Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

Lars Burfeindt

Konstruktion einer Middleware für computergestützte Gruppenarbeit in ubiquitärer Systemumgebung

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Lars Burfeindt

Konstruktion einer Middleware für computergestützte Gruppenarbeit in ubiquitärer Systemumgebung

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. rer. nat. Kai von Luck
Zweitgutachter : Prof. Dr. rer. nat. Jörg Raasch

Abgegeben am 17. August 2006

Lars Burfeindt

Thema der Diplomarbeit: Konstruktion einer Middleware für computergestützte Gruppenarbeit in ubiquitärer Systemumgebung

Stichworte CSCW, Ubiquitous Computing, Middleware, Stadtplanung, Collaborative Workplace

Zusammenfassung

Diese Arbeit beschreibt die Entwicklung einer Middleware für die Erstellung eines Konferenzraumes auf Basis einer ubiquitären Umgebung. Durch Integration mobiler Endgeräte und großer, hochauflösender Bildschirme soll Gruppenarbeit unterstützt und die Produktivität dieser gesteigert werden. Mit der hierzu entworfenen Anwendung wird die Verteilung von Dokumenten erleichtert und die Steuerung der Bildschirme mit Hilfe der mobilen Endgeräte ermöglicht. Die Anwendung basiert dabei auf der Middleware, welche im Rahmen dieser Arbeit entwickelt wird.

Lars Burfeindt

Thesis Title: Construction of a middleware for computer-supported collaborative work in an ubiquitous environment

Keywords CSCW, ubiquitous computing, middleware, town planning, collaborative workplace

Abstract

This paper decribes the development of a middleware for use in a conference room based on an ubiquitous environment. Integrating mobile clients and large, high resolution displays will support group work and thus improve productivity of it. The application developed in here will increase the ease of distributing documents and will enable the users to control the displays using their mobile clients. The application is based on the middleware which is developed in this paper.

Danksagung

Ich möchte mich bei allen Mitwirkenden der Projektgruppe Collaborative Workplace bedanken. Die vielen spannenden Diskussionen haben die Erstellung dieser Arbeit überhaupt erst möglich gemacht. Des weiteren bedanke ich mich bei meinen Eltern sowie meinen Freunden, insbesondere bei Ina Köhne, Olivia Reindorf und Johannes Hartz.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Motivation

Informationstechnologien sind stets im Wandel der Zeit. Der Beginn des Computerzeitalters war durch Großrechner geprägt. Ganze Räume waren für den Betrieb eines einzigen Rechners notwendig. Deren Einsatz war vor allem in Bereichen der Forschung zu finden. In der Wirtschaft beschränkte sich der Einsatz auf große Finanzunternehmen wie Banken und Versicherungen. Die zweite Generation von Computern brachte diese näher an den Menschen. Der Personal Computer (PC) kam auf, der Trend ging zum Computer für Jedermann. Der Computer wurde zum festen Einrichtungsgegenstand. Anfangs als Ersatz für die Schreibmaschine eingesetzt, entwickelte sich der PC mit Aufkommen des Internets zunehmend zu einem Informations- und Kommunikationsmedium. Seit Ende der 90er-Jahre drängen sich auch mobile Technologien auf den Markt. Laptops, PDAs und Mobiltelefone wurden zum ständigen Begleiter. Die Geräte werden immer kleiner und zudem immer leistungsstärker. Die Leistungsfähigkeit ist dabei nur ein Merkmal der so genannten Smart Devices. Kommunikationstechnologien wie Wireless LAN und Bluetooth sind heutzutage bereits integriert.

Teilten sich anfangs noch viele Personen einen Computer, so hat es sich heute in viele Computer für eine Person umgekehrt. Wir befinden uns somit inmitten der dritten Generation von Informationstechnologien, welche vor allem durch die Smart Devices geprägt ist.

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

Mark Weiser

Bereits 1991 hat Mark Weiser in [Weiser 1991] das Zusammenspiel der Smart Devices beschrieben und damit den Begriff des Ubiquitous Computings geprägt. Demnach wird die Rolle der Computer im alltäglichen Leben neu definiert werden. Anstatt wie heute im Fokus der Arbeit zu stehen, verschwinden Computer aus unserem direkten Sichtfeld und werden zu einem ständigen Begleiter. Mit Aufkommen der Smart Devices und den Möglichkeiten, diese entsprechend zu vernetzen, sind erste technische Vorraussetzungen für die Umsetzung von Mark Weisers Visionen gegeben.

Durch die Vielzahl der technischen Geräte die uns täglich begleiten, steigen auch die Möglichkeiten der Informationsgewinnung durch diese. Für den sinnvollen Einsatz der Computer ist daher die Filterung der für den Menschen wichtigen Informationen eine Schlüsselanforderung des Ubiquitous Computings [Streitz und Nixon 2005]. Für die Arbeit mit scheinbar verschwundenen Technologien stellt sich auch die Frage der Interaktionsformen zwischen Menschen und Computern. Die Eingabe über Tastatur und Maus hat sich besonders bei Verwendung von Personal Computern bezahlt gemacht, bei der Benutzung vieler kleiner Computer scheint dieses eher weniger sinnvoll.

Ein Forschungsfeld, welches sich ebenfalls mit der Mensch-Computer-Interaktion beschäftigt, ist Computer-Supported Collaborative Work (CSCW). Hierbei geht es um Gruppenarbeit und wie diese mit Hilfe von Computern unterstützt werden kann, um sie produktiver zu gestalten. Für computergestützte Gruppenarbeit scheint besonders der Einsatz großer, hochauflösender Bildschirme sinnvoll. Durch deren Einsatz ergeben sich eine Reihe neuer Möglichkeiten, um die Kommunikation innerhalb einer Gruppe zu verbessern. Im Rahmen des technologischen Fortschrittes sinkt auch der Preis für diese Bildschirme, so dass diesbezügliche Investitionen in naher Zukunft wirtschaftlich tragbar sein wird.

Verbindet man die beiden Themen Ubiquitous Computing und Computer-Supported Collaborative Work, so erhält man einen Gruppenarbeitsraum, der mit einer Vielzahl großer, hochauflösender Bildschirme ausgestattet ist und eine Reihe von Besprechungsteilnehmer, die ihre eigenen kleinen mobilen Geräte, samt ihren Dokumenten mitbringen. Dieses alles zusammen ergibt ein Netzwerk vieler verschiedener Geräte, die miteinander kommunizieren und zusammenarbeiten.

Ist die technische Ausstattung gegeben, stellt sich die Frage, wie Software das Zusammenwirken dieser Geräte umsetzen kann. Diese Arbeit beschäftigt sich mit der Middleware, welche die beschriebene Interaktion der einzelnen Geräte in die Wirklichkeit umsetzten soll. Eine Schlüsselfrage dabei wird sein, wie diese einzelnen Geräte miteinander kommunizieren, um sich gegenseitig bekannt zu machen, Dienste anzubieten und Dienste anderer Geräte zu nutzen.

Es gibt viele denkbare Szenarien, in denen Gruppendiskussionen vorkommen und für welche die Umsetzung eines solchen Besprechungsraumes interessant sein könnte. In dieser Arbeit wird das Szenario Stadtplanung aufgegriffen. Es soll gezeigt werden, wie die Arbeit in der Stadtplanung abläuft und wie der hier beschriebene Besprechungsraum im Rahmen der Stadtplanung sinnvoll Einsatz finden kann.

1.2. Problemstellung und Idee

1.2.1. Darstellung eines klassischen Besprechungsraumes

Projektbesprechungen sind ein elementarer Bestandteil einer jeden Gruppenarbeit. Gemeinsam werden Ideen gesammelt, Lösungswege ausgearbeitet, Aufgaben verteilt und Beschlüsse gefasst. Im Zuge dieses Vorganges werden Präsentationen gehalten, gemeinsam wird über verschiedene Dokumente diskutiert, es werden Skizzen erarbeitet und diverse Notizen gemacht. Für genau diese Zwecke werden Besprechungsräume auch mit entsprechendem Inventar ausgestattet.

Ein elementarer Bestandteil eines Besprechungsraumes ist der Beamer. Über diesen lässt sich der Laptop in den Besprechungsraum integrieren. Für Präsentationen ist der Beamer elementar, bei Besprechungen dient er dazu, gemeinsam ein Dokument zu betrachten.

Neben dem Beamer befindet sich vor allem ein Whiteboard in einem typischen Besprechungsraum. Das Whiteboard dient zur Erstellung von Skizzen, die zusammen mit der ganzen Gruppe erarbeitet werden. Vor allem durch seine Größe ist ein Whiteboard besonders für die Arbeit im Team geeignet. Es können mehrere Personen gleichzeitig vor einem Whiteboard stehen und gemeinsam daran arbeiten.

Auch für die Durchführung eines Brainstormings bietet sich das Whiteboard an. Karteikarten mit Ideen werden von den einzelnen Teilnehmern beschrieben und mittels Magnet am Whiteboard befestigt. Nach der Sammlung der Ideen werden diese gruppiert und es können weitere Beschriftungen am Whiteboard vorgenommen werden.

Häufig sind auch Flipcharts in Seminarräumen zu finden. Diese eignen sich ebenfalls zum Erstellen von Skizzen und Notieren von Stichworten. Im Vergleich zum Whiteboard sind Flipcharts wesentlich kleiner und man kann Geschriebenes nicht wieder wegwischen, dafür kann man die einzelnen Blätter aufbewahren, um später wieder auf diese zurückzugreifen.

Verwendung mehrerer Computer bei einem Beamer

Der Anschluss eines Beamers an einem Laptop erweist sich als große Hilfe, um gemeinsam ein Dokument zu betrachten. Hierfür muss das entsprechende Kabel aber jederzeit an dem Laptop angeschlossen bleiben. Hat nun eine zweite Person ebenfalls ein Dokument, welches alle Personen im Raum betrachten sollen, so muss das Anschlusskabel vom ersten Laptop entfernt und an den zweiten angeschlossen werden. Dieser Aufwand mag einmal erträglich sein, beim zweiten oder dritten Wechsel führt dieses aber zur deutlichen Arbeitsbehinderung. Das gleichzeitige Betrachten beider Dokumente ist dabei gänzlich ausgeschlossen.

Gemeinsames Bearbeiten von Dokumenten

Scheinbar unmöglich ist auch das gemeinschaftliche Bearbeiten eines Dokumentes. An einem Whiteboard können mehrere Personen jeweils einen Stift nehmen, an einem Computer ist aber nur eine einzige Maus und Tastatur vorhanden. Es müssten sich alle Personen an diesem einen Computer befinden.

Verwendung nicht-elektronischer Medien

Flipcharts und Whiteboards scheinen ein komfortables Mittel zu sein, um schnell und gemeinschaftlich Lösungen zu erarbeiten. Die Ergebnisse können aber nicht in elektronischer Form gespeichert werden. Um das Geschriebene oder Skizzierte später weiter benutzen oder gar an die anderen Personen verteilen zu können, muss dieses erst abgemalt bzw. abgeschrieben werden.

1.2.2. Vision

Ziel ist der Aufbau einer kollaborativen Arbeitsumgebung, welche Projektteams aus unterschiedlichen Bereichen als Plattform für die gemeinsame Arbeit dienen soll. Die Arbeitsumgebung hat das vorrangige Ziel, die gemeinschaftliche Arbeit zu fördern und sie effizienter und nachhaltiger zu gestalten. Zusätzlich sollen die Vorteile des klassischen Besprechungsraumes in dieser Umgebung wiederzufinden sein, ohne dabei neue Probleme zu erschaffen.

Zu diesem Zweck wurde an der HAW-Hamburg die Projektgruppe Collaborative Workplace gebildet, die gemeinsam an der Umsetzung dieser Ziele arbeitet. Maßgeblich bei der Umsetzung ist dabei der von Mark Weiser geprägte Begriff des Ubiquitous Computings, wonach sich verschiedene Computer (Laptops, PDAs, Mobiltelefone, usw.) in die allgemeine Umgebung nahtlos integrieren.

Aus technologischer Sicht bilden eine Reihe von hochauflösender, etwa 42” großer TFT-Bildschirme die Schlüsselelemente dieses Besprechungsraumes. Auf Grund ihrer Größe kann der Bildschirminhalt von allen Teilnehmern im Raum sehr gut gesehen werden, somit erfüllen diese grundsätzlich die Aufgaben des Beamers. Die Bildschirme sind zudem berührungssensitiv, wodurch sie sich sehr gut als Eingabegeräte für alle mit der Maus durchführbaren Computerinteraktionen eignen. Mit entsprechender Software ist auch handschriftliches Schreiben und das Erstellen von Skizzen an den Bildschirmen durchführbar. Die Vorteile von Whiteboard und Flipchart sind durch die Bildschirme demnach ebenso abgedeckt.

Die Vision geht aber über die Integration der Bildschirme hinaus. Mitgebrachte Endgeräte wie Laptops, PDAs und Mobiltelefone sollen Teil des Raumes werden. Gemäß ihrer Fähigkeiten sollen diese Endgeräte zum Betrachten und Bearbeiten sowie zur Steuerung des Raumes genutzt werden können.

Durch die Nutzung der großen Bildschirme ergibt sich die Möglichkeit, gemeinsam nicht nur an skizzierten Modellen, sondern auch direkt an den elektronischen Dokumenten zu arbeiten. Die Software, mit der diese Dokumente bearbeitet werden, soll auch in diesem Raum zur Verfügung stehen. Der Raum integriert die Arbeitsumgebung der einzelnen Benutzer und bietet dabei die Möglichkeit des Austausches. Die Bildschirme sind das neue Anzeigemedium, im Fokus der Arbeit bleiben jedoch die Dokumente.

1.3. Aufbau der Arbeit

Kapitel 2 beschäftigt sich mit dem Szenario Stadtplanung, welches für den Aufbau des Besprechungsraumes gewählt wurde. Es wird analysiert wie der Ablauf von Stadtplanungsprojekten ist und welche Rolle Informationstechnologien dabei spielen. Auf Basis dieser Analyse wird ein erstes Modell entwickelt, wie stadtplanerische Arbeit im Rahmen einer kollaborativen Umgebung umgesetzt werden kann. Hierzu wird eine erste Vision über das zu entwickelnde System erstellt und der Zusammenhang der einzelnen fachlichen Komponenten dargestellt. Abschließend werden die Anforderungen identifiziert, welche die zu erstellende Software aus Sicht des Benutzers zu erfüllen hat.

In Kapitel 3 werden die technischen Grundlagen vermittelt, die zur Konstruktion der Middleware notwendig sind. Es werden die grundlegenden Konzepte verteilter Systeme beschrieben sowie eine Reihe von Technologien vorgestellt, die diese Konzepte umsetzen und für die Realisierung des Konferenzraumes in Betracht gezogen werden können. Abschließend werden kurz einige verwandte Projekte vorgestellt, die sehr ähnliche Ziele wie das Collaborative Workplace verfolgen. Es werden dabei auch die Technologien vorgestellt, die in diesen Projekten verwendet werden.

Kapitel 4 stellt das Design und die Realisierung des Besprechungsraumes vor. Im Rahmen der Systemkonfiguration werden dabei die Hardwarekomponenten vorgestellt, die in diesem Raum benutzt werden. Im Abschnitt Interprozesskommunikation werden die Nachrichten identifiziert, die unter den Geräten ausgetauscht werden und eine Reihe möglicher Modelle miteinander verglichen, wie der Austausch von Nachrichten im Sinne des Ubiquitous Computings realisiert werden kann. Abschließend wird der Aufbau der Client- sowie der Serveranwendung erläutert.

Kapitel 5 fasst die Ergebnisse dieser Ausarbeitung noch einmal zusammen und gibt einen Ausblick über notwendige bzw. mögliche zukünftige Entwicklungen rund um das Projekt Collaborative Workplace.

2. Analyse

Dieses Kapitel befasst sich mit einer fachlichen Analyse des für diese Arbeit gewählten Szenarios, der Stadtplanung. Die Analyse beginnt mit einer Beschreibung der Tätigkeiten in der Stadtplanung. Davon ausgehend wird gezeigt, welche Rolle moderne Informationstechnologien in der Stadtplanung spielen.

Die Ergebnisse dieser Analyse werden auf die Arbeit in einer kollaborative Arbeitsumgebung transferiert. Es wird beschrieben, wie diese Umgebung sinnvollen Einsatz in der Stadtplanung findet. Aus den Erkenntnissen bezüglich der Stadtplanung werden konkrete Anforderungen für die Arbeitsumgebung erstellt. Eine erste Vision soll vermitteln, wie die Anwendung genutzt wird und was die Anwendung für die Teilnehmer einer Besprechung bedeuten soll. Eine Übersicht der fachlichen Komponenten soll schließlich zeigen, wie die in der Stadtplanung genutzten Informationstechnologien im Zusammenhang mit einer kollaborativen Arbeitsumgebung genutzt werden können.

2.1. Beschreibung des Szenarios

2.1.1. Einführung

Ziel der Stadtplanung ist die nachhaltige Entwicklung des Lebensraumes der Menschen, unter Berücksichtigung diverser Aspekte wie soziale Strukturen, Umweltschutz, Anforderungen der Wirtschaft und vieler Weiterer. Zu diesem Zweck arbeiten Stadtplaner in der öffentlichen Verwaltung sowie in privaten Stadtplanungsbüros und erstellen, meist im Auftrag des Landes, eine Vielzahl von Plänen, in denen die Nutzung von Flächen (Flächennutzungsplan) und die Art der Bebauung (Bebauungsplan) festgelegt werden.

Stadtplanerische Arbeit ist in Deutschland eine sehr formelle Tätigkeit, reglementiert durch das Baugesetzbuch. Für die Stadtplanung von besonderer Bedeutung sind die Vorgaben zum Planungsverfahren und die Zulässigkeit von Bauvorhaben unter städtebaulichen Gesichtspunkten. Das Baugesetzbuch wird zusätzlich durch die Landesbauordnungen der jeweiligen Länder ergänzt. Ein Überblick über den Inhalt und die Eckpunkte der gesetzlichen Vorgaben ist in [Albers 2002] zu finden.

