, sobald er die Funktion „bearbeiten“ auswählt. kora nodes erkennt den Knotentyp und stellt den entsprechenden Werkzeugkasten automatisch bereit.
Auch die anderen Funktionen werden wohl jeweils an den Knotentyp und dessen Struktur angepasst werden müssen. Dies soll der Benutzer allerdings nicht merken. Vielmehr soll kora nodes anhand des Knotentyps und dessen Metadaten selbst erkennen, welche Implementierung des Werkzeugs ausgeführt werden muss.
Eine sehr wichtige Eigenschaft, die die Implementierung besitzen muss, ist eine Versionskontrolle, um sicher zu stellen, dass nur mit der aktuellen Version eines Knoten gearbeitet wird. Diese kann sich unter Umständen sehr schnell ändern, wenn mehrere Benutzer gleichzeitig in einer Konferenz an einem Dokument arbeiten. Sicherzustellen ist somit auch, dass die Funktionskomponenten von mehreren Benutzern gleichzeitig bedient werden können.
In dieser Entwicklungsstufe können unter einer Oberfläche alle Knotentypen bearbeitet werden ohne die Grenzen, die in der ersten Stufe durch die Fremdanwendungen gezogen worden sind und auch bei herkömmlichen auf dem Markt üblichen Systemen existieren. Auch wird diese Version von kora nodes alle Anforderungen erfüllen, die in Tabelle 1 im Anhang aufgelistet und in Kapitel 3.3 zu lesen sind.

4.4 Leitbild und Benutzungsmodell

In Kapitel 3.2 wurden die Grundlagen des WAM-Ansatzes beschrieben, ein Leitbild für kora nodes erdacht und die Entwicklung eines Benutzungsmodells angestrebt. Zur Erstellung eines solchen Modells gibt es keine Anleitung, doch geht es im Wesentlichen um die Beantwortung der Fragen, wie eine Anwendungssoftware genutzt wird, was für die Benutzbarkeit in Hinblick auf Handhabung, Ergonomie und die fachlichen Gegenstände wichtig ist und wie das Bild des Arbeitsplatzes unter Betrachtung des Leitbildes und der Metaphern aussieht.
Die Frage zum Anwendungskontext und zu den Geschäftsprozesse, die kora nodes unterstützen soll, sowie die Anforderungen, die an kora nodes gestellt werden, sind in den Kapiteln 2 und 3 erörtert worden. Kurz zusammengefasst soll sie Diskussionsprozesse unterstützen und helfen, effektiver – durch einfache Handhabung der Artefakte – zu tagen. Aufwändige Nachbereitung von Konferenzen und schwierige