Aus diesen gesetzlichen Vorschriften über die Zulässigkeit von Bauvorhaben, bzw. den Vorgaben über das Planungsverfahren lässt sich auch der Nutzen einer kollaborativen Arbeitsumgebung ableiten. Die Erstellung von Bauleitplänen (Flächennutzungspläne und Bebauungspläne) ist im Wesentlichen ein Gruppendiskussionsprozess. Stadtplaner müssen grundsätzlich mit verschiedenen Behörden und Institutionen, die selbst Interessen oder bestimmte Zuständigkeiten im Planungsgebiet haben, zusammen arbeiten. Diese so genannten Träger öffentlicher Belange, die ebenfalls durch den Gesetzgeber festgelegt sind, sind in den Stadtplanungsprozess mit einzubinden, mit dem Ziel einen Konsens zwischen allen Beteiligten zu erreichen. Hierzu gehören u. a. weitere Staatsbehörden, wie das Straßenbauamt oder Wasserwirtschaftsamt, Energieversorgungsunternehmen, Kirchen, Industrie- und Handelskammer und Bürger. Nach Unterrichtung über die allgemeinen Planungsvorhaben sitzen Vertreter der betreffenden Gruppen zusammen mit Kommunalpolitikern und den Stadtplanern an einem Tisch und arbeiten das Detailkonzept aus. In einer Vielzahl an Sitzungen werden Pro und Kontra einzelner Alternativen gegeneinander abgewogen, so dass am Ende ein Gesamtkonzept entsteht, welches durch die Landespolitik beschlossen und somit rechtskräftig wird.

2.1.2. Projektphasen

Im Laufe eines Stadtplanungsprojektes gibt es zwei verschiedene Phasen, welche es zu unterscheiden gilt. Ein Projekt beginnt grundsätzlich mit der Planungsvorbereitung. In der Planungsvorbereitung wird eine Bestandsanalyse durchgeführt und das Problem bzw. der Handlungsbedarf definiert. Zusätzlich werden erste Lösungsansätze entwickelt. Die zweite Phase, die Planungsphase, besteht aus einer Vielzahl an Planungsdiskussionen, in der die Planungsgruppe Details zu den Lösungen entwirft und über die Vor- und Nachteile der einzelnen Alternativen diskutiert. Zwischen den einzelnen Planungsdiskussionen findet dabei eine aufwändige Vor- bzw. Nachbereitung der Diskussion statt. Diese bestehen z. B. aus dem Erstellen neuer Karten gemäß der Planungsideen oder dem Prüfen rechtlicher Aspekte verschiedener Planungsalternativen.

Der eigentliche Planungsprozess kann stark vereinfacht als ein linearer Vorgang dargestellt werden. Die wesentlichen Schritte dabei sind das Entwerfen von Plänen, das Abschätzen der Folgen und das Bewerten und Abwägen der Pläne. Im Laufe der Planung ist es jedoch üblich, dass zu einem der vorigen Schritte zurückgesprungen wird, sobald festgestellt wird, dass durch diese Lösung nicht das gewünschte Ziel erreicht werden kann. Abb. 2.1 zeigt die einzelnen Schritte im Planungsprozess und ihre Rückkopplungen. Eine detaillierte Beschreibung über die Methodik bei den einzelnen Planungsabschnitten ist in [Miese 1980] zu finden.

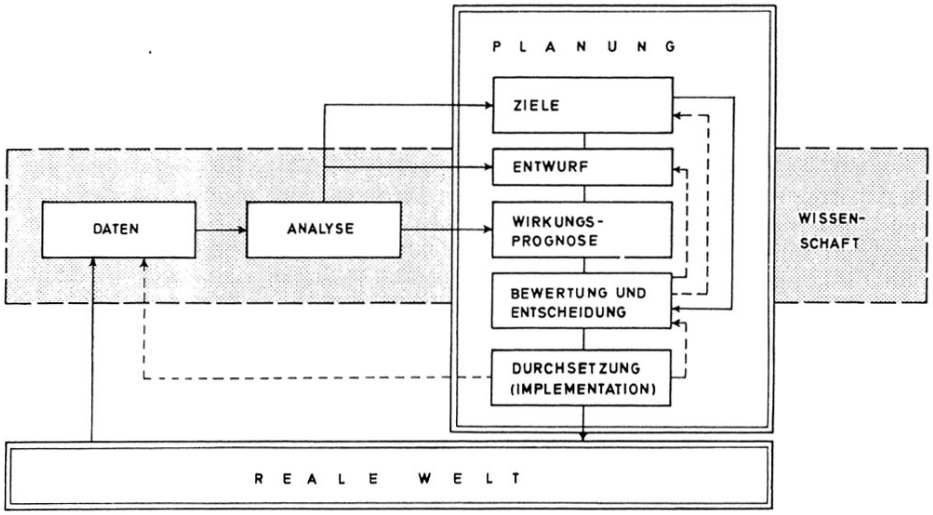

Abbildung 2.1.: Der Planungsprozess nach Britton Harris [Harris 1967]

2.1.2.1. Planungsvorbereitung

Bestandsaufnahme

Zu Beginn eines Stadtplanungsprojektes gilt es eine Bestandsanalyse durchzuführen und somit den Ist-Zustand zu definieren. Die Bestandsanalyse besteht aus der Erfassung und Analyse der Gegebenheiten, bezogen auf soziale, wirtschaftliche und räumliche Aspekte. Die Bestandsanalyse basiert auf der kartografischen und statistischen Erfassung der gegebenen Umweltverhältnisse. Bei der Bestandsanalyse gilt es u. a. folgende Aspekte zu berücksichtigen: Bebauung (Nutzen, Zustand, Alter), Einwohnerdichte, Topographie, Vegetation und Verkehrsinfrastruktur.

Die meisten dieser Informationen sind in Form von gedruckten oder elektronischen Karten vorhanden. Elektronisches Kartenmaterial ist dabei den Geografischen-Informations-Systemen (GIS) zu entnehmen. Zum besseren Verständnis von den Gegebenheiten des Planungsgebietes werden Fotos von besonderen Plätzen des Gebietes erstellt. Sowohl Luft-, als auch Bodenaufnahmen von Straßenzügen, Plätzen, Flächen und besonderen Gebäuden gehören zu den Materialien über das Planungsgebiet.

Zusätzlich zur Analyse der aktuellen Situation, gehört auch die Entwicklung in der Vergangenheit zur Bestandsaufnahme. Um zukünftige Entwicklungen vorauszusagen, gilt es die vergangene Entwicklung zu erkennen und nachzuvollziehen. Nur wenn die einzelnen Zusammenhänge in der Entwicklung verstanden werden, ist es möglich, die zukünftige Entwicklung erfolgreich durch Maßnahmen zu steuern.

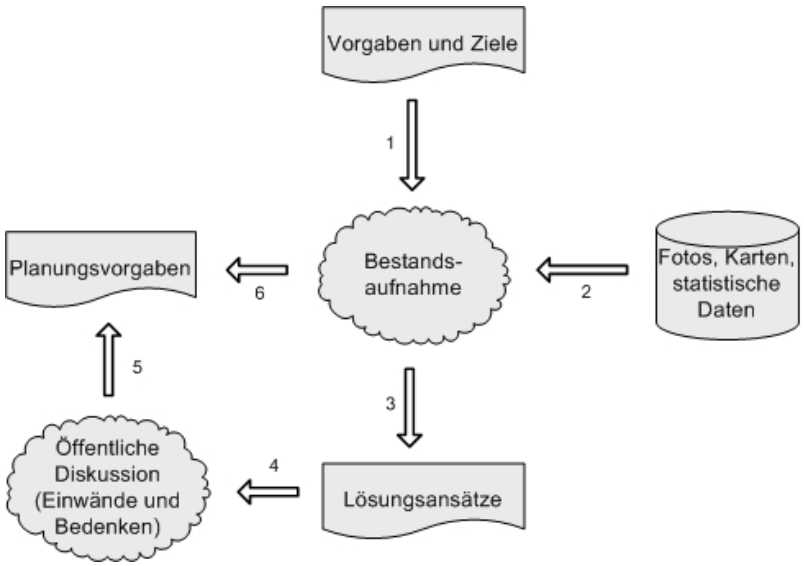

Abbildung 2.2.: Workflow der Planungsvorbereitung

Albers fasst in [Albers 2002] den Vorgang der Bestandsaufnahme in vier Fragen zusammen, welche es zu klären gilt:

- Was besteht?
- Wie verändert es sich?
- Wie hängt es zusammen?
- Wie wird es sich voraussichtlich weiterentwickeln?

Abbildung 2.2 zeigt den Ablauf der Planungsvorbereitung. Zur ersten Bestandsaufnahme sind die (behördlichen) Vorgaben zu ermitteln und die Planungsziele zu definieren (1). Fotos, Karten und statistische Daten (2) werden hier analysiert. Ergebnis der Bestandsaufnahme sind die Planungsvorgaben (6). Es werden zudem erste Lösungsansätze (3) entwickelt. Die Lösungsansätze werden veröffentlicht (4), so dass Bürger und Träger öffentlicher Belange hierzu Stellung nehmen können (5).

Träger öffentlicher Belange

Von besonderer Bedeutung für Stadtplanungsprojekte sind die Träger öffentlicher Belange. Dies sind die von dem Planungsvorhaben direkt oder indirekt betroffenen Behörden und Institutionen. Es ist unerlässlich, dass die Planung mit Fachkräften aus den einzelnen Behörden und Institutionen abgesprochen wird. Bereits zu Projektbeginn müssen diese Stellen identifiziert und über das Planungsvorhaben informiert werden.

Das Baugesetzbuch schreibt vor, dass die Träger öffentlicher Belange zu einer Stellungnahme bzw. zur Mitarbeit an der Planung verpflichtet sind, soweit Planungsaspekte in deren Zuständigkeitsbereich liegen. Eine Liste mit verschiedenen Planungsaspekten und deren zuständiger Träger öffentlicher Belange in Anhang A zu finden.

2.1.2.2. Planungsdiskussion

Grundlegendes Ziel der Planungsdiskussion ist die Erstellung eines in der Gruppe konsensfähigen Konzeptes im Sinne der Planungsvorgaben. Die Planungsgruppe besteht i. A. aus Stadtplanern, Abgesandten der Träger öffentlicher Belange und evtl. Bürgervertretern[1]. Diese treffen sich zu einer Reihe von Planungsdiskussionen, bei denen die einzelnen Probleme diskutiert, Lösungsvorschläge gesammelt und gegeneinander abgewogen werden. Dabei kann der ganze Planungsvorgang, je nach Umfang des Projektes, einige Monate bis hin zu einigen Jahren andauern. Gängige Praxis ist es dabei, dass die einzelnen Planungstreffen im Abstand von wenigen (2-5) Wochen stattfinden. Für die einzelnen Planungstreffen werden vorher die jeweiligen Themen- und Arbeitsschwerpunkte festgelegt, so dass einzelne Treffen entsprechend vorbereitet werden können. Zu einzelnen Terminen können zusätzliche Experten, die nicht direkt zu den Trägern öffentlicher Belange gehören, mit in die Planungsgruppe einbezogen werden.

Abbildung 2.3 zeigt den schematischen Ablauf der Planungsphase. Die identifizierten Planungsziele (1) und die vorliegenden Fotos, Karten und Daten (4b) bilden die Basis der Diskussion. In dieser werden, meist über ein Brainstorming, verschiedene Ideen gesammelt. Aus den Planungstreffen gehen konkrete Arbeitsaufträge (2) hervor. Hierzu gehört z. B. das Einholen fachlicher Gutachten (3a) und das Erstellen neuer Karten, entsprechend den Planungsergebnissen (4a). Beides fließt in zukünftige Treffen wieder mit ein (3b)(4b). Die Ergebnisse der Planungstreffen werden grundsätzlich protokolliert (5).

Zur weiteren Veranschaulichung des Diskussionsprozesses in der Stadtplanung sei hier auf ein öffentliches Protokoll über die Entstehung eines Verkehrskonzeptes des Stadtteils Hütteldorf der Stadt Wien, zu finden in [Verkehrskonzept Hütteldorf 2002], verwiesen.

2.1.3. Computergestützte Arbeit in der Stadtplanung

Wie bereits den Abbildungen 2.2 und 2.3 zu entnehmen ist, werden im Laufe eines Stadtplanungsprojektes verschiedene Dokumente erzeugt.

[1] nur bei öffentlichen Planungsverfahren

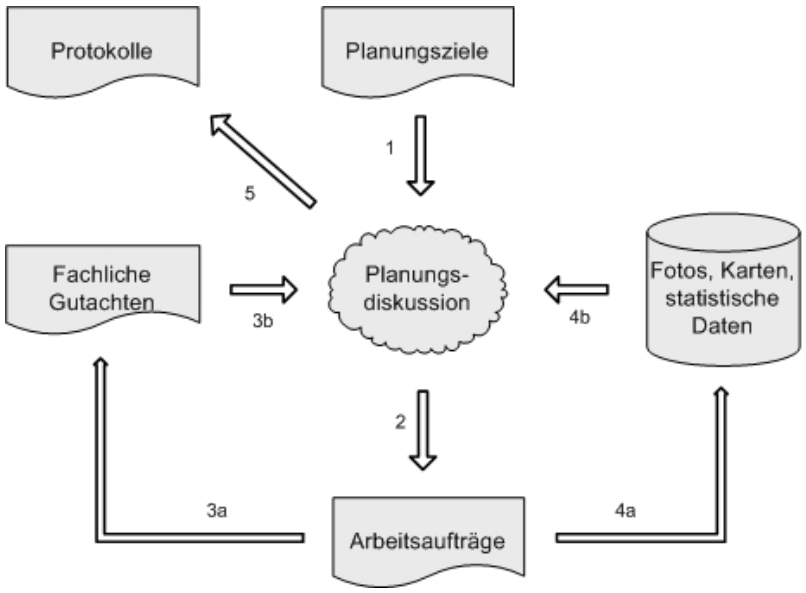

Abbildung 2.3.: Workflow der Planungsphase

Neben den typischen Büroanwendungen (Textverarbeitung, Tabellenkalkulation, Präsentation) spielen Anwendungen für Fotos, Karten, (raumbezogene) statistischen Daten und 2- bzw. 3-dimensionale Stadt- und Landmodelle in der Stadtplanung eine besondere Rolle.

Fotos und Karten werden vielfach als Rastergrafiken (Bitmap, JPEG, TIFF) gespeichert. Deren Bearbeitung erfolgt über Standardsoftware zur Bildbearbeitung z.B. Adobe Photoshop. Mit Programmen für GIS und CAD (Computer Aided Design) kommen in der Stadtplanung aber auch sehr individuelle Softwarepakete zum Einsatz. GIS-Programme beinhalten geografische Informationen, die in einer sehr umfangreichen Datenbank gespeichert werden. In dieser Datenbank sind den einzelnen geografischen Flächen eine Vielzahl alphanumerischer Daten zugeordnet. Die Anzeige einer Karte gleicht in diesem Fall einer umfangreichen Datenbankabfrage. GIS ist somit vor allem ein Werkzeug zur Analyse von Daten. Demgegenüber stehen CAD-Programme, die als Konstruktions- und Zeicheninstrument in der Stadtplanung zur Erstellung von Modellen und Anfertigung von Plänen genutzt werden. Zu den kartenorientierten Programmen kommen noch Programme zur statistischen Auswertung von Daten. Hier ist vor allem das gleichnamige Produkt der Firma SPSS von sehr großer Bedeutung. Weitere Informationen über die Verwendung von GIS- und CAD-Programmen wurden von der Arbeitsgemeinschaft EDV in der Stadtplanung in [EDV-Stadtplanung 1999] zusammengestellt.

Die Tabelle 2.1 fasst die wichtigsten in Stadtplanungsbesprechungen vorkommenden Dokumente noch einmal zusammen.

Dokument	Dokumenttyp
Protokolle	Text
Planungsziele	Text
Gutachten	Text
Wirkungsprognosen	Text
Vorträge / Referate	Präsentation
Planungsentwürfe	CAD / Text
Geographische Informationen	GIS / CAD
Statistische Daten	GIS / SPSS / Tabellen
Digitale Fotos	Rastergrafik
Karten (Papier)	*nicht elektronisch*
Fotos (Papier)	*nicht elektronisch*

Tabelle 2.1.: Verwendete Dokumente und ihre Dokumenttypen

2.2. Stadtplanung in kollaborativen Umgebungen

2.2.1. Allgemeine Verwendung des Konferenzraumes

Die stadtplanerische Arbeit ist in erster Linie ein Diskussionsprozess. Dieser Diskussionsprozess wird begleitet von einer Vielzahl an Dokumenten, wie in Tabelle 2.1 dargestellt. Zum Anzeigen dieser Dokumente ist entsprechende Software über die öffentlichen Bildschirme[2] bedienbar. Für den Diskussionsprozess ist vorwiegend die Betrachtung von Karten und Bildern wichtig. Hierfür eignen sich vor allem die an der Wand montierten Bildschirme. Für das Erstellen neuer Karten mit Hilfe von CAD-Programmen eignet sich besonders der in den Konferenztisch eingebaute Bildschirm. Um diesen herum können jedoch nur maximal acht Personen stehen. Während der Diskussion erstellte Protokolle können im Anschluss an die Diskussion direkt über die öffentlichen Bildschirme eingesehen werden bzw. sofort an die Teilnehmer in elektronischer Form verteilt werden.

Neben dem Betrachten und Bearbeiten von Dokumenten sind die großen Bildschirme besonders für eine Anwendung zum Durchführen eines Brainstormings gut geeignet. Einzelne Stichworte können, z. B. über eine verteilte Anwendung, gesammelt und an einem der Bildschirme angezeigt werden. Das Gruppieren der Stichworte wird direkt durch das Ziehen der einzelnen Objekte am Bildschirm ermöglicht.

Durch die Vielfalt der Bilder und Karten, die im Rahmen des Diskussionsprozesses betrachtet werden müssen, ist die Verwendung von beschreibenden Metadaten zu diesen Dokumen-

[2] Der Begriff öffentliche Bildschirme bezeichnet die in der Einleitung erwähnten ca. 42” großen, hochauflösenden Bildschirme. Der Begriff wird im weiteren Verlauf auch als Synonym für den Computer benutzt, an welchem der Bildschirm angeschlossen ist.

ten von großer Bedeutung. Dateinamen sind meist nicht aufschlussreich genug, um direkt auf den Inhalt der Dokumente zu schließen. Besonders für Personen, die das Dokument niemals zuvor betrachtet haben, ist der Einsatz von Metabeschreibungen sinnvoll. Ohne beschreibende Metadaten ist das einzige Mittel die Festlegung von Ordnerstrukturen und Dateinamenkonventionen, um die Dokumente systematisch zu ordnen. Dieses Verfahren ist jedoch auf die Einhaltung durch die Mitwirkenden angewiesen, weshalb dieses Verfahren nur bei Gruppen von 2-3 Personen anwendbar ist. Durch die Definition von Attributen zu den Dokumenten wird sich die Suche nach diesen ebenfalls einfacher gestalten.