Materialverteilung sollen vermindert und die Dokumentation sowie die Speicherung der Artefakte in Softwareprojekten vereinfacht werden.
In Hinblick auf die Ergonomie ist für kora nodes wichtig, dass es intuitiv bedienbar ist, da eine Gruppe, die ein Meeting abhält, nicht viel Zeit hat, sich in ein kompliziertes System einzuarbeiten. Die Artefakte des Projekts sollten gut strukturiert und offensichtlich vorliegen, ohne dass die Benutzer lange suchen müssen, wo Artefakte oder Werkzeuge zu finden sind. Sollte ein bestimmtes Dokument nicht sofort zu finden sein, muss es eine effektive Suchfunktion bzw. einen Dokumentenfilter geben. Zusätzlich müssen die Funktionen, die die Anwendung hat, schnell erkennbar sein.
In Kapitel 3.2.1 wurde festgestellt, dass das Leitbild Arbeitsplatz für eigenverantwortliche Expertentätigkeit von seiner Definition her auf das Konferenzsystem passen müsste. Die Entwurfsmetaphern Werkzeug, Material und Behälter konnten auf die Funktionen und Artefakte von kora nodes vorerst übertragen werden. Somit ist es bis jetzt nicht erforderlich, ein neues Leitbild für den kollaborativen Arbeitsplatz zu entwerfen. Letztendlich sind die Arbeitsplätze im Konferenzraum Einzelarbeitsplätzen sehr ähnlich. Es bleibt aber offen und zu beobachten, wie die Benutzer den Raum tatsächlich nutzen und wie die Konferenzen in diesem neu gestalteten Raum wirklich ablaufen. Erst wenn Szenarien für die Arbeit in dem neuen Raum stehen, kann eine finale Aussage zum Leitbild, das diesem Konferenzsystem zu Grunde liegt, getroffen werden. Es wäre also denkbar, dass das Leitbild angepasst oder doch noch neu entworfen werden muss.
Zur Benutzung des Systems wurde in Kapitel 4.2 bei der Vorstellung des User Interfaces einige Punkte erklärt. Die Grundidee bei kora nodes ist, den Schwerpunkt auf die Artefakt- bzw. Dokumentenverwaltung zu legen. Das Zusammenspiel zwischen den einzelnen Werkzeugen soll dadurch harmonisiert werden. Bisher haben die Artefakte eine Endung, durch die das System erkennt, mit welcher Anwendung es geöffnet werden soll. Die Abhängigkeit zum Beispiel von einem Use-Case und einer Java-Datei erkennt es aber nicht. Die Artefakte in kora nodes bekommen Zusatzinformationen, durch die u. a. ihre Abhängigkeiten untereinander deutlich werden. Wird nun ein Artefakt verändert, macht kora nodes mit Hilfe einer Versionskontrolle den Benutzer darauf aufmerksam, dass die Änderungen ggf. Auswirkungen auf die Artefakte haben, die mit dem veränderten in Beziehung stehen. Ändert zum Beispiel ein Entwickler die Struktur einer Java-Klasse, wirkt sich das u. U. auch auf das dazugehörige UML-Diagramm und weitere Artefakte aus. Die daraus resultierenden Änderun-

gen muss der Benutzer zwar nach wie vor selber machen, aber die Wahrscheinlichkeit, dass Unstimmigkeiten entstehen bzw. unentdeckt bleiben, wird verringert. Zusätzlich zur Versionskontrolle hilft bei dieser Problematik auch die in Abbildung 22 dargestellte Ansicht der Artefakte. In der Ansicht eines jeden Artefakts existiert eine Link-Liste, in der alle abhängigen Artefakte aufgelistet bzw. verlinkt sind. Bearbeitet ein Benutzer ein Artefakt, kann er mit Hilfe dieser Liste sehr schnell erkennen, auf welche Artefakte sich die Änderung auswirkt. Die Artefakte sind durch einen Mausklick leicht zu öffnen. Da dem Artefakt bekannt ist, mit welchem Werkzeug bzw. mit welcher Anwendung es bearbeitet werden muss, wird das Artefakt gleich in der richtigen Umgebung gestartet.