2.2.2. Verwendung von GIS-Programmen

Ein besonderes Problem für den Aufbau des Konferenzraumes stellen die GIS-Programme dar. Nach einer in [Coors und Zipf 2005] publizierten Marktübersicht, sind in Deutschland derzeit mehr als 20 verschiedene GIS-Programme im Einsatz. Des weiteren beziehen GIS-Anwendungen ihre Daten von speziellen Geodatenservern. Die Daten sind jedoch nicht standardisiert, GIS-Anwendungen sind somit abhängig von dem jeweils genutzten Server. Für die Integration von GIS-Systemen in das Collaborative Workplace bieten sich vor allem Webbasierte GIS-Systeme an. Zu deren Bedienung ist keine Installation spezieller Software auf den Clients notwendig. Es wird lediglich ein Webbrowser benötigt. Webbasierte GIS-Anwendungen sind i. A. in ihrer Funktionalität gegenüber nicht-webbasierten Anwendungen limitiert.

Das im Auftrag des Bundesministeriums für Wirtschaft und Technologie gestartete Projekt XPlanung beschäftigt sich mit der Standardisierung von Geodaten. Ziel des Projektes ist die Vereinheitlichung von Geodaten für den elektronischen Datenaustausch zwischen den Ländern und Kommunen in Deutschland. Vorrangig ist dabei vor allem die Vereinheitlichung von Flächennutzungs- und Bebauungsplänen [XPlanung 2003]. Die Ergebnisse dieses Projektes könnten die Verwendung einer GIS-Anwendung innerhalb des Konferenzraumes erleichtern bzw. deren Einsatz für weitere Gruppen aus dem Bereich Stadtplanung ermöglichen.

2.2.3. Analyse der Besprechungsteilnehmer

Wie in Abschnitt 2.1.2.2 erläutert, nehmen an den Besprechungen sowohl Experten aus dem Bereich Stadtplanung sowie verschiedene Träger öffentlicher Belange teil. Die teilnehmenden Stadtplaner sind dabei als Fachexperten im Umgang mit der verwendeten Software einzustufen. Es kann auch davon ausgegangen werden, dass diese über mobile Endgeräte verfügen, die sie in den Besprechungsraum integrieren.

Bei den Trägern öffentlicher Belange lässt sich dagegen keine direkte Aussage über deren Erfahrungen mit Computeranwendungen machen. Sie sind Fachexperten ihres Bereiches, jedoch muss vereinzelt davon ausgegangen werden, dass sie weder mobile Endgeräte mitbringen können, noch dass sie mit dem Umgang dieser sehr vertraut sind. Da die Besprechungen lediglich im Abstand mehrer Wochen stattfinden, ist auch die Möglichkeit der Einarbeitung nicht unbedingt gegeben. Damit sich diese Personen dennoch mit eigenen Dokumenten einbringen können, muss ihnen für die Dauer der Besprechung ein entsprechendes Gerät zur Verfügung gestellt werden. Dieses wird für den weiteren Verlauf der Arbeit als Rahmenbedingung festgesetzt. Durch das möglicherweise geringe technische Vorwissen und die geringen Einarbeitungsmöglichkeiten ergeben sich für die Konferenzraumsoftware besonders hohe Ansprüche an die intuitive Bedienbarkeit.

Neben dem für diese Arbeit gewählten Szenario Stadtplanung wurde die Entwicklung des Konferenzraumes bereits für die Szenarien Besprechungen im Rahmen von Softwareentwicklungsprojekten und Besprechungen in einer Notfallleitzentrale untersucht. Carola Neumann beschreibt in [Neumann 2006] die Tätigkeit bei Softwareentwicklungsprojekten als eigenverantwortliche Expertentätigkeit. Demzufolge verfügen Softwareentwickler über das technisches Wissen für die Bedienung der Software im Konferenzraum. Die technische Ausstattung in Form von mobilen Endgeräten kann als wahrscheinlich angenommen werden. Bei der Verwendung in einer Notfallleitzentrale stellen sich besondere Anforderungen an die Verfügbarkeit des Systems. Das System dient in diesem Fall der Koordinierung von Notfalleinsätzen der Berufsfeuerwehr. In diesem Szenario, beschrieben von Andreas Piening in [Piening 2006], kann die Tätigkeit ebenfalls als Expertentätigkeit eingestuft werden. Für die Benutzung des Systems werden eigens Schulungen durchgeführt.

Im Vergleich mit den anderen Szenarien ist das Stadtplanungsszenario weniger auf die Verfügbarkeit des Systems angewiesen, jedoch sind die Benutzer des Systems mit diesem am wenigsten vertraut. Die Akzeptanz des Systems ist stark von der Bedienbarkeit und den funktionalen Möglichkeiten abhängig.

2.2.4. Systemidee

Die zu erstellende Anwendung Kora besteht aus einer Server- und einer Clientkomponente. Die Benutzer müssen die Clientkomponente auf ihrem eigenen Gerät installieren. Eventuell notwendige Konfigurationen nimmt das System selbstständig vor. Nur im Zweifel ist die Hilfe des Benutzers notwendig. Für die Verteilung der Software, wird diese im lokalen Intranet zur Verfügung gestellt. Updates der Software nimmt diese selbstständig beim Start der Anwendung vor.

Die Anwendung Kora versteht sich als Metabetriebssystem. Für den Besprechungsteilnehmer ist die Anwendung ein Desktopersatz. Für die Dauer des Aufenthaltes im Bespre-

chungsraum ist dieses die zentrale Anwendung für jegliche Arbeit mit Dokumenten. Von diesem Desktop aus werden alle dokumentenorientierten Aktionen durchgeführt. Der Anwender kann dabei wählen, ob er ein Dokument auf dem eigenen Gerät oder auf einem der öffentlichen Bildschirme anzeigen möchte. Die Anwendung ist neben der Anzeige der Dokumente auch für deren Verwaltung zuständig. Für den Im- und Export der Dokumente dient die Anwendung dem Benutzer als Schnittstelle zum eigenen System. Darüber hinaus integriert Kora eine Reihe weiterer Anwendungen, bzw. startet diese aus Kora heraus. Für den Im- und Export von Dokumenten bzw. Dokumentmetadaten steht eine weitere Anwendung zur Verfügung, mit der von außerhalb des Raumes Zugriff auf das System genommen werden kann.

Die Benutzer der Anwendung sind die Teilnehmer einer Besprechung. Entweder besitzen diese ein Endgerät oder bekommen eines gestellt. Die Anwendung wird aber nicht nur auf den Geräten der Besprechungsteilnehmer ausgeführt, sondern läuft auch auf den öffentlichen Bildschirmen. Somit ist jeder Besprechungsteilnehmer Benutzer des Systems. Wie im vorigen Abschnitt erläutert, wird das System nicht nur von Experten, sondern auch von Gelegenheitsanwendern genutzt. Die Akzeptanz des Systems wird stark von der intuitiven Bedienbarkeit des Systems abhängen. Die externen Anwendungen, die von Kora integriert werden sind vor allem den Fachexperten geläufig. Eine weitere Benutzergruppe sind Administratoren des Systems, diese sind zuständig für das Einrichten von Projekten und Benutzern. Administratoren benutzen das System nicht über die Anwendung Kora, sondern über eine eigene Administrationsanwendung.

Die einzelnen Clients interagieren untereinander, um Dokumente auszutauschen. Die Clients interagieren auch mit den öffentlichen Bildschirmen, um die Fernsteuerung dieser zu ermöglichen. Die Interaktion erfolgt dabei eventgesteuert. Die Kontaktaufnahme wird selbstständig vom System durchgeführt. Das System hat jederzeit Kenntnis über die weiteren verfügbaren Clients und die öffentlichen Bildschirme. Für die Interaktion bieten die Clients und die öffentlichen Bildschirme einzelne Dienste an. Diese Dienste bestehen in erster Linie aus dem Anzeigen und Bearbeiten von Dokumenten.

2.2.5. Fachliche Komponenten

Zentrales Element der fachlichen Komponenten sind die Projektdaten und -dokumente. Sämtliche Anwendungen beziehen hieraus ihre Daten. Zu diesen Anwendungen zählt vor allem eine GIS-Anwendung, CAD-Anwendung und SPSS. Hinzu kommen weitere Programme wie ein Büroanwendungspaket, eine Anwendung zum Durchführen eines Brainstormings sowie ein Webbrowser. Die Grenzen zwischen GIS- uns CAD-Anwendungen werden nicht fest definiert. Häufig ist dieses eine einzige Anwendung, mit verschiedenen

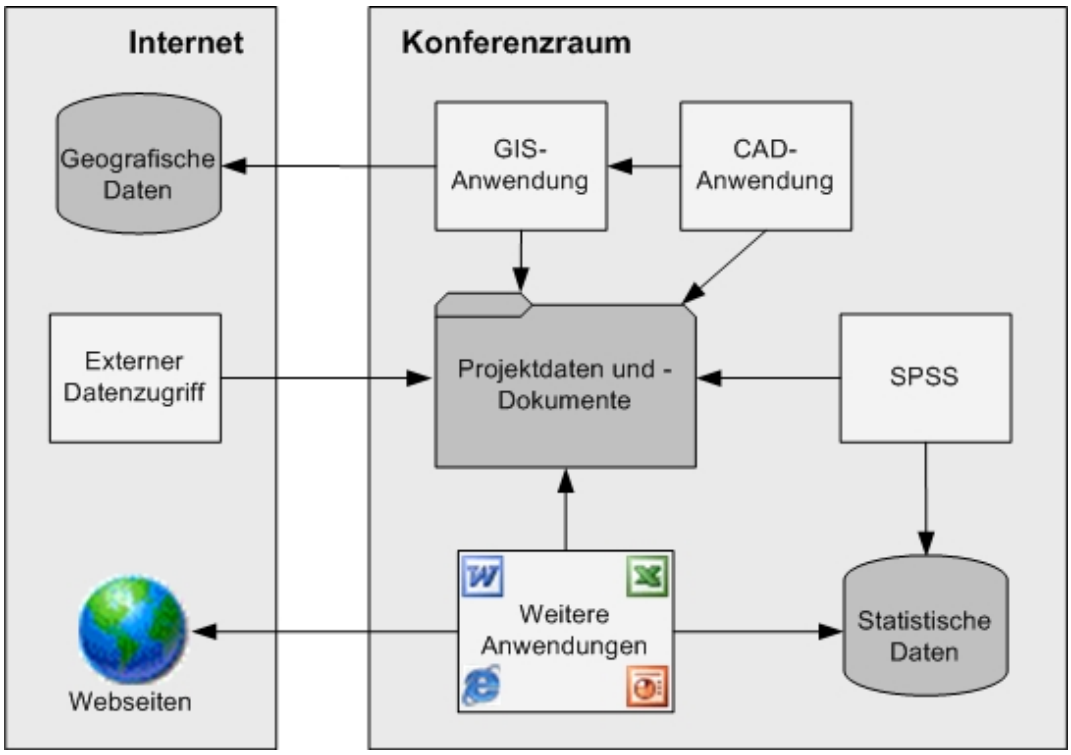

Abbildung 2.4.: Fachliche Komponenten

Ansichten auf Karten und Bildern. GIS-Anwendungen nutzen dabei mindestens eine Geodatenbank, aus welcher sie ihre Daten beziehen. CAD-Anwendungen importieren diese Daten aus GIS-Anwendungen. SPSS bezieht seine Daten aus einer statistischen Datenbank. Ein internes Data-Warehouse integriert ebenfalls diese Daten. Abbildung 2.4 stellt den Zusammenhang der Komponenten noch einmal grafisch dar.

2.3. Anforderungen

Aus dem vorangegangenen Beispiel über die Arbeit der Stadtplanung mit Hilfe eines Konferenzraumsystems sollen in diesem Abschnitt konkrete Anforderungen an das System identifiziert werden. Grundsätzlich wird dabei zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden.

2.3.1. Funktionale Anforderungen

2.3.1.1. Dokumente verwalten

Synchronisierung

Die Arbeit in der kollaborativen Arbeitsumgebung nimmt nur einen Teil der Projektarbeit ein. Dokumente werden vielfach außerhalb der Arbeitsumgebung erstellt bzw. bearbeitet. Um sämtliche projektrelevante Dokumente serverseitig zu speichern, ist eine Synchronisierung der Dokumente auf dem PC des Benutzers mit den serverseitig gespeicherten Dokumenten notwendig. Zur Synchronisierung der Dokumente gehört sowohl das Speichern von Kopien öffentlicher Dokumente auf dem Computer des Benutzers sowie das Kopieren der vom Benutzer offline bearbeiteten Dokumente auf den Server.

Die Synchronisierung wird automatisch ausgeführt sobald sich der Benutzer am System anmeldet. Sie kann aber auch manuell durch den Benutzer gestartet werden. Dies kommt dann zum Tragen, wenn der Benutzer vergessen hat ein Dokument dem Projekt zuzuordnen oder dieses erst während des Aufenthaltes im Konferenzraum beschließt.

Die Synchronisierung wird ebenfall durchgeführt wenn der Benutzer eine Datei mit einer externen Anwendung bearbeitet und anschließend gespeichert hat. Die Synchronisierung kann dabei vom System ausgehen oder explizit durch den Benutzer angestoßen werden. Damit diese vom System ausgehen kann, muss dieses entweder erkennen dass die Datei verändert wurde oder die verwendete Anwendung führt selbstständig die Synchronisation durch.

Veröffentlichung

Dokumente sind grundsätzlich Eigentum ihres Erstellers und somit nicht öffentlich zugänglich. Es gibt zwei Möglichkeiten Dokumente an andere Personen zu verteilen:

- Dokumente werden explizit an andere Benutzer weitergegeben. Das bedeutet, dass der empfangende Benutzer eine Kopie des Dokumentes erhält. Auf das kopierte Dokument hat der Benutzer volles Zugriffsrecht.
- Anderen Personen wird ein Leserecht auf das Dokument gewährt. Der Benutzer kann zwar durch den Export eine lokale Kopie auf dem eigenen PC verändern, diese Änderungen können jedoch nicht wieder zurückgeschrieben werden.

Metainformationen

Zu jedem Dokument werden sehr umfangreiche Metadaten gespeichert. Der Benutzer muss die Möglichkeit haben, auf diese Metadaten zuzugreifen und diese ggf. zu bearbeiten, sofern der Benutzer die entsprechenden Rechte hierfür besitzt. Die Metadaten werden dabei in zwei verschiedene Kategorien aufgeteilt:

Sicherheitsrelevante Metainformationen in Form eines Rechtemodells zur Steuerung der Zugriffsrechte (lesende und schreibende Zugriffe) auf ein Dokument.

Inhaltliche Metainformationen wie Dokumentbeschreibungen und Verweise zu verwandten Dokumenten. Dateinamen sind häufig nicht aussagekräftig genug. Besonders bei Bildern und Karten kann der Nutzen von Metainformationen sehr groß sein. Beschreibungen zu dem Inhalt und den Zusammenhängen mit anderen Karten, Bildern, statistischen Daten und Texten können vom Benutzer gelesen werden, ohne dass das Dokument selbst geöffnet werden muss.

Mit dem Thema der sicherheitsrelevanten Metainformationen in kollaborativen Umgebungen hat sich insbesondere Horst Mund in [Mund 2006] beschäftigt.
Mit den inhaltlichen Metainformationen und deren möglicher Darstellung beschäftigt sich Petra Ehlers in [Ehlers 2006].

Dokumentbrowser

Der Benutzer muss die Möglichkeit haben, nach Dokumenten im gemeinsamen Datenspeicher zu suchen. Es muss eine Abbildung der Dokumente anhand von Verzeichnisstrukturen geben. Diese Darstellungsform ist dem Benutzer von seinem Betriebssystem vertraut und wird daher große Akzeptanz finden. Durch die Einführung der Metainformationen ergibt sich auch die Möglichkeit einer dem Inhalt nach strukturierten Darstellung der Dokumente. In dieser Darstellungsform lassen sich themenverwandte Dokumente sehr schnell auffinden. Diese Darstellungsform eignet sich besonders bei größeren Gruppen, die nicht ständig an einem Projekt zusammenarbeiten.

Versionierung

Alle Dokumente sollen grundsätzlich der Versionierung unterliegen. Dieses beinhaltet auch die Möglichkeit auf ältere Dokumente zurückzugreifen. Bei Textdateien sollte generell die Möglichkeit bestehen, die Unterschiede zwischen zwei Versionen anzuzeigen.

2.3.1.2. Dokumente bearbeiten

Bearbeiten an einem öffentlichen Bildschirm

Der offensichtlichste Anwendungsfall in diesem Konferenzraum ist das Betrachten bzw. Bearbeiten eines Dokumentes an einem öffentlichen Bildschirm. Hier kann der Benutzer an seinem eigenen Gerät wählen, an welchem der verfügbaren Bildschirme das Dokument geöffnet werden soll. Die Kontrolle über das Dokument wird von dem Moment an, an den Rechner übermittelt, der das Dokument anzeigt. Von diesem aus wird das Dokument auch wieder gespeichert.

Das Dokument bleibt grundsätzlich Eigentum des Benutzers, der es geöffnet hat. An einem öffentlichen Bildschirm kann dieses aber von jeder Person im Raum arbeitet werden.

Für die Verwendung der öffentlichen Bildschirme, muss der Teilnehmer auf seinem eigenen Gerät eine Liste der verfügbaren Bildschirme angezeigt bekommen.

Bearbeiten am eigenen PC

Analog zum Bearbeiten an einem öffentlichen Bildschirm kann der Benutzer das Dokument an seinem eigenen PC bearbeiten. Auch in diesem Fall öffnet der Benutzer das Dokument mit Hilfe der Konferenzraumanwendung. Über diese ist der Benutzer auch in der Lage, das Dokument wieder zu speichern.

Neue Dokumente

Neben den verfügbaren Dokumenten muss dem Benutzer die Möglichkeit gegeben werden, aus der Konferenzraumanwendung heraus neue Dokumente zu erstellen. Diese können anschließend auf dem eigenen PC oder einem der öffentlichen Bildschirme bearbeitet und von dort aus auch gespeichert werden.

2.3.1.3. Systemverwaltung

Mit Hilfe einer Systemverwaltung müssen die Projekte, Benutzer und Benutzergruppen angelegt, bearbeitet und gelöscht werden können. Auf die Systemverwaltung selbst muss es dabei unterschiedliche Sichten geben. Systemadministratoren haben Zugriff auf die Projektadministration. Projektadministratoren richten Benutzer und Benutzergruppen ein. Die Systemverwaltung ist dabei als eigenständige Anwendung zu sehen. Direkte Kommunikation mit der Konferenzraumanwendung ist nicht nötig. Es muss lediglich auf die Datenbank zugegriffen werden.