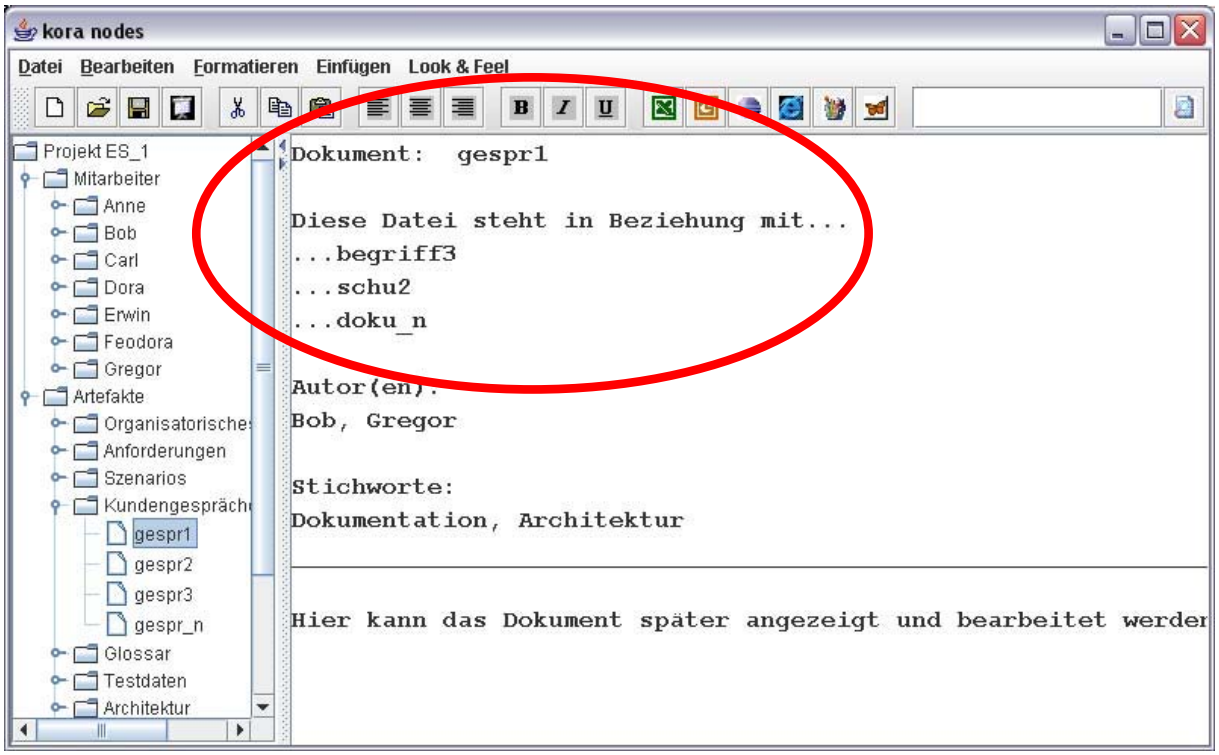

Abbildung 22: Beziehungen eines Artefakts

4.5 Ergebnis

In diesem Kapitel wurde auf Basis der Überlegungen aus den vorherigen Kapiteln ein Prototyp erstellt, der die Vision zur ersten Entwicklungsstufe von kora nodes widerspiegelt. In diesem Prototyp sind die Anforderungen an das System weitestgehend umgesetzt bzw. im Ansatz erkennbar. Allerdings gibt es noch viele offene Punkte, die sich erst beantworten lassen, wenn das System tatsächlich im Einsatz ist bzw. wenn erste funktionsfähige Prototypen entstanden sind, an denen Usability-Tests durchge-

führt werden können. Zu diesen offenen Punkten gehören die Fragen der Darstellung, ob das Netz besser in dem hier gewählten 2D-Baum oder besser in einem 3D-Netz präsentiert werden sollte.
Weiterhin unklar bleibt auch, wie die Werkzeuge im weiteren Verlauf des Projekts eingebunden werden sollen. Die hier verwendeten Fremdanwendungen sind nach wie vor nur zur Bearbeitung einzelner Dokumenttypen. Das bedeutet, dass die Schranken der Einzelanwendungen nicht aufgehoben werden. Somit ist das Ziel der „Harmonisierung der Anwendungen“ in dieser ersten Entwicklungsstufe nicht erreicht. Für die Fremdanwendungen spricht jedoch, dass sie einen so weiten Funktionsumfang bieten, dass der Konferenzraum auch als Gruppenarbeitsraum verwendet werden könnte. Das Hauptziel ist aber, die Grenzen der Anwendungen zu sprengen und Werkzeuge zu erschaffen, mit denen alle Artefakte bearbeitet werden können. Folglich ist der Funktionsumfang der Komponenten später an die tatsächlichen Bedürfnisse der Benutzer anzupassen. Diese zeigen sich allerdings erst, wenn der Raum in Benutzung ist.
Auch das gewählte Leitbild ist zwar vorerst in Ordnung und lässt sich scheinbar problemlos auf diesen Anwendungskontext übertragen. Ob es jedoch haltbar ist, wird erst der weitere Verlauf des Projekts zeigen.
Weil derzeit aber aus technischen Gründen die praktischen Erfahrungen noch fehlen, bleiben hier viele offene Fragen, für deren Lösung in dieser Arbeit keine Antwort gefunden werden kann.

5 Zusammenfassung und Zukunftsaussichten

Die Aufgabe dieser Arbeit bestand darin, die Anforderungen für ein Konferenzsystem zu analysieren und daraus ein Konzept bzw. einen Prototypen für die Benutzungsschnittstelle des Systems zu erarbeiten. Zu diesem Zweck wurde zu Beginn der Arbeit der Anwendungskontext genauer betrachtet. In diesem Fall sind es Diskussionsprozesse in der Softwareentwicklung. Um festzustellen, welche Art von Diskussionen in einem solchen Projekt typischer Weise auftreten, wurden zunächst Softwareprojekte auf deren typische Situationen und Abläufe hin betrachtet. Aus diesen Abläufen wurde Workflows erstellt. Auf Basis dieser Workflows konnten die Anforderungen an das System sowie ein Konzept für eine mögliche Darstellung erarbeitet werden, woraus anschließend ein Prototyp und ein Benutzungsmodell entwickelt worden sind.

Der Prototyp bietet viele Funktionen an, die in dieser Version allerdings nicht implementiert sind. Er veranschaulicht, wie die Artefakte und deren Zusammenhänge in einem Graphen dargestellt und wie vorerst externe Anwendungen als Werkzeuge zur Bearbeitung der Artefakte eingebunden werden können. Auf diese Weise bietet er dem Benutzer eine gute Übersicht über sämtliche Projektdokumente und deren Beziehungen untereinander. Auch vereinfacht er das Zusammenspiel der Anwendungen, mit denen die unterschiedlichen Artefakttypen bearbeitet werden. Der Nutzer hat die Möglichkeit durch eine geschickte Sicht auf die Daten, diese in kora nodes zu öffnen, ohne sich darum zu kümmern, welche Anwendung er dafür benötigt, da jeder Artefaktknoten weiß, mit welchem Werkzeug er bearbeitet wird. Eine wirkliche Harmonisierung der Anwendungen ist aber noch nicht erreicht. Dafür müssen eigene Funktionskomponenten für die Knotentypen geschrieben werden.
Der Prototyp verkörpert die Vision von kora nodes und wird u. a. in der Diplomarbeit von Petra Ehlers (Ehlers 2006) weiterentwickelt. In der Weiterentwicklung wird es interessant zu beobachten sein, ob die Darstellung der Artefakte und der dazugehörigen Werkzeuge vom Benutzer tatsächlich als vereinfacht empfunden wird. Um herauszufinden, inwieweit die hier erarbeitete Vision im Gebrauch als hilfreich empfunden wird, sollte durch Usability-Tests erforscht werden.

In Hinblick auf die Entwicklung des Prototypen, angelehnt an den WAM-Ansatz, ließen sich Werkzeug und Materialien recht einfach definieren. Die Entwicklung eines

Benutzungsmodells ist nach wie vor unkonkret, allerdings ist der Grund dafür deutlich geworden. Während des Vorgehens nach dem WAM-Ansatz macht der Entwickler sich Gedanken, was für Werkzeuge und Materialien im Anwendungskontext benötigt werden, um die Arbeitsabläufe beim Kunden zu unterstützen und wie diese eingesetzt werden. Aus diesen Überlegungen entsteht ein Prototyp, bei dem der Entwickler genaue Vorstellungen hat, wie er zu benutzen ist. Diese Überlegungen niedergeschrieben verkörpern quasi das Benutzungsmodell. Somit ergibt sich das Benutzungsmodell aus der Entwicklung der Software.