2.3.2. Nicht-funktionale Anforderungen

2.3.2.1. Authentifikation & Autorisation

In einer personalisierten Systemumgebung, in welcher die Daten vor dem Zugriff von unbefugten Personen geschützt bleiben sollen, muss sich der Benutzer dem System gegenüber authentifizieren. Dies wird bereits aus der dokumentorientierten Sicht der funktionalen Anforderungen deutlich. Hinzu kommt die Autorisation, die bei Zugriff auf Dokumente sowie der Nutzung der Dienste notwendig wird. Die Authentifikation wird ein einziges Mal beim Anmelden am System durchgeführt. Die Autorisation muss bei jedem Zugriff auf autorisationspflichtige Dienste durchgeführt werden.

Die Benutzer sind dabei in verschiedene Klassen zuzuordnen. Es gibt eine grundlegende Unterteilung in Projekte und innerhalb der Projekte verschiedene Rollen wie Administrator, Projektleiter, Protokollant und Gast. Für die Zuordnung der Rechte bedarf es eines rollenbasierten Rechtemodells, wie es Horst Mund in [Mund 2006] beschrieben hat.

2.3.2.2. Ubiquitous Computing

Die Nutzung des Systems muss sich ohne administrative Eingriffe durchführen lassen. Eine hieraus entstehende Anforderung ist die automatisierte Installation und Konfiguration des Systems. Die Autorisation am System sollte nur ein einziges Mal durchgeführt werden müssen. Bei einem kurzfristigen Verbindungsabbruch muss diese ohne Eingreifen des Benutzers wieder hergestellt werden. Selbiges gilt, wenn die Verbindungsunterbrechung durch den Ruhezustand des Gerätes erzwungen wurde.

Die von anderen Geräten angebotenen Dienste müssen selbstständig ausgetauscht und laufend aktualisiert werden. Dabei ist zu beachten, dass ein Benutzer auch an mehreren Geräten gleichzeitig eingeloggt sein kann. Bei Auftreten von Fehlern, darf das Gesamtsystem nicht beeinträchtigt werden.

2.3.2.3. Ergonomie

Das System sollte für den Benutzer leicht verständlich sein. Die Bedienung muss intuitiv und leicht erlernbar sein. Auf Grund der Zeitabstände zwischen den Besprechungen wird das System von einigen Benutzern nur selten genutzt werden. Die Lernkurve wird somit gering sein. Bei Auftreten von Fehlern müssen diese für den Benutzer verständlich sein. Diese Fehler dürfen auch nicht zum Blockieren der Anwendung führen. Im Allgemeinen sollte die Anwendung eine hohe Gebrauchstauglichkeit aufweisen.

Mit den Anforderungen an eine Benutzerschnittstelle für computergestützte Diskussionsprozesse hat sich Carola Neumann in [Neumann 2006] sehr ausführlich beschäftigt.

3. Technische Grundlagen

Zur Konstruktion einer Middleware für einen computergestützen Gruppenarbeitsraum ist es notwendig, einige technische Grundlagen im Bezug auf verteilte Systeme zu erläutern. In diesem Kapitel werden die grundlegenden Eigenschaften und Konzepte von verteilten Systemen beschrieben. Neben den theoretischen Aspekten werden dazu einige für diese Arbeit relevante Technologien vorgestellt. Abschließend wird eine kurzer Einblick in verwandte Projekte und die Technologien, die in diesen verwendet werden, gegeben.

3.1. Grundlagen verteilter Systeme

3.1.1. Einführung

In [Tanenbaum und van Steen 2002] wird ein verteiltes System folgendermaßen definiert:

> A distributed system is a collection of independent computers that appears to its users as a single coherent system.

Ein verteiltes System ist danach ein für den Benutzer als einzige Einheit erscheinendes System, bei dem im Hintergrund eine unbestimmte Anzahl Maschinen zusammenwirken. Es ist nicht ersichtlich, wie viele Maschinen in einem verteilten System miteinander interagieren, ebenso bleibt die Art der Maschineninteraktionen hinter der Anwendung verborgen. Ein verteiltes System kann aus einer homogenen Landschaft weniger Computer sowie aus einer heterogenen vieler Computer bestehen. Je nach Art und Einsatz der Konzepte kann das System unabhängig von Betriebssystem, Kommunikationsprotokoll und Programmiersprache sein.

Eine Reihe von Eigenschaften charakterisieren ein verteiltes System, die im Folgenden kurz erläutert werden.

Transparenz: Bereits aus der Definition von Tanenbaum kann man auf die Transparenz eines verteilten Systems schließen. Dem Benutzer gegenüber erscheint das System als eine Einheit, Details bleiben für ihn verborgen. Es ist für den Benutzer nicht nachvollziehbar, welche Maschinen an einem verteilten System beteiligt sind. Daraus ergibt

sich zwangsläufig eine Ortstransparenz, wodurch es dem Benutzer verborgen bleibt, wo Daten verarbeitet oder gespeichert werden. Durch die Ortstransparenz entsteht auch Fehlertransparenz. Wenn verschiedene Systeme zusammenarbeiten müssen, um eine bestimmte Aufgabe zu erledigen und eines dieser Systeme ausfällt, erscheint dieses, sofern keine detaillierte Fehlermeldung ausgegeben wird, für den Benutzer wie ein Ausfall des gesamten Systems.

Ressourcenteilung: Die Ressourcenteilung ist eines der wichtigsten Merkmale verteilter Systeme. Einzelne Ressourcen wie Drucker, Speichergeräte oder Datenbanken werden von einer bestimmten Instanz verwaltet und können einzelnen Benutzern zugewiesen werden.

Parallelität: Innerhalb eines verteilten Systems laufen verschiedene Prozesse gleichzeitig ab, ohne sich dabei gegenseitig zu beeinflussen. Bei der Ressourcenteilung muss daher auf die Koordinierung der Ressourcen Rücksicht genommen werden. Hierzu werden Sperren und Transaktionen verwendet.

Skalierbarkeit: Ein verteiltes System besteht grundsätzlich aus einzelnen Komponenten. Durch Aufteilung der Aufgaben lassen sich einzelne Komponenten leichter austauschen, erweitern oder weiter aufteilen. Durch Hinzufügen weiterer Ressourcen lässt sich somit ein Performancegewinn erzielen.

Das Fundament eines verteilten Systems ist die Middleware. Diese stellt Dienste für die darüberliegenden Anwendungen bereit und ermöglicht so das Zusammenspiel zwischen den einzelnen Komponenten. Die Middleware ist, wie in Abbildung 3.1 dargestellt, eine logische Schicht zwischen den verteilten Anwendungen und den Betriebssystemen der einzelnen Maschinen.

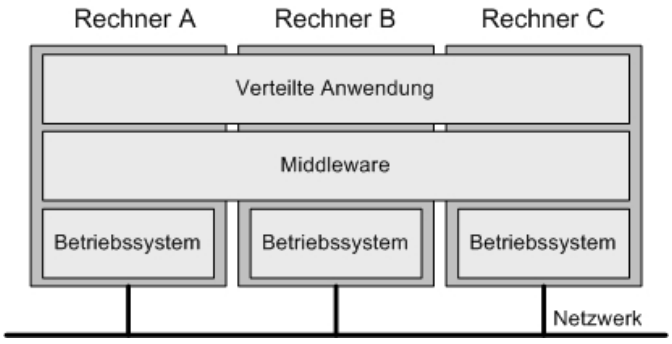

Abbildung 3.1.: Position der Middleware in einem verteilten System [Tanenbaum und van Steen 2002]

Aus Sicht der Anwendung stellt sich die Middleware als eine einheitliche Programmierschnittstelle dar und abstrahiert somit die darunter liegenden Schichten wie Betriebssystem

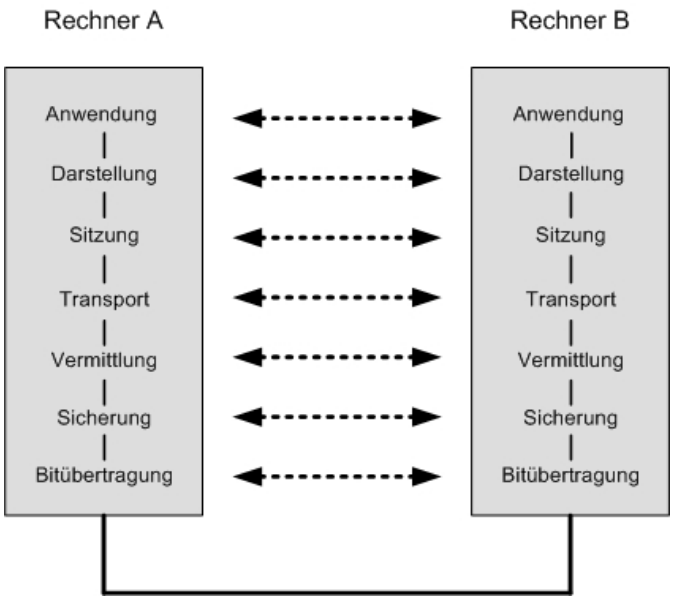

Abbildung 3.2.: ISO OSI-Schichtenmodell

und Kommunikationsprotokoll. Erst durch diese Abstraktion werden die zuvor beschriebenen Eigenschaften eines verteilten Systems ermöglicht.

3.1.2. OSI-Modell

Ein zentrales Problem in verteilten Systemen ist die Kommunikation zwischen den einzelnen beteiligten Maschinen. Da es keinen gemeinsamen Speicher gibt, auf den alle Maschinen zugreifen können, müssen einzelne Nachrichten zwischen den Maschinen ausgetauscht werden. Zum Verständnis, wie Nachrichten in einem Netzwerk ausgetauscht werden, bietet sich das von ISO (International Standards Organisation) entwickelte OSI-Modell an [Zimmermann 1980]. Das OSI-Modell, dargestellt in Abbildung 3.2, beschreibt sieben verschiedene Ebenen, die sich mit jeweils einem bestimmten Aspekt der Netzwerkkommunikation befassen.

Bitübertragungsschicht: Die Aufgabe der Bitübertragungsschicht ist es, die einzelnen Nullen und Einsen zu übertragen. Das dazugehörige Protokoll definiert, wie einzelne Bits übertragen werden. Je nach verwendeter Technologie wird z. B. die Höhe eines Spannungsimpulses oder die Frequenz einer Funkwelle jeweils für das 0-Bit und das 1-Bit definiert.

Sicherungsschicht: Während die Bitübertragungsschicht die einzelnen Bits versendet, ist die Sicherungsschicht zuständig für die korrekte Übertragung von Bits. Mit Hilfe von

Prüfsummen über eine bestimmte Einheit von Bits wird kontrolliert, ob alle Bits korrekt interpretiert wurden.

Netzwerkschicht: Die Netzwerkschicht ist zuständig für die Einteilung in einzelne Datenpakete und die Auslieferung dieser an eine bestimmte Adresse. Switches und Router sind dafür zuständig, dass die Pakete an die richtige Adresse weitergeleitet werden.

Transportschicht: Auf der Ebene der Transportschicht wird eine Nachricht in einzelne Pakete unterteilt. Das jeweils verwendete Transportprotokoll (meist TCP/IP) ist für die korrekte Zerlegung und Übertragung verantwortlich.

Sitzungsschicht: Die Sitzungsschicht ist zuständig für die Synchronisation während der Datenübertragung.

Darstellungsschicht: In der Darstellungsschicht bekommen die übertragenen Daten ihre Bedeutung zurück. Die Bitfolgen werden interpretiert und als Daten, z. B. als alphanumerische Werte, gespeichert.

Anwendungsschicht: Die oberste Ebene ist die Anwendungsschicht, welche die Daten im Sinne des Anwendungsprotokolles interpretiert. Typische Beispiele hierfür sind FTP und HTTP.

Für die Konstruktion einer Middleware sind in diesem Modell vor allem die Darstellungs- und Anwendungsschicht von Bedeutung. Alle darunterliegenden Schichten werden vom Betriebssystem im Zusammenspiel mit der Netzwerkinfrastruktur abstrahiert. Bei Verwendung von TCP/IP ist die korrekte Übermittlung der einzelnen Bits gesichert, somit beschränkt sich die weitere Untersuchung auf die verschiedenen Datenformate, den Inhalt und die Bedeutung des Inhalts der zu übermittelnden Nachrichten.

3.1.3. Konzepte zum Datenaustausch

Ein wesentliches Problem bei dem Austausch von Nachrichten ist das Datenformat, in dem die Daten gesendet werden. So kann z. B. in der Programmiersprache C eine Integervariable je nach verwendetem Compiler zwei, vier oder acht Byte lang sein. Wenn Programme verschiedener Programmiersprachen miteinander kommunizieren, ist das interne Datenformat ein generelles Problem. Es gibt in den später vorgestellten Technologien unterschiedliche Ansätze für den Austausch von Nachrichten, von denen nicht alle für heterogene Umgebungen geeignet sind. Im Folgenden werden drei Ansätze vorgestellt, welche in den Middlewaretechnologien, die in Kapitel 3.2 erläutert werden, Verwendung finden.

3.1.3.1. Binäre Objektserialisierung

Bei der Verwendung von Objektserialisierung, wie es in Java bzw. C# möglich ist, werden ganze Objekte einer Klasse in binärer Form ausgetauscht. Das Problem bei der Objektserialisierung ist das Fehlen von Metainformationen über die Daten. Sowohl der Sender wie der Empfänger müssen die interne Struktur der Klasse kennen, die durch die Daten repräsentiert wird. Dieses setzt automatisch voraus, dass sowohl Sender wie Empfänger in der selben Programmiersprache entwickelt wurden.

3.1.3.2. Interface Definition Language - IDL

Ein weiteres Konzept des Datenaustausches, ist die Verwendung einer IDL (Intermediate Definition Language). Mit Hilfe der IDL wird eine plattform- und sprachunabhängige Darstellung der zu versenden Daten erstellt. Es wird sowohl für den Client, wie auch für den Server die Kommunikationskomponente in der entsprechenden IDL geschrieben, wodurch sichergestellt ist, dass beide Parteien dieselbe Sprache für den Datenaustausch verwenden.

3.1.3.3. Extensible Markup Language - XML

Immer größerer Beliebtheit erfreut sich XML (Extensible Markup Language) als Basis für den Datenaustausch. Mit Hilfe von XML können beliebig komplexe Datenstrukturen innerhalb eines Dokumentes dargestellt werden. Zwei Eigenschaften zeichnen XML als Datenaustauschformat besonders aus: Zu jedem Datum in einem XML-Dokument gehören Metadaten, welche die Semantik dieses Datums beschreiben. Zudem kann einem XML-Dokument ein DTD-Dokument (Document Type Definition) zugrunde gelegt werden, welches die Struktur des XML-Dokumentes beschreibt. Durch die beschreibenden Metadaten sind XML-Dokumente für Menschen sehr gut lesbar, weshalb XML-Dokumente häufig auch in Konfigurationsdateien Verwendung finden. Durch den logischen Aufbau und das strukturbeschreibende DTD-Dokument sind XML-Dokumente auch für Maschinen sehr gut lesbar. Zudem sind für das Einlesen von XML-Dokumenten eine Vielzahl an Parsern für diverse Programmiersprachen verfügbar. Die Darstellung der Daten erfolgt in XML grundsätzlich als Text, unter Angabe des verwendeten Zeichensatzes (z. B. UTF-8).

3.2. RPC-basierte Middlewaretechnologien

3.2.1. Remote Procedure Call - RPC

Der erste weit verbreitete Ansatz zum Aufrufen von Prozeduren auf fremden Rechnern waren Remote Procedure Calls. Der Aufruf einer Prozedur auf einem anderen Rechner sollte dabei für den Client gleich dem Aufruf einer lokalen Prozedur aussehen. Dabei wird der aufrufende Prozess so lange blockiert, bis die entfernte Prozedur abgearbeitet wurde und eine Antwort an den aufrufenden Rechner zurückgesendet worden ist. Bei Prozeduren ohne Rückgabewert wird vom Server nur der Empfang des Aufrufes bestätigt und der Client kann ohne große Verzögerung weiterarbeiten. Um entfernte Prozeduraufrufe transparent erscheinen zu lassen, wird für jede dieser Prozeduren ein Client-Stub entwickelt. Diese lokale Prozedur ruft das jeweils auf dem Server befindliche Gegenstück, den Server-Stub, auf. Der Server-Stub leitet den Aufruf an die entsprechende Prozedur auf dem Server weiter. Remote Procedure Calls werden sehr selten verwendet, da die meisten Programme einen objektorientierten und weniger prozeduralen Ansatz verfolgen. Dennoch dient das RPC-Konzept als Grundlage fast aller aktueller Technologien.

3.2.2. Remote Method Invocation - RMI

Remote Method Invocation ist ein in Java integrierter Ansatz für das Einbinden von Code, der auf anderen Maschinen liegt. Ziel von RMI ist es dabei, entfernte Objekte so transparent zu gestalten, dass kein nennenswerter Unterschied zu lokalen Objekten zu sehen ist. Die Übertragung der Parameterobjekte und das Aufrufen des entfernten Objektes erfolgen dabei über den Client-Stub bzw. über den Server-Skeleton. Für den Datenaustausch nutzt Java-RMI den Java eigenen Persistenzmechanismus, der in 3.1.3 beschrieben wurde, wodurch Java-RMI an die Programmiersprache Java gebunden ist.

3.2.3. Common Object Request Broker Architecture - CORBA

Ein von der Bindung an eine Programmiersprache oder Betriebssystem gänzlich unabhängiger Ansatz wird von CORBA umgesetzt. CORBA gilt als Industriestandard für verteilte objektorientierte Anwendungen für heterogene Umgebungen. Zentraler Bestandteil von CORBA ist der ORB (Object Request Broker), welcher für die Kommunikation in der verteilten Anwendung zuständig ist.

Die Unabhängigkeit von Betriebssystem und Programmiersprache wird dabei durch Einführung der CORBA-IDL erreicht. Die Methodensignaturen verteilter Objekte werden in

CORBA-IDL abgebildet. CORBA selbst ist ein von der OMG verabschiedeter Standard, beschrieben in [OMG 2002]. Die einzelnen CORBA-Implementierungen sind dadurch untereinander kompatibel. Für die einheitliche Konvertierung von Objekten und Strukturen der verwendeten Programmiersprache hin zu CORBA-Objekten wurden Regeln veröffentlicht. Diese Regeln sind u.A. für C, C++, Java, Smalltalk, Ada und Cobol verfügbar.

Neben dem Einsatz der IDL als Austauschformat und den ORBs als Kommunikationsgrundlage setzt CORBA auf einige von CORBA-Objekten angebotene, unabhängige Dienste, die verschiedene Aufgaben in einem CORBA-basiertem System erfüllen. Diese Dienste umfassen z.B. Sicherheitsaspekte, Objektlebenszyklen, Objektpersistenz, Objektabfragen, systemübergreifende Namensgebung und asynchrone Kommunikation. Auf die asynchrone Kommunikation mittels Events und auf die systemübergreifende Namensgebung wird im weiteren Verlauf dieses Kapitels noch gesondert eingegangen.