Im weiteren Ausblick wäre es zweckmäßig, wenn für die Werkzeuge keine externen Anwendungen genutzt werden müssten, sondern kora nodes eigene schlanke Werkzeuge besäße, die über ausreichende Funktionen verfügen. Der Funktionsumfang der Werkzeuge von kora nodes muss nicht den gleichen Umfang haben, wie eine Anwendung für die tägliche ausgefeilte Arbeit, da während einer Diskussion hauptsächlich Dinge besprochen und Entscheidungen getroffen werden sollen. Die eigentliche Arbeit an den Artefakten findet nach wie vor am Arbeitsplatz statt. Somit braucht kora nodes beispielsweise keine komplette Entwicklungsumgebung wie Eclipse, es würde auch ein Texteditor mit Syntax-Highlighting reichen, der den Quellcode gut strukturiert und übersichtlich anzeigt. Allerdings ist es bei der Ausstattung und dem Potential des Raumes – sofern er entsprechend umfangreiche Werkzeuge integriert – durchaus denkbar, dass er nicht nur für Meetings bzw. Konferenzen sondern auch für Teamarbeit genutzt werden könnte. Es bleibt zu überlegen oder ist später im Gebrauch auszutesten, inwieweit die umfangreichen Werkzeuge von den Diskussionen ablenken. So wäre es zum Beispiel vorstellbar, dass Entwickler anfangen, über Quellcode zu diskutieren und diesen im Raum direkt verändern und debuggen. Damit würde die Entwicklung vom eigentlichen Arbeitsplatz in den Konferenzraum verlegt werden. Ist das gewünscht, können umfangreiche Werkzeuge im Raum bleiben, ansonsten sollte für die Zukunft eine andere Lösung gefunden werden.

Eine Problemstellung in Bezug auf kora nodes, die in dieser Arbeit nicht behandelt wurde, soll an dieser Stelle kurz angedacht werden. Die Situation, dass Konferenzteilnehmer einen neuen Knoten anlegen wollen, ist sicher ein Szenario, das häufig vorkommen wird. Das Erstellen eines Worddokuments in einem Windowssystem ist ein sehr einfacher schlicht gehaltener Vorgang. Die einzigen Eingaben, die der Be-

nutzer machen sollte, ist der Name und ggf. der Speicherpfad. Bei einem neuen Knoten in kora nodes kann dieser Vorgang nicht ganz so einfach und schlicht werden, da der neue Knoten Attribute benötigt. Ohne die Attributwerte kann aus den Knoten kein Graph werden und die hier ausgearbeitete Dokumentenstruktur würde nicht in der erdachten Form dargestellt und aufbereitet werden können. Somit sollte es für die Erstellung neuer Knoten eine Art Formular bzw. einen Dialog geben, in dem der Benutzer möglichst wenige Attribute einträgt. Muss er viele Werte selbst definieren, könnte er die Lust daran verlieren, da es zu lange dauert. Absolut wichtige manuell nachzutragende Attribute sind u. a. der Artefakttyp, der Name, die Beziehungen und Stichworte für die Suche und den Dokumentenfilter. Die besondere Schwierigkeit liegt hier wohl in der Pflege der Beziehungen. Dieses Problem ist unabhängig von der gewählten Darstellung – ob Netz oder Baum, 2D oder 3D – immer vorhanden. Wobei die Pflege der Beziehungen in der 3D-Netzdarstellung noch wichtiger ist als im 2D-Baum, denn hier herrscht durch die Behälter eine gewisse Ordnung, die im Netz nicht vorhanden ist. Fehlen im Netz die Kanten, schwirren viele Knoten ohne Zusammenhang in einem Raum, was zur Folge hätte, dass der Benutzer keine Orientierungsmöglichkeit mehr hätte. Hier muss eine möglichst einfache Lösung gefunden werden, damit kora nodes überhaupt richtig arbeiten kann. Auch die Pflege der Stichworte für die Such- bzw. Filterfunktion sollte nicht vernachlässigt werden, wobei es dafür eine Möglichkeit zumindest zur teilweisen Automatisierung geben könnte.

Nachdem in dieser Diplomarbeit ein Schritt hin zu einer zukunftsorientierten Konferenztechnik mit der Möglichkeit zur Effizienzsteigerung von Diskussionsprozessen getan und eine Lösung zur besseren Darstellung der Projektdaten gefunden wurde, könnte der Weiterentwicklung des Projekts „kora“ eine große Zukunft bevorstehen. Allerdings stellt es nach wie vor eine große Herausforderung an das Entwicklerteam, diese Technik bis hin zum alltäglichen Einsatz voranzutreiben. Dem ersten funktionsfähigen Prototypen wird mit großer Erwartung entgegen geblickt.