Sowohl auf Seiten des Clients als auch des Servers gibt es eine Reihe von Komponenten, die in einem CORBA-System zusammenspielen. Abbildung 3.3 zeigt die einzelnen Komponenten.

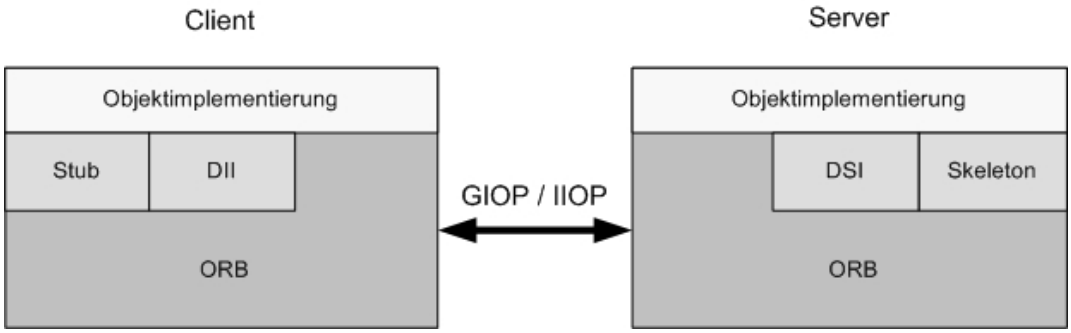

Abbildung 3.3.: Aufbau eines CORBA-Systems

Objekt Request Broker

Der Objekt Request Broker (ORB) ist das Bindeglied bei der Kommunikation verteilter Systeme mit CORBA. Auf jeder Maschine, auf der sich CORBA-Objekte befinden, läuft ein ORB. Der ORB ist für den Datenaustausch zwischen den einzelnen Rechnern zuständig. Bei der Einbindung eines Objektes auf einem fremden Rechner, nimmt der Client-ORB Kontakt zu dem entsprechenden Server-ORB auf und übermittelt die Nachrichten.

Proxy und Skeleton

Proxy und Skeleton sind mit den Stubs aus dem RPC-Konzept vergleichbar. Ihre Aufgabe ist die Konvertierung in das Datenaustauschformat von CORBA. Der IDL-Proxy verpackt den Methodenaufruf samt Parameter und schickt diesen mit Hilfe des ORBs an den Server. Der Skeleton entpackt die Nachricht und sorgt für die Ausführung. Anschließend verpackt

der ORB den Rückgabewert in einer Antwortnachricht und schickt diese zurück an den Client. Dieser Ansatz setzt voraus, dass die IDL-Beschreibung der Objekte bereits bei der Kompilierung des Quellcodes verfügbar ist.

Dynamic Invocation Interface und Dynamic Skeleton Interface

Das Dynamic Invocation Interface (DII) und das Dynamic Skeleton Interface (DSI) ermöglichen die dynamische Einbindung von Schnittstellen. Diese kommen dann zum Einsatz, wenn die IDL-Objektbeschreibung dem Client nicht zur Verfügung steht. Die Schnittstelle dieses Objektes muss vom Server in einem Schnittstellenrepository gespeichert werden. Das Dynamic Invocation Interface kann anhand der Beschreibung aus dem Repository den Aufruf dynamisch erzeugen. Auf der Serverseite befindet sich das Dynamic Skeleton Interface, welches Aufrufe ohne IDL-Beschreibung entgegennimmt.

General Inter-ORB Protokoll und Internet Inter-ORB Protokoll

Für die Kommunikation zwischen den einzelnen ORBs hat die OMG spezielle Protokolle entworfen. Die Basis bildet das General Inter-ORB Protokoll (GIOP). Für die TCP/IP-basierte Kommunikation steht das Internet Inter-ORB Protokoll (IIOP), eine Spezialisierung des GIOP. Auf Grund der sehr hohen Verbreitung von TCP/IP ist IIOP das meist verwendete Kommunikationsprotokoll.

Eventbasierte Kommunikation in CORBA

Alle bisher gezeigten Modelle basierten auf direktem Nachrichtenaustausch zwischen zwei Parteien. Mit dem Event-Dienst und dem Notification-Dienst führt CORBA zwei Dienste ein, die es erlauben, Nachrichten an mehrere Clients zu verteilen anstatt sie einem bestimmten Client zu übermitteln. So kann das Eintreffen eines bestimmten Ereignisses als Nachricht an eine beliebige Anzahl von Clients weitergegeben werden. Dabei lassen sich zwei verschiedene Varianten unterscheiden. Beim Push-Modell gibt der Erzeuger einer Nachricht beim Auftreten eines Ereignisses eine Nachricht an den Ereigniskanal. Die Interessenten einer Nachricht (Verbraucher) registrieren sich bei diesem Ereigniskanal und werden dann bei Auftreten des Ereignisses benachrichtigt. Genau anders herum ist das Vorgehen beim Pull-Modell. Hier registrieren sich die Erzeuger der Ereignisse am Ereigniskanal. Die Verbraucher fragen bei dem Ereigniskanal nach, ob ein Ereignis vorliegt. Der Ereigniskanal selbst fragt darauf hin alle registrierten Erzeuger, ob bei ihnen ein bestimmtes Ereignis eingetreten ist.

Der Eventdienst von CORBA ist nicht in der Lage zwischen unterschiedlichen Events zu unterscheiden, was für eine Filterung von Events nötig wäre. Für jeden Eventtyp muss daher ein eigener Ereigniskanal eingerichtet werden. Unterstützt wird die Filterung von Ereignissen jedoch vom Notificationdienst, der eine Erweiterung des Eventdienstes darstellt.

Laut [Tanenbaum und van Steen 2002] ist die Weitergabe von Ereignissen in CORBA grundsätzlich unzuverlässig. Die CORBA-Spezifikation der OMG trifft keine Aussage über die Garantie, dass ein aufgetretenes Ereignis auch tatsächlich an die Interessenten weitergegeben wird.

Lokationstransparente Aufrufe in CORBA

Mit Hilfe des Naming-Dienstes ist es in CORBA möglich innerhalb eines bestimmten Namenkontextes einzelnen Objekten Namen zuzuweisen. Hierdurch wird die Möglichkeit geschaffen, entfernt auf Objekte zuzugreifen, bei denen nicht bekannt ist, auf welchem Rechner diese sich befinden bzw. durch welchen ORB diese verwaltet werden. Für die Benutzung des Objektes muss lediglich der Name bekannt sein.

3.2.4. Web Services

Web Services sind webbasierte Dienste, die über eine direkte Adresse, einen Unified Resource Identifier (URI), aufgerufen werden. Web Services nutzen das Simple Object Access Protocol (SOAP), ein XML-basiertes und damit plattformübergreifendes Protokoll für die Übertragung von Nachrichten. Der Aufruf eines Web Service sowie die Rückantwort von diesem werden in einer SOAP Nachricht verpackt. Als Middlewareprotokoll von Web Services findet der vom W3C verwaltete Standard SOAP eine laufend zunehmende Verbreitung. Die wesentlichen Eigenschaften von SOAP ergeben sich dabei aus den Eigenschaften der Protokolle mit denen SOAP benutzt wird. Durch XML ist SOAP unabhängig von Plattform und Programmiersprache. Üblich ist es, HTTP[1] Datenübertragungsprotokoll als Träger für SOAP-Nachrichten zu verwenden, dadurch kann auch SSL[2] für die sichere Übertragung der Nachricht verwendet werden. Die Nachricht wird dabei mit dem HTTP-Request versendet, die Antwort wird mit dem HTTP-Response zurückgesandt.

Eine SOAP-Nachricht setzt sich aus drei Komponenten zusammen. Der SOAP-Envelope dient als Behälter für die gesamte SOAP-Nachricht. In dem optionalem SOAP-Header befinden sich Metainformationen zu der Nachricht. Dieses sind vor allem Informationen zu den Themen Sicherheit und Transaktionsmanagement. Im SOAP-Body befindet sich der Funktionsaufruf und die für diesen notwendigen Parameter. Im SOAP-Body wird dabei zwischen Request, Response und Fault unterschieden.

Eine weitere Komponente von Web Services ist Universal Description, Discovery and Integration (UDDI). UDDI ist ein öffentlicher, SOAP-basierter Verzeichnisdienst zur Veröffentlichtlichung von Web Services. Für die Beschreibung eines Web Service wird dabei die Web

[1] HTTP (Hypertext Transfer Protocol) ist ein Protokoll zur Datenübertragung im Netzwerk. HTTP wird hauptsächlich zur Übertragung von Webseiten verwendet.

[2] SSL (Secure Socket Layer) ist ein Verschlüsselungsprotokoll zur sicheren Datenübertragung im Internet.

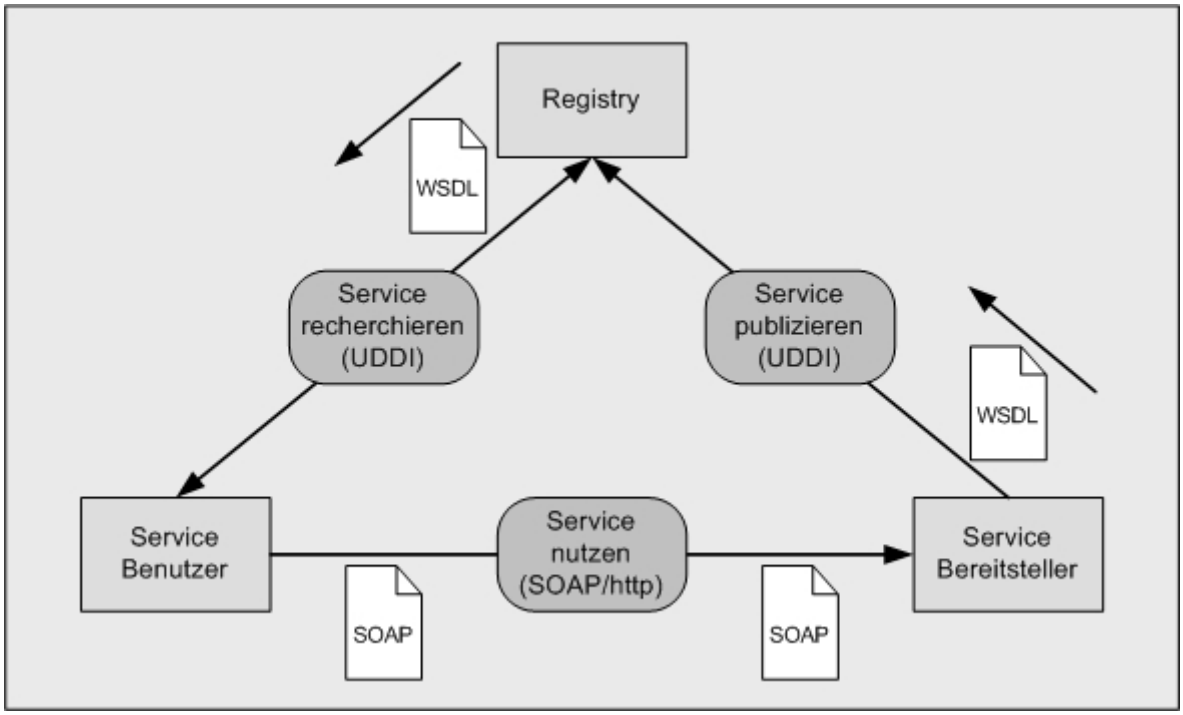

Abbildung 3.4.: Web Services

Service Description Language (WSDL) verwendet. WSDL ist eine XML-basierte Beschreibung des Web Service. Diese enthält die URI sowie Informationen über Methodennamen, Parameter und Rückgabewerte. Hinzu kommen Metainformationen wie Beschreibung und Anbieter des Web Service.

Mit Hilfe der WSDL-Datei kann der für den Aufruf des Web Service notwendige Code automatisch generiert werden. Die Integration eines Web Service kann hierdurch weitestgehend automatisiert werden. Das Zusammenspiel der Komponenten ist in Abbildung 3.4 dargestellt. Für weiterführende Erläuterungen zu Web Services sei auf [Mörike 2003] verwiesen.

3.2.5. Fazit

Es wurden verschiedene RPC-basierte Middlewaretechnologien vorgestellt. In erster Linie sind diese vor allem mit einem bestimmten Programmmierparadigma, dem klassischen Client-Server-Computing, verbunden. Die Schnittstellen des Empfängers müssen im Vorwege publiziert werden und der Methodenaufruf wird grundsätzlich über eine Punkt zu Punkt Kommunikation umgesetzt.

Java-RMI ist eine Technologie die zwar in jeder Java-Laufzeitumgebung verfügbar ist, jedoch auf Java limitiert ist. Mit Web Services und CORBA sind zwei Standards definiert worden, die große Akzeptanz in der Wirtschaft gefunden haben. Bei Verwendung einer dieser Technologien muss noch eine Produktevaluation durchgeführt werden. Beide Standards sind sehr umfangreich, wodurch ein hoher Entwicklungsaufwand bei der Realisierung zu vermuten ist. Im Rahmen der Produktevaluation muss auch die Umsetzbarkeit auf ressourcenarmen Geräten geprüft werden. CORBA bildet eine Ausnahme des zuvor erwähnten Programmierparadigmas. Die zusätzlichen Event- und Naming-Dienste bieten grundsätzlich die Möglichkeit der entkoppelten Kommunikation.

3.3. Blackboardbasierte Middlewaretechnologien

Mit Ausnahme des Event-Dienstes von CORBA basieren alle bisher beschriebenen Modelle zur Interprozesskommunikation auf synchroner Punkt zu Punkt Kommunikation. Voraussetzung für die synchrone Kommunikation ist die Verfügbarkeit von Client und Server während der gesamten Dauer der Kommunikation. Des weiteren muss der Client wissen, wie der Server anzusprechen ist, z. B. über eine Netzwerkadresse oder einem Computernamen. Der Naming-Dienst von CORBA ist hier wiederum die Ausnahme.

Blackboardbasierte Kommunikation ist ein vollständig entkoppelter Ansatz, der weder die Voraussetzung hat, dass ein bestimmter Empfänger zum Zeitpunkt der Übermittlung der Nachricht verfügbar ist, noch dass dieser überhaupt bekannt ist. Das Blackboard dient dabei als Nachrichtenspeicher, an den Nachrichten verschickt werden und an dem sich an dieser Nachricht interessierte eben diese Nachricht abholen. Durch den Polling-Mechanismus kann jeder Client selber entscheiden, wann er eine Nachricht erhalten möchte und welche Nachrichten er erhalten möchte. Hierdurch kann auch der kurzweilige Ausfall eines Rechners, z. B. durch Netzwerkverbindungsprobleme oder auf Grund zu geringer Rechenleistung überbrückt werden. Einzelne Clients haben die Möglichkeit, Nachrichten zu filtern, d.h. sie erhalten nur die Nachrichten, die für sie von Interesse sind. Mit Hilfe des Blackboards lässt sich auch die vielfache Verteilung einer Nachricht ermöglichen. So muss nur eine einzige Nachricht versendet werden, um eine unbestimmte Anzahl an Clients zu erreichen.

Die Umsetzung von Client-Server-Kommunikation ist bei Verwendung eines Blackboards gleich der Client-Client-Kommunikation. Der Server, der eine Nachricht verarbeitet, nutzt das Blackboard in gleicher Weise wie Clients, indem auch dieser sich die Nachrichten abholt, für die er sich interessiert.

Die Idee von blackboardbasierter Kommunikation stammt aus den siebziger Jahren, wo erste blackboardbasierte Systeme Einsatz in Anwendungen der künstlichen Intelligenz fanden. David Gelernter entwickelte Anfang der achtziger Jahre an der Yale University die verteilte

Programmiersprache Linda und definierte damit ein grundlegendes Modell zur Interprozesskommunikation in verteilten Systemen mit Hilfe von Tupelräumen[3]. Moderne Implementierungen von Tupelräumen basieren alle auf den von Gelernter definierten Prinzipien. Für die Arbeit mit Tupelräumen wurden dabei drei elementare Operationen definiert (vgl. [Gelernter 1985]):

- `read` gibt ein Tupel zurück, welches auf ein gegebenes Suchmuster zutrifft.
- `take` gibt ebenfalls ein dem Suchmuster entsprechendes Tupel zurück. Das Tupel wird dabei aus dem Tupelraum entfernt.
- `write` speichert ein Tupel im Tupelraum.

Für die lesenden Zugriffe mittels `read` und `take` muss ein entsprechendes Tupel als Vorlage definiert werden. Bei Ausführung der lesenden Operation werden ein oder mehrere Tupel zurückgeliefert, die dem Vorlagetupel entsprechen.

Basierend auf der Idee von Gelernter wurden eine Reihe Implementierungen von Tupelräumen durchgeführt. Bei allen Implementierungen sind die Basisoperationen `read`, `take` und `write` wieder zu finden. Moderne Implementierungen haben jedoch einen größeren Umfang an Funktionalität als Linda. Neben den nachfolgend erwähnten Implementierungen wurden weitere Linda-basierte Systeme in [Wells u. a. 2004] verglichen.

3.3.1. JavaSpaces

JavaSpaces ist eine Referenzimplementierung der gleichnamigen Architektur von SUN. JavaSpaces ist Teil der JINI Technologie [Jini], welches eine serviceorientierte Architektur für Anwendungen in heterogenen, dynamischen Netzwerken darstellt.

JavaSpaces ist eine objektorientierte Implementierung von Tupelräumen, die anstatt spezieller Tupel beliebige Objekte, die ein Interface Entry implementieren, benutzt. JavaSpaces erweitert das von Gelernter vorgeschlagene Modell vor allem durch Definition der Lebenszeit von Tupeln im Tupelraum und durch Einführung von Operationen um mehrere Aktionen in einem transaktionalen Kontext unter Einhaltung der ACID-Eigenschaften durchzuführen. JavaSpaces ist eine rein Java-basierte Umsetzung von Tupelräumen. Neben der Referenzimplementierung von SUN, gibt es zu JavaSpaces auch eine kommerzielle Implementierung namens GigaSpaces [GigaSpaces]. Diese unterstützt auch Clients, die nicht in Java geschrieben wurden, indem auch SOAP als Transportmittel für Nachrichten eingesetzt werden kann.

[3] Für den Begriff Tupelräume wird in der deutschen Literatur vielfach der englische Begriff Tuplespace verwendet.