Literaturverzeichnis

Balzert 2005 Heide Balzert: Lehrbuch der Objektmodellierung Analyse und Entwurf mit der UML 2; 2. Auflage, Spektrum Akademischer Verlag; 2005

Balzert 2000 Helmut Balzert: Lehrbuch der Softwaretechnik; 2. Auflage; Spektrum Akademischer Verlag; 2000

Bartnik 2006 Roman Bartnik: Weiterentwicklung einer Technologiebasis für interaktive Gruppenarbeitsräume; HAW Hamburg, Diplomarbeit 2006

Beier und von Gizycki (Hrsg.) 2002 Markus Beier (Hrsg.), Vittoria von Gizycki (Hrsg.): Usability Nutzerfreundliches Web-Design; Springer-Verlag, 2002

BfA 2006 Bundesagentur für Arbeit, BERUFENET, http://infobub.arbeitsagentur.de/berufe, 12.01.2006

Boehm 1981 Barry. W. Boehm: Software Engineering Economics, Prentice Hall, 1981

Brauner 2004 Philipp M. Brauner: Metriken zur Evaluation und Vergleichbarkeit von Benutzerschnittstellen in virtuellen Realitäten, Technische Hochschule Aachen, 13.02.2004

Burfeindt 2006 Lars Burfeindt: Konstruktion einer Middleware für computergestützte Gruppenarbeit in ubiquitärer Systemumgebung; HAW Hamburg, Diplomarbeit 2006 in Vorbereitung

CEN 1995 Europäisches Komitee für Normung: EN ISO 9241-10, Grundsätze der Dialoggestaltung, 1995

Churchill u. a. (Hrsg.) 2001 Elizabeth F. Churchill (Hrsg.), David N. Snowdon (Hrsg.), Alan J. Munro (Hrsg.): Collaborative Virtual Environments, Digital Places and Spaces for Interaction; Springer-Verlag, 2001

Crabtree 2003 Andy Crabtree: Designing Collaborative Systems, A Practical Guide to Ethnography; Springer-Verlag, 2003

Dittrich u. a. (Hrsg.) 2002 Yvonne Dittrich (Hrsg.), Christiane Floyd (Hrsg.), Ralf Klischewski (Hrsg.): Social Thinking Software Practice; MIT Press, 2002

Ehlers 2006 Petra Ehlers: Diplomarbeit HAW Hamburg 2006 in Vorbereitung; HAW Hamburg, Diplomarbeit 2006

Freund 2006 Jakob Freund: Geschäftsprozesse vs. Workflows, http://www.bpm-guide.de/articles/3, BPM-Guide.de, 22.03.2006

Fromherz 2006 Markus Fromherz : Information Visualization & Interaction, http://www.parc.com/research/projects/sensemaking/visualization_interaction/default.html, Xerox Parc, 01.06.2006

Gottwald 2006 Michael Gottwald: Diplomarbeit HAW Hamburg 2006 in Vorbereitung; HAW Hamburg, Diplomarbeit 2006

Heinsen und Vogt (Hrsg.) 2003 Sven Heinsen (Hrsg.), Petra Vogt (Hrsg.): Usability praktisch umsetzen – Handbuch für Software, Web, Mobile Devices und andere interaktive Produkte; Hanser, 2003

Herczeg 1994 Michael Herczeg: Software-Ergonomie – Grundlagen der Mensch-Computer-Kommunikation; Addison-Wesley, 1994

Israel 1999 Johann Habakuk Israel: Kommandofreie Interaktion http://useworld.net/ausgaben/10-1999/israel.pdf, Techische Universität Berlin, Oktober 1999

Jeckle u. a. 2003 Mario Jeckle, Chris Rupp, Jürgen Hahn, Barbara Zengler, Stefan Queins: UML 2 glasklar, Carl Hanser Verlag, 2003

Krug 2002 Steve Krug: Don’t make me think! Web Usability – Das intuitive Web; mitp-Verlag, 2002

Mund 2006 Horst Mund: Berechtigungsstrukturen in kollaborativen Umgebungen; HAW Hamburg, Diplomarbeit 2006 in Vorbereitung

Neumann 2005 Carola Neumann: User Centered Design oder Benutzerzentrierte Softwareentwicklung, Studienarbeit HAW Hamburg, August 2005

Raasch u. a. 2006 Prof. Dr. Jörg Raasch, Prof. Dr. Olaf Zukunft, Prof. Dr. Kahlbrandt: 023v-WAM1: Leitbilder und Metaphern, Vorlesungsfolien, März 2006

Rich 1988 Elaine Rich: Künstliche Intelligenz, KI-Einführung und Anwendungen; McGraw-Hill, 1988

Schümmer 2005 Till Schümmer: A Pattern Approach for End-User Centered Groupware Development; JOSEF EUL VERLAG, 2005

Shaw und Garlan 1996 Mary Shaw; David Garlan: Software Architecture, Perpectives on an emerging discipline; Pentice Hall 1996

SMART Technologies Inc. 2005 SMART Technologies: SMART Board Software – Features http://www.smarttech.de/sbsoftware/, Stand 14.09.2005

Starke 2002 Gernot Starke: Effektive Software – Architekturen, Ein praktischer Leitfaden; Hanser; 2002

Stroisch 2003 Jörg Stroisch: Job-Trends – Im Büro der Zukunft sitzt man auf dem „CommChair“, Stand 09/2005 http://www.jobpilot.de/content/journal/beruf/office06-03.html

Shneiderman und Plaisant 2005 Ben Shneiderman; Catherine Plaisant: Designing the user interface, fourth edition; Pearson Education, 2005

Tandler u. a. 2002 P. Tandler; N. A. Streitz; Th. Prante: Roomware – Moving Toward Ubiquitous Computers, IEEE Micro, Nov/ Dec 2002 http://www.ipsi.fraunhofer.de/ambiente/paper/2002/IEEEMicro-tandler-NovDec2002.pdf

Thaller 2002 Georg Erwin Thaller: Interface Design – Die Mensch-Maschine-Schnittstelle gestalten – Konzepte für Programm- und Web-Oberflächen; Software & Support Verlag 2002

WfMC 2006 The Workflow Management Coalition, http://www.wfmc.org/index.html, Stand 05/2006

Winograd und Flores 1989 Terry Winograd, Fernando Flores: Erkenntnis Maschinen Verstehen; Rotbuch Verlag, 1989

Züllighoven u. a. 1998 Heinz Züllighoven u. a.: Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz; dpunkt.verlag, 1998

Anhang

Tabelle 1: Auflistung der Anforderungen

Nr.	Kurztext	Erklärung	Priorität
01	Gebrauchstauglichkeit	Das System muss einfach und intuitiv bedienbar sein	+++
02	Datenstruktur	Das System braucht eine einfache und übersichtliche, leicht durchschaubare Darstellung der Daten	+++
03	Filter	Kriterienbasierter Dokumentenfilter, um eine bestimmte Sicht auf die Daten zu bekommen mit Erhalt der Datenstruktur	++
04	Suche	Kriterienbasierte Suchfunktion um Dokumente zu suchen ohne Erhalt der Datenstruktur	-
05	Internet	Browser für zum Beispiel Internetrecherche	-
06	mobile Geräte	Anbindung von Mobiltelefonen und PDAs	-
07	Steuerung der Displays	Displays sollen von allen Geräten aus und direkt gesteuert werden können.	+++
08	Bearbeitung am Display	Bearbeitung von Dokumenten soll von allen Geräten aus sowie direkt durchgeführt werden können.	+++
09	Anzeige am Display	Es sollen alle Dokumenttypen angezeigt bzw. geöffnet werden können unabhängig von vorhandenen Funktionskomponenten (FK)	+++
10	benutzerabhängige Public/ Private-Bereiche	Jeder Benutzer soll einen privaten und öffentlichen Bereich zur Ablage seiner Daten haben.	+
11	Dokumentenaustausch	Die Benutzer sollen Dokumente bzw. Daten untereinander austauschen können	+++
12	Im-/ Export von Daten	Daten sollen Im- bzw. Exportiert werden können.	++
13	Drucken	Es soll die Möglichkeit geben, Dokumente auszudrucken.	-
14	Scannen	Es soll die Möglichkeit geben, Dokumente einzuscannen	-
15	Versionskontrolle	Das System soll eine Versionskontrolle haben.	+++
16	FK auswählbar nach Benutzergruppe	Die Funktionskomponenten sollen je nach Benutzergruppe (Stadtplanung/ Softwareentwicklung etc.) auswählbar sein	+
17	neu	neuen Knoten erstellen	+++