3.3.2. TSpaces

TSpaces [TSpaces] ist eine von IBM entworfene Technologie zur Umsetzung von Tupelräumen in Java. TSpaces setzen sich vor allem durch eine direkte Datenbankanbindung von anderen Implementierungen ab. Dadurch ergeben sich vor allem neue Möglichkeiten bei der Abfrage von Tupel. Der Umfang der Funktionalität von TSpaces gleicht dem von JavaSpaces. TSpaces bietet im Vergleich zu JavaSpaces eine Schnittstelle für das Hinzufügen weiterer Funktionalität. Eine sehr nützliche Erweiterung von TSpaces ist die Integration von XML-Dokumenten als Übermittlung von Tupel, wie in [Tolksdorf und Glaubitz 2001] beschrieben.

3.3.3. Event Heap

Der Event Heap ist eine im Rahmen des Interactive Workspaces Projekt (siehe auch 3.4.1) der Stanford University entwickelte tupelraumbasierte Technologie, die speziell auf die Bedürfnisse des Interactive Workspaces Projekts angepasst wurde. Der Event Heap ist Teil der Software iROS, die sich als Metabetriebssystem für das Interactive Workspaces Projekt versteht. In [Johanson und Fox 2004] berichten Johanson und Fox über die Limitierungen von gängigen Tupelraumimplementierungen und stellen Anforderungen zusammen, um mit Hilfe von Tupelräumen die Koordination von verteilten Anwendungen in Gruppenarbeitsräumen durchführen zu können. Eine der wichtigsten Erweiterungen gegenüber dem Linda Tupelraumkonzept sind Metainformationen zu den gegebenen Tupeln. Tupel im Event Heap sind selbstbeschreibend, wodurch ein Reverse Engineering anhand der einzelnen Felder eines Tupels ermöglicht wird. Der Event Heap, ausführlich in der Dissertation von Johanson [Johanson 2002] beschrieben, führt eine Reihe von Standardfeldern ein, die in jedem Tupel verfügbar sind. Zu diesen Feldern gehören Informationen über den Ersteller des Tupels z. B. Anwendungsname, Computername, SessionID und Weitere. Diese Felder werden vom Event Heap gesetzt und sind somit in jedem Tupel verfügbar.

Im Gegensatz zu vielen anderen Tupelraumimplementierungen bietet der Event Heap Unterstützung für eine Reihe von Plattformen und Programmiersprachen für Clients an. Der Event Heap selbst ist in Java geschrieben und läuft somit auf Betriebsystemen wie Windows, Linux und MAC OS. Neben Java ist auch für C/C++ und Python eine Client-API verfügbar. Der Event Heap nutzt dabei ein proprietäres Format, welches jedoch in [Johanson und Salgar 2002] ausführlich dokumentiert ist. Neben der direkten Socketverbindung zu dem Event Heap verfügt dieser über die Möglichkeit, Tupel in HTTP-Requests zu verpacken. Hierdurch lässt sich jedes Gerät welches über einen Internetbrowser verfügt als Client verwenden.

Der Event Heap wurde an der Stanford University ohne weitere Sicherheitsmechanismen implementiert. Hierzu wurde zusammen mit dem Swedish Royal Institute of Technology im Rahmen des Projekts iSecurity der Event Heap um Sicherheitsmechanismen wie verschlüsselter Datenübermittlung und Authentifikation am Tupelraum erweitert [Song u. a. 2003].

3.4. Verwandte Projekte

In einer Reihe von Forschungsprojekten wurden bereits die Themen computergestützte Gruppenarbeitsräume und ubiquitäre Umgebungen behandelt. Mit dem Interactive Workspaces Projekt und dem i-Land Projekt werden an dieser Stelle zwei sehr bedeutende Projekte kurz vorgestellt.

3.4.1. Interactive Workspaces

Das Projekt Interactive Workspaces der Stanford University beschäftigt sich mit neuen Interaktionsformen zwischen Menschen und Computer im Rahmen ubiquitärer Umgebungen. Hierzu wurde der Gruppenarbeitsraum iRoom entwickelt, der vor allem auf die Benutzung großer, hochauflösender Bildschirme setzt. Von denen sind vier an der Wand montiert und zusätzlich wurde einer in einen Konferenztisch eingelassen. Der Aufbau des Gruppenarbeitsraumes ist in Abbildung 3.5 dargestellt. Besonderes Ziel dieses Projektes ist es, die Bedienung des Konferenzraumes so einfach und intuitiv wie möglich zu gestalten [Johanson u. a. 2002a].

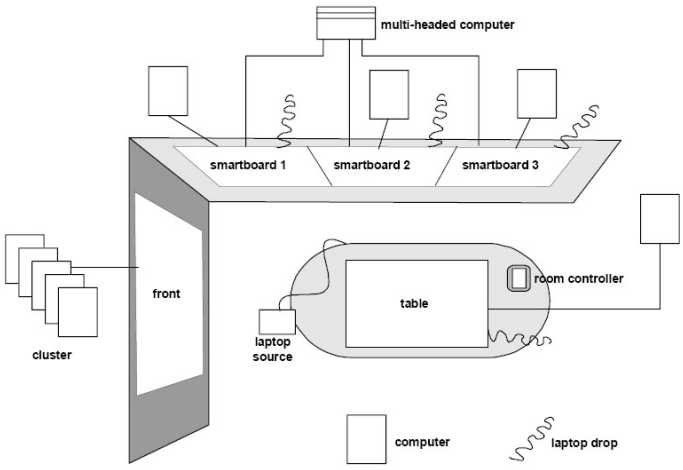

Abbildung 3.5.: Aufbau des iRooms [Johanson und Fox 2002]

Für den Betrieb des Arbeitsraumes wurde hierzu das Metabetriebssystem iROS entwickelt. iROS hat das Ziel, die vorhandenen Geräte und die durch die Benutzer mitgebrachten Geräte miteinander zu vernetzen und von jedem Gerät aus die Steuerung des Raumes zu ermöglichen. Zur Koordination der Anwendungen setzt iROS auf den zuvor erläuterten Event Heap. Für den Betrieb des Arbeitsraumes wurden eigens eine Reihe von Anwendungen entwickelt, die speziell auf den Event Heap aufsetzen. Für die Verteilung von Dokumenten verwendet iROS den Data Heap. Der Data Heap ist eine Komponente die, basierend auf einem WebDAV-Server Dateien für alle beteiligten Geräten im Raum zugänglich macht. Dateien auf dem Data Heap werden mit Hilfe verschiedener beschreibender Attribute anstatt von Verzeichnispfaden gespeichert. Für den Zugriff auf die Dateien ist somit keine Kenntnis des Dateisystems notwendig. Die dritte Schlüsselkomponente von iROS ist der iCrafter, welcher für das dynamische Veröffentlichen und Einbinden einzelner Dienste zuständig ist.

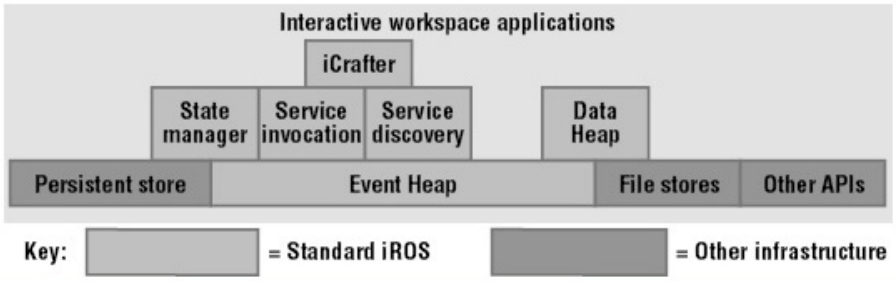

Abbildung 3.6.: Komponenten in iROS [Johanson u. a. 2002a]

3.4.2. i-Land

Das Projekt i-Land am Fraunhofer-Institut in Darmstadt befasst sich ebenfalls mit den Themen Human-Computer-Interaction (HCI) und Computer-Supported Coorparative Work (CSCW). Im Rahmen des Unterprojektes Roomware wurden hierzu technische Geräte direkt in das Mobiliar eines Konferenzraumes eingelassen. Durch diese enge Bindung zu Computern sollen kreative Arbeitsprozesse noch stärker als zuvor möglich durch technische Hilfsmittel unterstützt werden. Die direkte Vernetzung der vorhandenen Gegenstände soll dabei die Arbeit auf einer gemeinsamen Datenbasis förden, was zu einer inhaltlichen Verbesserung der Arbeit führen soll [Tandler u. a. 2002], [Streitz u. a. 1999].

Die technologische Grundlage des i-Land Projektes bildet die eigens für dieses Projekt entwickelte Software BEACH [Tandler 2003]. BEACH basiert auf dem COAST-Framework [Schümmer u. a. 2000], welches eine Infrastruktur für verteilte, synchrone Bearbeitung von Objekten darstellt. COAST wurde dabei speziell für Einsatz in Groupware entwickelt.

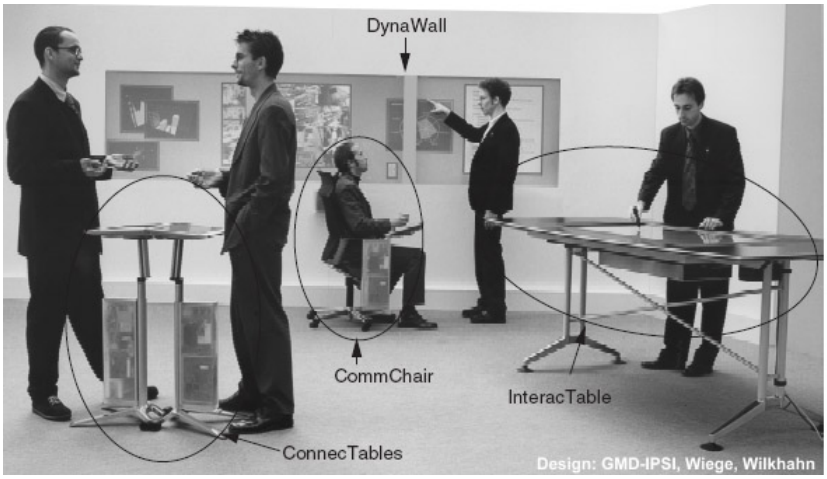

Abbildung 3.7.: Roomware - Second Generation [Tandler u. a. 2002]

3.4.3. Fazit

Mit Interactive Workspaces und i-Land wurden zwei Projekte vorgestellt, die wie das Collaborative Workplace das Ziel verfolgen, Gruppenarbeit effizienter zu gestalten. Der Schwerpunkt des i-Land Projektes liegt bei der Konstruktion einer festen Infrastruktur und der synchronen Bearbeitung verteilter Objekte. Die zu diesem Zweck erstellte Software BEACH bzw. COAST bietet hierfür das Fundament. Das Interactive Workspaces Projekt setzt dagegen auf die Integration mobiler Endgeräte. Die Software iROS ist darauf ausgelegt, die Kontrolle der Bildschirme und das Anzeigen von Daten auf vielen verschiedenen Endgeräten zu ermöglichen. Der Hauptunterschied zwischen dem Collaborative Workplace und Interactive Workspaces besteht in der benutzerzentrierte Ausrichtung des Collaborative Workplace. Im Collaborative Workplace bedarf die Teilnahme am Konferenzraum einer Authentifizierung. Zusätzlich sind Dokumente Benutzern zugeordnet und können von anderen Personen, sofern diese kein Zugriffsrecht auf dieses Besitzen, nicht gesehen werden. Dieses setzt jedoch ein grundlegendes Sicherheitsmodell für eine kollaborative Umgebung voraus.

4. Design und Realisierung

4.1. Systemkonfiguration

Die Hardwareausstattung des Konferenzraumes lässt sich in vier Bereiche einteilen: Die mobilen Endgeräte, die statischen Endgeräte (öffentliche Bildschirme), die Server und die Netzwerkinfrastruktur. Abbildung 4.1 zeigt, wie die einzelnen Komponenten miteinander verbunden sind.

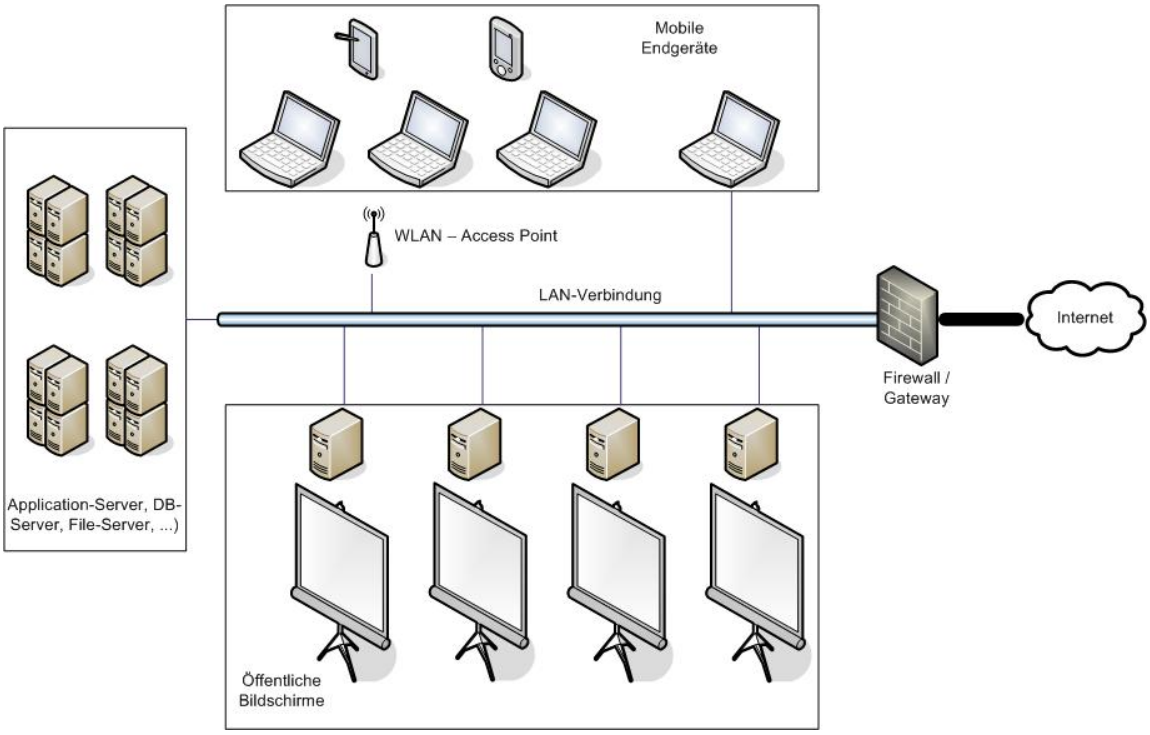

Abbildung 4.1.: Technische Infrastruktur des Konferenzraumes

4.1.1. Öffentliche Bildschirme

Kern des Konferenzraumes sind die öffentlichen Bildschirme. Mehrere dieser Bildschirme werden nebeneinander an der Wand platziert. Aufgrund ihrer Größe von etwa 42” Bildschirmdiagonale sind diese für jede Person im Raum auch aus mehreren Metern Entfernung gut sichtbar. Zusätzlich befindet sich auf der Oberfläche dieser Bildschirme eine berührungssensitive Folie, wodurch diese Bildschirme als Touchscreen fungieren. Das Arbeiten an diesen Bildschirmen gestaltet sich ähnlich wie vor einem Whiteboard. Mehrere Personen können gleichzeitig davor stehen und mit Hilfe eines Eingabestiftes bzw. durch Berührung mit dem Finger das Gerät bedienen.

Zusätzlich zu den an der Wand montierten Bildschirmen können weitere öffentliche Bildschirme in einzelne Konferenztische eingelassen werden. Diese Bildschirme sind mit etwa 30” etwas kleiner als die Wandbildschirme, sind aber ebenfalls berührungssensitiv.

Die öffentlichen Bildschirme können als statisch bezeichnet werden. Die Konfiguration im Raum kann zwar durch Hinzunahme oder Wegfall einzelner Bildschirme variieren, dennoch wird es hier keine besondere Fluktuation geben.

Jeder dieser Bildschirme wird dabei an einem separaten Computer angeschlossen. Dadurch sind die Bildschirme unabhängig voneinander und es kann an allen Bildschirmen gleichzeitig gearbeitet werden.

4.1.2. Mobile Endgeräte

Die Teilnehmer einer Besprechung haben die Möglichkeit, eigene mobile Endgeräte mit in den Raum zu bringen. Hierzu zählen Laptops, Handhelds, PDAs und Mobiltelefone. Auf diesen Geräten muss eine Clientanwendung laufen, über welche die Benutzer Zugang zu den öffentlichen und den eigenen privaten Dokumenten bekommen. Die Steuerung der öffentlichen Bildschirme, z. B. das Anzeigen eines Dokumentes auf einem bestimmten Bildschirm, erfolgt ebenfalls über die Clientanwendung.

Anders als bei den öffentlichen Bildschirmen, werden die mobilen Endgeräte dynamisch dem System hinzugefügt bzw. aus dem System entfernt. Hinzu kommt, dass einzelne Geräte nicht immer verfügbar sind, sondern zwischendurch ausgeschaltet bzw. in den Ruhezustand versetzt werden. Zu den mobilen Endgeräten gehören neben Laptops auch ressourcenarmen Geräte wie PDAs und Handhelds.

4.1.3. Server

Eine Reihe von Servern stellen allgemeine Dienste für die Clientanwendungen bereit. Die Basis des Systems bildet eine zentrale Serveranwendung, an der sich die einzelnen Benutzer authentifizieren müssen, um in dem Raum zu arbeiten. Diese Anwendung stellt verschiedene Dienste bereit, um auf zentral gespeicherte Daten zuzugreifen. Die einzelnen Dienste bedienen sich einer Datenbank, in der benutzerspezifische Daten sowie das Rechtemodell für den Zugriff auf diese Daten abgebildet sind. Die zentrale Speicherung von Dokumenten wird ebenfalls über einen Dienst der Serveranwendung realisiert. Die Dokumente können entweder in einem Dateisystem oder direkt in der Datenbank gespeichert werden, wie es in [Mund 2006] vorgeschlagen wird.

Die hier identifizierten logischen Elemente zentrale Anwendung, Datenbank und Dokumentenspeicherung lassen sich alle mit Hilfe eines einzigen Servers realisieren. Durch die logische Trennung der Elemente entsteht hier aber eine sehr gute Skalierbarkeit. Bei Performanceengpässen können einzelne Aufgaben ohne bedeutenden Aufwand auf verschiedene Server aufgeteilt werden.

4.1.4. Netzwerkinfrastruktur

Zur Integration in den Konferenzraum, müssen alle Geräte über eine Netzwerkanbindung verfügen. Die einzelnen Geräte werden entweder über LAN oder WLAN an das Netzwerk angebunden. Über dieses Netzwerk wird auch der Zugriff auf ein Internetgateway bereitgestellt. Bei der Verwendung von WLAN ist generell anzumerken, dass zusätzliche Kontrollen notwendig sind, um den Zugriff auf das Netzwerk von außerhalb des Raumes zu unterbinden.

Die Netzwerkanbindung der öffentlichen Endgeräte sowie der Server erfolgt vorzugsweise über eine LAN-Verbindung. Die Anbindung über WLAN ist ebenfalls denkbar und könnte zum Einsatz kommen, wenn die Netzwerkkabel ein großes Hindernis beim Aufstellen der Bildschirme darstellen. Bei einer mobilen Konstruktion kann dieses in Betracht gezogen werden. Die dadurch entstehende geringere Datenübertragung, ist nur bei der Ansicht sehr großer Dokumente (Dokumente größer als 10MB) für den Benutzer deutlich zu bemerken.

Die mobilen Endgeräte werden üblicherweise über WLAN an das Netzwerk angebunden. Bei einer Vielzahl an Geräten würden Netzwerkkabel sehr störend im Raum sein. Geräte ohne LAN-Anschluss, z. B. Handhelds, müssen sogar über WLAN verbunden werden. Durch den Im- und Export ist der Nachteil der geringeren Datenübertragung bei den Geräten der Besprechungsteilnehmer deutlicher als bei den öffentlichen Bildschirmen. Werden bei dem

Im- und Export viele Dokumente übertragen, führt die geringere Datenübertragungsrate zu einer deutlich spürbaren Verzögerung.

4.2. Interprozesskommunikation im Konferenzraum

Dieser Abschnitt behandelt die Kommunikation zwischen den verteilten Anwendungen im Konferenzraum. Ziel dieses Abschnittes ist es, ein Konzept zu entwickeln, wie die einzelnen Geräte miteinander kommunizieren und Nachrichten untereinander austauschen.

4.2.1. Identifikation der Nachrichten

Die in Kapitel 4.1 identifizierten Elemente Server, mobile Endgeräte und öffentliche Bildschirme, werden hier auf die auszutauschenden Nachrichten untersucht. Dabei soll herausgestellt werden, welche Nachrichten ausgetauscht werden und wer die beiden Kommunikationspartner sind.

Ein wesentlicher Teil der auszutauschenden Nachrichten fällt durch die Kommunikation zwischen den Clients und einem Server, der verschiedene Systemdienste bereitstellt, an. Hierzu gehören u. a. die An- und Abmeldung am System, der Im- und Export von Dokumenten und der Austausch der entsprechenden Dokumentmetadaten.

Für die direkte Interaktion zwischen zwei Endgeräten werden Nachrichten zwischen diesen beiden Parteien ausgetauscht. Der am häufigsten vorkommende Anwendungsfall ist das Anzeigen bzw. Bearbeiten eines Dokumentes. Diese Nachrichten werden i. A. von öffentlichen Bildschirmen empfangen, dennoch können auch mobile Endgeräte Empfänger dieser Nachrichten sein, z. B. zum Anzeigen von Dokumenten oder Internetseiten.

Eine weitere Form der Interaktion entsteht bei verteilten Anwendungen, die für diesen Konferenzraum entwickelt werden. Hierbei könnten Anwendungen auf verschiedenen Geräten gleichzeitig laufen, aber generell verschiedene Rollen einnehmen bzw. verschiedene Ansichten auf die selben Daten zeigen. Dieses sind Anwendungen, die entweder auf einer gemeinsamen Datenbasis in Form einer Datenbank arbeiten, oder die ihre Daten untereinander austauschen. Diese Anwendungen müssen grundsätzlich einen gemeinsamen Mechanismus zur Koordinierung und Synchronisierung umsetzen. Bei diesen Applikationen sind die genauen Empfänger einer Nachricht nicht unbedingt zu identifizieren. Änderungen an Daten, ob bei verteilten oder zentral gehaltenen Daten, sind für eine unbestimmte Menge von Clients von Bedeutung.

Bei Nachrichten mit allgemeinen Zustandsinformationen eines Clients sind die Empfänger ebenfalls schwer zu identifizieren. Diese Nachrichten könnten grundsätzlich jeden Client

interessieren, bei ressourcenarmen Geräten kann man argumentieren, dass diese Geräte jene Nachrichten nicht weiter verarbeiten wollen und deshalb nicht an ihnen interessiert sind.

Von der zentralen Serveranwendung können ebenfalls Nachrichten versendet werden. Der Server sollte z. B. Informationen über geänderte Dokumentmetadaten an Clients verschicken, damit diese ihre Daten entsprechend aktualisieren können.

4.2.2. Klassifizierung der Nachrichten

Die im vorigen Abschnitt identifizierten Nachrichten lassen sich in drei verschiedene Nachrichtentypen einteilen:

- **Systemnachrichten** sind Nachrichten, die zwischen Clients und dem Server ausgetauscht werden. Diese Nachrichten werden genutzt um, die vom Server angebotenen Dienste in Anspruch zu nehmen. Systemnachrichten werden nur zwischen Clients und Server ausgetauscht und haben somit immer einen eindeutigen Empfänger.
- **Interaktionsnachrichten** sind Nachrichten, die einen oder mehrere Clients auffordern, eine bestimmte Aktion, wie das Anzeigen eines Dokumentes, durchzuführen. Informationsnachrichten haben einen oder mehrere Empfänger. Die Empfänger sind bekannt und können explizit angegeben werden.
- **Informationsnachrichten** sind Nachrichten, die unkritische Statusinformationen beinhalten. Informationsnachrichten werden von Clients als auch vom Server verschickt und können sowohl definierte, wie undefinierte Empfänger haben. Nachrichten, die keinen definierten Empfänger besitzen, können also als Broadcastnachrichten bezeichnet werden.

Die Tabelle 4.1 gibt einen groben Überblick über die in jedem Fall auszutauschenden Nachrichten im Konferenzraum. Es werden Sender, Empfänger und der Inhalt der Nachricht herausgestellt. Zusätzlich wird jede Nachricht gemäß obiger Klassifizierung einem bestimmten Typ zugeordnet.

4.2.3. Konsequenzen für die Middleware

Aus den im vorherigen Abschnitt vorgestellten Nachrichten wird ersichtlich, dass sich für die Middleware Anforderungen aus verschiedenen Bereichen der Interprozesskommunikation ableiten lassen. Ein wesentlicher Teil der ausgetauschten Nachrichten ist als direkte Client-Server-Kommunikation zu betrachten. Ein Client, in diesem Fall entweder ein von einem Konferenzteilnehmer mit in den Raum gebrachter Laptop oder einer der öffentlichen Bildschirme, ruft einen serverseitig angebotenen Dienst auf und bekommt

Sender	Empfänger	Nachricht	Typ
Mob. Endgerät	Server	Login	System
Mob. Endgerät	Server	Import / Export	System
Mob. Endgerät	Server	Anfordern von Dokument Metadaten	System
Mob. Endgerät	Server	Ändern von Dokument Metadaten	System
Mob. Endgerät	Öffentl. Display	Dokument Anzeigen	Interaktion
Mob. Endgerät	Alle Endgeräte	Eigener Status	Info
Mob. Endgerät	Interessierte Geräte	Nachricht verteilter Anwendung	Interaktion
Öffentl. Display	Server	Ändern von Dokument Metadaten	System
Öffentl. Display	Server	Empfangen von Dokumenten	System
Öffentl. Display	Server	Schreiben von Dokumenten	System
Öffentl. Display	Öffentl. Display	Dokument Anzeigen	Interaktion
Öffentl. Display	Alle Endgeräte	Eigener Status	Info
Öffentl. Display	Interessierte Geräte	Nachricht verteilter Anwendung	Interaktion
Server	Interessierte Geräte	Aktualisierte Metadaten	Info

Tabelle 4.1.: Auswahl einiger Nachrichten im Konferenzraumsystem

von dem Server eine entsprechende Rückantwort. Die Kommunikation folgt dem in Kapitel 3.2.1 vorgestellten RPC-Muster.

Daneben erfolgt auch Kommunikation zwischen den Clients untereinander, so dass ein nachrichtenempfangender Client hier die Rolle des Servers einnimmt. Der als Server agierende Client muss also ebenfalls Dienste bereitstellen. Von dem klassischen RPC unterscheidet sich diese Kommunikation dadurch, dass diese Nachricht auch an verschiedene Empfänger gleichzeitig verschickt werden kann.

Neben der direkten Kommunikation, bei welcher der Client vorher weiß, an wen diese Nachricht verschickt wird, und dass dieser Empfänger auch den beanspruchten Dienst anbietet, erfolgt auch Kommunikation mit Clients, von denen der sendende Client nicht weiß, wer Interesse am Erhalt dieser Nachricht hat. Für die Middleware bedeutet dieses, dass sie einen Mechanismus bereitstellen muss, um den Empfang einer Nachricht von seiten des empfangenden Clients zu steuern. Diese Nachrichten sind nicht als genutzte Dienste zu verstehen, sondern dienen als allgemeine Information für den Client. Das Vorkommen dieser Nachrichten wurde bereits bei der Analyse der Nachrichten erläutert, der ersten Entwicklungsphase des Konferenzraumes sind jedoch nur wenige Anwendungsfälle geplant, bei denen nur bestimmte Clients an Informationsnachrichten interessiert sind. Es sei jedoch erwähnt, dass in der weiteren Entwicklung des Konferenzraumes Anwendungen entworfen bzw. integriert werden sollen, die auf genau diese Art und Weise miteinander interagieren sollen. Die Middleware muss diesen Fall also unbedingt berücksichtigen.

Wie in dem Kapitel 4.1 erläutert, kann bei den mobilen Endgeräten keine verlässliche Aussage über die Verfügbarkeit der Geräte gemacht werden. Auch wenn Clients die Möglichkeit

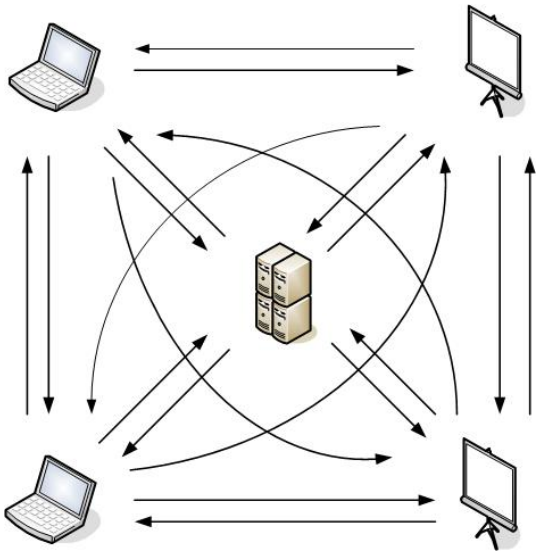

Abbildung 4.2.: Logische Sicht des Nachrichtenaustausches

haben, sich vom System abzumelden, sollte der Fall, dass ein Kommunikationspartner nicht auf eine Nachricht reagiert, als normal betrachtet werden. Das System darf hierdurch nicht beeinträchtigt werden. Die Middleware sollte in jedem Fall sicherstellen, dass Informationen über verfügbare Clients aktuell sind. Um eine möglichst große Breite mobiler Endgeräte in dem Konferenzraum zu unterstützen, darf durch die Middleware keine Einschränkung auf eine bestimmte Systemplattform entstehen.

Aus den genannten Konsequenzen zeigt sich, dass die Anforderungen an den Konferenzraum aus zwei gegensätzlichen Welten stammen: Zum einen gibt es Anforderungen aus der Welt des klassischen Client-Server Computings, zum anderen gibt es verschiedene Anforderungen, die aus dem Bereich des Ubiquitous Computings bekannt sind.

4.2.4. Problematik des Dokumentenaustausches

Ein generelles Problem für die Middleware ist der Dokumentenaustausch zwischen den Clients und dem Server. Es gibt drei grundlegende Anforderungen, welche die Middleware erfüllen soll:

1. Die Middleware muss sicherstellen, dass die Benutzer nur Zugriff auf die Dokumente haben, auf die diese zumindest ein Leserecht besitzen. Für schreibenden Zugriff ist zusätzlich das Schreibrecht notwendig.

2. Die Middleware soll sicherstellen, dass elementare Dokumentmetadaten zu den Dokumenten gespeichert werden.

3. Die Middleware muss die Dokumente auch externen Programme, für die lediglich ein Adapter zur Steuerung vorliegt, zur Verfügung stellen.

Eine sehr nahe liegende Lösung ist die Nutzung SMB-Dienstes[1] für den Zugriff auf die Dokumente. Hiermit lässt sich besonders die Integration externer Programme sehr erleichtern. Die Programme nehmen direkten Lese- und Schreibzugriff auf die auf dem Server liegenden Dateien. Die Umsetzung der Zugriffsrechte wird direkt über das darunterliegende Dateisystem gesteuert. Die Umsetzung eines rollenbasierten Rechtemodells wird damit nicht möglich sein. Die Versionierung von Dokumenten schließt diese Lösung ebenfalls aus. Beim Speichern eines Dokumentes wird dieses auf dem Server überschrieben.

Eine weitere Schwierigkeit liegt in der Sichtbarkeit von Dokumenten, die nicht in dem Verzeichnis des Benutzers liegen, sondern Dokumente sind, für die andere Benutzer diesem Benutzer ein explizites Lese- oder Schreibrecht erteilt haben. Diese Dokumente werden nicht direkt angezeigt, obwohl der Benutzer ein Leserecht auf dieses Dokument besitzt. Bei der Nutzung des SMB-Dienstes müssten somit sämtliche Dateien in einem einzigen Verzeichnis liegen. Dieses hat zur Folge, dass die Dateinamen auch benutzerübergreifend eindeutig sein müssen.

Als weitere Schwierigkeit erweist sich die Pflege der Metadaten. Auf Ebene des Dateisystems lässt sich lediglich der Eigentümer des Dokumentes feststellen. Für ausführliche Metadaten bleibt die Anbindung an eine Datenbank notwendig. Diese kann lediglich einen Verweis in Form eines genauen Pfades zu der Datei halten. Das Umbenennen eines Dokumentes, welches nicht über eine Konferenzraumanwendung durchgeführt wird hätte zur Folge, dass die Metadaten anschließend nicht mehr der Datei zuzuordnen sind.

Dieses Problem wirft die Notwendigkeit auf, Dateien vor dem direkten Zugriff zu schützen und den Zugriff nur mit Hilfe eines Dienstes der vorliegenden Middleware zu ermöglichen. Dadurch ergeben sich jedoch Probleme für externe Programme, wenn diese auf Dateien zugreifen müssen. Es kann lediglich auf einer lokalen Kopie der Datei gearbeitet werden, welche anschließend wieder in den zentralen Dokumentspeicher geschrieben wird.

Für das Erstellen lokaler Kopien kommen generell zwei Ansätze in Frage: Entweder wird die Datei im Rahmen des Nachrichtenaustausches in einer Nachricht verpackt oder es wird eine direkte Socketverbindung zwischen Client und Server aufgebaut. Bei Verwendung von Socketverbindungen muss zusätzlich mit Hilfe von Systemnachrichten ein Mechanismus zur Koordinierung der Socketverbindungen vorausgehen.

Die im Folgenden beschriebenen Konzepte für den Nachrichtenaustausch beinhalten jeweils eine kurze Analyse, ob die Dokumente auch im Rahmen des regulären Nachrichtenaus-

[1] SMB (Server Message Block) ist ein Netzwerkprotokoll für Datei- und Druckdienste. Mit Hilfe vom SMB kann der Dateizugriff von Windows auf Unix-basierte Systeme ermöglicht werden.

tausches übermittelt werden können oder ob eine zusätzliche Socketverbindung notwendig ist.

4.2.5. Lösungsmöglichkeiten für den Nachrichtenaustausch

In diesem Abschnitt werden zwei verschiedene Paradigmen zum Aufbau der Kommunikationsinfrastruktur vorgestellt. Zum einen der klassische Ansatz Dienste bereitzustellen, die über eine Art von Remote Procedure Call aufgerufen werden und zum anderen der Aufbau einer Blackboardarchitektur, über welche die einzelnen Partner miteinander kommunizieren. Für beide Ansätze soll gezeigt werden, welche Konsequenzen sich dabei für die Umsetzung der Middleware ergeben und an welchen Stellen daraus weitere Probleme resultieren.

4.2.5.1. RPC-basierter Nachrichtenaustausch

Nachrichtenaustausch

Bei der Umsetzung mit Hilfe einer RPC-basierten Technologie, z. B. mittels CORBA oder SOAP, muss dem Client immer bekannt sein, an wen er eine Nachricht sendet. Systemnachrichten lassen sich damit sehr leicht umsetzen, da der Empfänger immer der Server ist. Hierfür muss der Client lediglich wissen, wie er mit dem Server kommunizieren kann. Bei Interaktionsnachrichten sind die Empfänger ebenfalls bekannt. Es tritt aber die Schwierigkeit auf, dass dem Client bekannt sein muss, ob der diese Nachricht versteht. Der als Server agierende Client muss hierfür die von ihm angebotenen Dienste für alle weiteren Clients veröffentlichen. Bei Informationsnachrichten tritt zusätzlich das Problem der Verteilung von Nachrichten auf. Der Client weiß erst einmal nicht, an wen er die Nachricht überhaupt verschicken soll.

Für die zuverlässige Übermittlung der Interaktions- und Informationsnachrichten werden drei Lösungsansätze vorgestellt:

- **Peer-to-Peer:** Jeder Client bietet einen Dienst an, in dem er die von ihm angebotenen Dienste veröffentlicht. Meldet sich ein Client am System an, fragt er bei allen anderen angemeldeten Clients, welche Dienste diese anbieten. Erfährt ein Client von der Anmeldung eines weiteren Clients, muss er diesen fragen, welche Dienste dieser anbietet. Schwierigkeiten ergeben sich hier, wenn angebotene Dienste zur Laufzeit hinzugefügt oder entfernt werden. Dieser Fall tritt auf, sobald eine Anwendung, welche spezielle Dienste bereitstellt, gestartet bzw. beendet wird. Entweder muss die Abfrage der aktuellen Dienste regelmäßig wiederholt werden oder der Client benachrichtigt alle weiteren Clients über die Veränderung der angebotenen Dienste.

- **Peer-to-Peer mit Registrierung am Server:** Jeder Client registriert die von ihm angebotenen Dienste an einem Server. Bei Beendigung eines Dienstes muss dieser am Server entsprechend deregistriert werden. Alle weiteren Clients fragen den Server nach den angebotenen Diensten der anderen Clients. Für die Gewährleistung aktueller Daten muss entweder diese Abfrage regelmäßig wiederholt werden oder es wird vom Server ein Benachrichtigungsdienst bereitgestellt, der die anderen Clients über Veränderungen der Dienste informiert.
- **Client-Server:** Für die Kommunikation zwischen den einzelnen Clients fungiert der Server als zentraler Nachrichtenverteiler. Alle Clients registrieren und deregistrieren die von ihnen angebotenen Dienste, bzw. die Informationsnachrichten, an denen sie interessiert sind. Alle Nachrichten werden an den Server gesendet, welcher die Nachricht an einen oder mehrere Clients weiterleitet.