18	speichern	Einen Knoten abspeichern	+++
19	anzeigen	Ein Dokument wird am großen Display angezeigt/ geöffnet	+++
20	veröffentlichen	Ein Dokument wird für alle Benutzer zugänglich gemacht	+++
21	UnDo/ ReDo	Einzelne Bearbeitungsschritte können schrittweise rückgängig gemacht oder wieder hergestellt werden	+
22	Reset	alle Änderungen können rückgängig gemacht werden	-
23	bearbeiten	Unterschiedliche Knotentypen sollen bearbeitet werden können.	+++
24	Hilfesystem	Das System benötigt ein Hilfesystem, falls sich ein Benutzer nicht zurecht findet	-
25	Einbindung externer Anwendungen	Externe Anwendungen sollen bei Bedarf auch im Raum nutzbar sein.	-
26	Raumbelegungsplan	Ein Raumbelegungsplan soll für die Verwaltung des Raums vorhanden sein.	-
27	Anmeldung/ Registrierung	Benutzer sollen sich registrieren und anmelden können.	++
28	Zugriff von außen	Die Daten sollen den Benutzern auch von außerhalb des Raumes zur Verfügung stehen.	-
29	Durchgängigkeit	Modelle aus einem Entwicklungsabschnitt (EA) sollen im Zusammenhang in weiteren Eas zur Verfügung stehen.	+++
30	Harmonisierung	Die Grenzen der Anwendungen sollen überwunden werden.	+

+++ muss unbedingt vorhanden sein
++ muss vorhanden sein
+ sollte vorhanden sein
- kann vorhanden sein

Die Prioritäten sind nach Gewährleistung der Funktionalität gestaffelt.

Tabelle 2: Übersicht über die Knotentypen

Knotentyp	**Attribute**	**erbt von**
Knoten	ID Name	
Projekt	Typ Erstellungsdatum	Knoten
Artefakt	Typ Projektzugehörigkeit Autor Werkzeug Erstellungsdatum Letzte Bearbeitung Beziehungen Stichworte (für Suche/ Filter) Zugriffsberechtigte Person Thema	Knoten
Mitarbeiter	Projektzugehörigkeit Rolle Erstellte Artefakte	Knoten
Werkzeug		Knoten
Spezialwerkzeug	Artefakttyp	Werkzeug
Artefaktfragment	Artfaktzugehörigkeit Autor Erstellungsdatum Letzte Bearbeitung	Knoten
Behälter	Typ	Knoten

Mögliche Werte für das Attribut „Typ“ für Projekt:
Softwareentwicklung, Stadtplanung

Mögliche Werte für das Attribut „Typ“ für Artefakte:
Text, UML, Quellcode, Bild, Präsentation, Tabellenkalkulation, sonstige

Mögliche Werte für das Attribut „Name“ für Werkzeug:
1. Entwicklungsstufe: Textbearbeitung, UML-Tool, Entwicklungsumgebung, Bildbearbeitung, Präsentationswerkzeug, Kalkulationswerkzeug
Weiterentwicklung: Neu, Speichern, Bearbeiten, Kopieren, Anzeigen, Veröffentlichen, UnDo, ReDo, Reset

Mögliche Werte für das Attribut „Typ“ für Behälter:
public, private

Bei Bedarf kann diese Liste beliebig erweitert bzw. ergänzt werden.

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.

Reinbek, den 29.06.2006
Ort, Datum

Unterschrift