Damit bei einem plötzlichen Wegfall eines Clients keine unerreichbaren Dienste mehr angeboten werden, wird generell ein zusätzlicher Mechanismus zur regelmäßigen Verifizierung eines Dienstes benötigt. Dieses lässt sich durch regelmäßiges Wiederholen der Veröffentlichung der Dienste bzw. dem regelmäßigen Abfragen über die Verfügbarkeit der Dienste realisieren.

Die drei vorgestellten Lösungsansätze unterscheiden sich maßgeblich in der Verteilung der Verantwortlichkeit für einzelne Aufgaben. Bei einer reinen Peer-to-Peer Lösung liegt es in der Verantwortung der Clients, die Nachrichten direkt an andere Clients zu übermitteln, herauszubekommen welche Dienste von einzelnen Clients angeboten werden und sich zu informieren, welche Informationsnachrichten an welchen Client zu versenden sind. Bei der Umsetzung als Peer-to-Peer mit Registrierung am Server liegt die Verantwortung der Nachrichtenverteilung noch immer bei den Clients. Die Clients sind aber nicht in der Verantwortung alle anderen Clients über ihre eigenen angebotenen Dienste zu informieren, sondern können diese Informationen über den Server verteilen.

Bei einer reinen Client-Server-Architektur ist der Server alleine dafür verantwortlich, die Nachrichten ausschließlich, dafür aber zuverlässig an die Clients zuzustellen, die diese Nachricht auch verstehen. Die einzelnen Clients haben nur noch die Verantwortung, Informationsnachrichten dem Server zu melden.

Dokumentaustausch

Dokumente können bei RPC-basierten Lösungen generell direkt in den Nachrichten verpackt und übermittelt werden[2]. Dieser Weg eine Datei zu übertragen ist generell unperformanter als die Übertragung per Socketverbindung oder gar dem direkten Zugriff mittels SMB-Share. Der Vorteil dieser Lösung ist jedoch, dass sich auf diese Weise Dokumente und die dazugehörigen Metadaten zusammen in einer Nachricht übermitteln lassen.

[2] z. B. als ByteArray bei einer Java-basierten Lösung

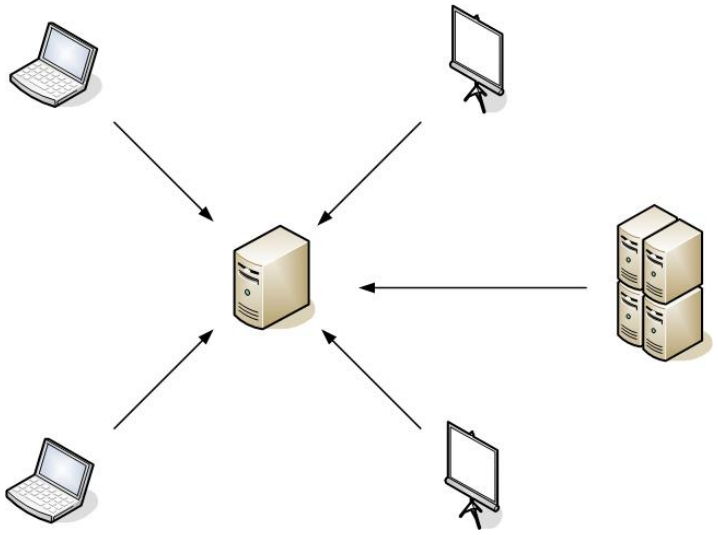

Abbildung 4.3.: Physikalische Kommunikationswege bei blackboardbasierter Kommunikation

4.2.5.2. Blackboardbasierter Nachrichtenaustausch

Nachrichtenaustausch

Bei der Umsetzung mit Hilfe blackboardbasierter Systeme, findet überhaupt keine direkte Kommunikation zwischen den Clients und auch nicht zwischen den Clients und dem Server statt. Sämtliche Nachrichten werden an das Blackboard gesendet, welches die Nachrichten lediglich speichert, aber nicht direkt weiterleitet. Es liegt in der Verantwortung des Servers bzw. der weiteren Clients, sich die Nachrichten, an denen diese interessiert sind vom Blackboard abzuholen. Nachrichten werden also nicht direkt übermittelt, sondern lediglich bereitgestellt. Auch die Rolle des Servers muss entsprechend neu interpretiert werden. Dieser nimmt bei der Kommunikation keine Sonderrolle mehr ein, sondern agiert wie die anderen Clients auch, indem er selbst am Blackboard anmelden muss, welche Nachrichten für ihn interessant sind. Abbildung 4.3 zeigt die Kommunikationswege zwischen den einzelnen Geräten inklusive dem Server bei Einführung eines zentralen Blackboards.

Es entsteht eine völlige Entkopplung der einzelnen Geräte, wobei weder der Client selbst, noch der Server dafür zuständig ist, dass andere Geräte Nachrichten erhalten. Die Verantwortung der einzelnen Clients besteht lediglich in der Bereitstellung der Nachrichten, nicht mehr in deren Zustellung.

Wie die folgende Analyse zeigt, hat die vollständige Entkopplung direkte Konsequenzen für die Umsetzung der einzelnen Nachrichtentypen im Konferenzraum:

- **Systemnachrichten:** Durch den Wegfall der direkten Kommunikation zwischen einem Client und dem Server muss auch der Austausch von Systemnachrichten, bei den die Kommunikationspartner eigentlich eindeutig sind, mit Hilfe des Blackboards realisiert werden. Der Server muss am Blackboard entsprechend anmelden, dass dieser an allen Systemnachrichten interessiert ist. Nach Verarbeitung der Nachricht wird eine entsprechende Rückantwort mit dem Ergebnis des Dienstes auf das Blackboard gelegt. In der Rückantwort muss eindeutig gekennzeichnet sein, auf welche Nachricht dieses die Rückantwort ist. Der Client blockiert seinen Prozess bis dieser die als Rückantwort gekennzeichnete Nachricht auf dem Blackboard vorfindet.
- **Interaktionsnachrichten:** Für die Interaktion zwischen den einzelnen Clients ist es notwendig, dass die einzelnen Clients mit Hilfe des Blackboards die von ihnen unterstützen Interaktionsnachrichten veröffentlichen. Durch die Veröffentlichung mittels einer dauerhaft verfügbaren Nachricht auf dem Blackboard, kann jeder Client zu jeder Zeit erkennen, welche Dienste von welchem Gerät angeboten werden. Beim Hinzufügen und Entfernen eines Dienstes wird die Nachricht auf dem Blackboard entsprechend aktualisiert. Die Clients sind also verpflichtet, auf Aktualisierung der Dienste zu warten. Bei der Nutzung eines Dienstes, der keinen Systemdienst darstellt, muss die entsprechende Nachricht zum Nutzen des Dienstes unbedingt kennzeichnen, wessen Dienst aufgerufen wird und wie dieser Dienst genau heißt.

  Als Alternative zur Verteilung mittels Informationsnachrichten kann auch, wie als Lösung bei der RPC-basierten Kommunikation vorgeschlagen, der Server als Registrierungsstelle für Dienste fungieren. Es müsste ein Systemdienst zur Registrierung und Deregistrierung vom Server bereitgestellt werden.
- **Informationsnachrichten:** Für das Verteilen von Informationsnachrichten muss eine entsprechende Nachricht lediglich auf dem Blackboard platziert werden. An welchen Informationsnachrichten ein Client interessiert ist, weiß jeder Client am besten selber und muss für den Erhalt dieser Nachrichten lediglich auf eine entsprechend gekennzeichnete Nachricht auf dem Blackboard warten.

Ein weiteres Merkmal der blackboardbasierten Kommunikation ist die fehlende Rückmeldung über den Erhalt einer Nachricht. Damit der Client eine entsprechende Rückmeldung über die Zustellung einer Nachricht erhält, muss bei Erhalt eine entsprechende Empfangsbestätigung auf dem Blackboard abgelegt werden. Wenn diese Nachricht nach einer bestimmten Wartezeit nicht auf dem Blackboard vorhanden ist, muss der Client davon ausgehen, dass diese Nachricht nicht empfangen wurde und entsprechend darauf reagieren.

Die indirekte Kommunikation mit Hilfe des Blackboards hat zur Folge, dass alle mit dem Blackboard kommunizierenden Geräte die dort abgelegten Nachrichten unbemerkt lesen können. Informations- und Interaktionsnachrichten sind generell unkritische Daten, die von allen Geräten gelesen werden können, bzw. von allen Geräte gelesen werden sollen. Bei

Systemnachrichten hingegen werden Daten ausgetauscht die von niemand anderem gelesen werden dürfen, z. B. Benutzerkennwörter. An dieser Stelle sind weitere Konzepte notwendig, damit Daten nicht in unverschlüsselter Form lesbar sind. Eine Möglichkeit hierfür ist die Verwendung von Kerberos.

Dokumentaustausch

Da bei blackboardbasierter Kommunikation grundsätzlich drei Parteien beteiligt sind, muss jede Nachricht stets zweimal über das Netzwerk übertragen werden. Bei dem normalen Nachrichtenaustausch kann dieser Zusatzaufwand vernachlässigt werden, da die Größe einer Nachricht auch im ungünstigsten Fall nur wenige Kilobytes beträgt. Werden Dokumente in diesen Nachrichten verpackt, führt die doppelte Übertragungsdauer für den Benutzer zu einer spürbaren Wartezeit. Hinzu kommt, dass eine Nachricht mindestens bis zu ihrer Zustellung gespeichert bleibt. Sollten viele Dokumente gleichzeitig ausgetauscht werden, kann es dadurch zu Speicherproblemen seitens des Blackboards kommen. Für den Dokumentenaustausch kann die Blackboardkommunikation generell als ungeeignet eingestuft werden.

4.2.6. Auswahl einer Middlewaretechnologie

4.2.6.1. Rahmenbedingung

Als Entscheidungsgrundlage wird generell davon ausgegangen, dass sich die Geräte in einer gutartigen Umgebung befinden. Voraussetzung einer gutartigen Umgebung ist, dass nur die für diese Middleware entworfene und nicht durch dritte modifizierte Software Zugriff auf die Kommunikationsinfrastruktur nimmt. Diese Annahme wird dahingehend jedoch relativiert, dass eine Zugriffsbeschränkung für das Netzwerk, insbesondere für den Zugriff über WLAN, vorhanden ist. Konsequenzen für den Wegfall dieser Annahmen werden im späteren Verlauf dieses Kapitels diskutiert.

4.2.6.2. Entscheidung

Wie aus Kapitel 4.2.5 ersichtlich wird, kann für die Middleware sowohl ein RPC-basierter, wie auch ein blackboardbasierter Ansatz gewählt werden; beide Ansätze bieten verschiedene Vor- und Nachteile. Für diese Arbeit wird der blackboardbasierte Ansatz gewählt. Durch den Pollingmechanismus und die Einführung eines zentralen Nachrichtenspeichers kann die Middleware einen sehr hohen Grad an Entkopplung bieten. Bei einer RPC-basierten Lösung kann dieses lediglich durch die zentrale Nachrichtenverteilung mit Hilfe des Servers erreicht werden. Die Verantwortung des Servers reduziert sich automatisch auf die Bereitstellung der Systemdienste, somit ist bei der Entwicklung neuer, auf die Middleware aufbauender Anwendungen keine Anpassung des Servers notwendig. Durch die Reduzierung

der Verantwortlichkeiten der Clients auf die Bereitstellung von Nachrichten wird auch die Integration von ressourcenarmen Geräten, wie PDAs, Handhelds, TablesPCs und sogar Mobiltelefonen erleichtert. Diese Geräte müssen lediglich in der Lage sein mit dem Blackboard zu kommunizieren, damit diese die grundlegenden Systemdienste nutzen können.

Für die Umsetzung wird der in Kapitel 3.3.3 vorgestellte Event Heap verwendet. In dem vergleichbaren Projekt Interactive Workspaces hat der Event Heap seine Tauglichkeit als Tupelraumimplementierung grundsätzlich bewiesen. Performancetests in [Johanson 2002] haben gezeigt, dass der Event Heap auch mehreren hundert eintreffenden Events pro Sekunde standhalten kann. Die Verwendung des Event Heaps lässt auch die Möglichkeit offen, einzelne Unterprojekte des Interactive Workspace in dieses Projekt einfließen zu lassen. Besonders interessant ist hier die Anwendung PointRight (siehe [Johanson u. a. 2002b]), mit der sich die Maus- und Tastatursteuerung von einem entfernten Rechner aus durchführen lässt. Der Event Heap ist im Quellcode verfügbar und somit offen für Erweiterungen. Ein weiterer Grund für die Verwendung einer Tupelraumtechnologie ist die sehr geringe Größe der für den Client benötigten Bibliotheken. Für die Verwendung des Event Heaps beträgt diese lediglich 48 KB [Johanson 2002]. Somit können auch Handhelds und Mobiltelefone als Clients eingesetzt werden.

Die Verwendung des Event Heaps impliziert die Verwendung von Java als Programmiersprache für die Umsetzung. Bezüglich der Anforderungen ist diese sogar zu empfehlen, Laufzeitumgebungen für Java sind bei einer Vielzahl von Plattformen, inklusive moderner Handhelds und Mobiltelefonen, bereits integriert.

4.2.6.3. Konsequenzen bei Änderung der Sicherheitsvoraussetzungen

Ein Blackboard ist grundsätzlich offen bezüglich des Lesens der abgelegten Nachrichten, wodurch ein unbemerktes Mitlesen der Nachrichten durch Dritte ermöglicht wird. Für sicherheitskritische Nachrichten muss daher ein Verschlüsselungsmechanismus in Betracht gezogen werden.

Systemdienste sind generell als sicherheitskritisch einzustufen. Für deren Umsetzung bietet sich der RPC-basierten Ansatz an. Wie in Kapitel 4.2.5 dargestellt, hat der RPC-basierte Ansatz bei Systemdiensten keine Nachteile gegenüber dem blackboardbasierten Ansatz. RPC-basierte Ansätze bieten auch die Möglichkeit der Nachrichtenverschlüsselung.

Eine nachträgliche Änderung der Kommunikationsinfrastruktur ist mit hohem Entwicklungsaufwand verbunden. Dieser lässt sich minimieren, indem die Kommunikationsschnittstelle als eigenständige Komponente realisiert wird. Die Schnittstellen der Kommunikationskomponente müssen dabei unabhängig von der verwendeten Technologie gewählt werden, damit auch bei nachträglicher Änderung der Implementierung die Schnittstellen ihre Gültigkeit behalten.

4.3. Definition von Designprinzipien für den Nachrichtenaustausch

4.3.1. Einführung von Pflichtfeldern

Bei der Verwendung einer Tupelraumtechnologie ist es notwendig, Regeln für das Design der Tupel festzulegen. Der Event Heap gibt bereits eine Reihe von Tupelfeldern vor. Dies sind vor allem Felder mit Informationen über den Versender und dem Empfänger einer Nachricht. Das Feld Target ist dafür vorgesehen, den Computernamen des Empfängers zu definieren. Dieses bedeutet nicht, dass nicht auch andere Computer diese Nachricht lesen können, jedoch lässt sich dieses Feld für die Filterung der Nachrichten nutzen.

Ein vorgegebenes Feld, welches in jeder Nachricht angegeben werden muss, ist das Feld *Eventtype*. Der *Eventtype* wird dafür genutzt, um die Nachrichten nach den Nachrichtentypen Systemnachricht, Interaktionsnachricht und Informationsnachricht zu unterscheiden. Diese Unterscheidung ist besonders für die Behandlung von Systemnachrichten bedeutend. Systemnachrichten repräsentieren grundsätzlich den Aufruf eines Serverdienstes. Neben dem vorgegebenen Feld *Eventtype*, wird das Feld *Subtype* eingeführt. Dieses dient zur genauen Benennung des Dienstes. Bei Verwendung einer RPC-basierten Technologie, wäre das Feld *Subtype* mit dem Methodennamen vergleichbar.

4.3.2. Rückgabewerte aufgerufener Dienste

In der klassischen objektorientierten Programmierung haben Methoden i.d.R. einen Rückgabewert. Auch bei verteilten objektorientierten Systemen haben Methoden Rückgabewerte. Bei objektorientierter Interprozesskommunikation, z. B. mittels Java RMI oder CORBA, wird nach dem entfernten Methodenaufruf ein Rückgabewert an den Aufrufenden übertragen. Bei tupelraumbasierter Kommunikation werden lediglich Tupel in einem öffentlichen Raum abgelegt, die Übermittlung eines Rückgabewertes muss also ebenfalls über das Ablegen eines Tupels umgesetzt werden. Jedes abgelegte Tupel ist über eine Sessionnummer und eine Sequenznummer, die automatisch vom Client beim Ablegen des Tupels generiert werden, eindeutig identifizierbar. Mit Hilfe dieser Felder kann auch die Übermittlung eines Rückgabewertes umgesetzt werden. Hierzu muss der Empfänger einer Nachricht bei der Übermittlung des Rückgabewertes explizit angeben, auf welche Nachricht diese Nachricht sich bezieht. Zusätzlich wird an den *Eventtype* der Suffix `_response` angehängt. Der Prozess, der den Dienst aufgerufen hat, muss explizit auf den Erhalt dieser Nachricht warten.

Sollte bei der Ausführung eines Dienstes ein Fehler auftreten, wird der aufrufende Client mit Hilfe der Antwortnachricht über diesen informiert. Hierzu wird in dem optionalen Feld *Exceptiontype* der Typ des Fehlers, in Form der Java-Exceptionklasse, angegeben.

4.4. Entwicklung der Clientanwendung

Die Clientanwendung Kora stellt die allgemeine Steuerungsanwendung für den Konferenzraum dar. Die Entwicklung wird dabei auf die Middleware fokussiert, welche eine Schnittstelle definiert, die die in Kapitel 2.3 identifizierten funktionalen Anforderungen zur Verfügung stellt. Neben den bekannten Anforderungen muss die Schnittstelle auch Gültigkeit für weitere Applikationen, die auf dieser Middleware aufbauen, besitzen. Für beide Aspekte der Schnittstelle gilt es, Methoden bereitzustellen, welche unabhängig von der verwendeten Technologie für den Nachrichtenaustausch benutzt werden können.

4.4.1. Verwendung des MVC-Konzeptes

Die Entwicklung einer Clientanwendung wird durch Unterteilung der Anwendung in verschiedene kleine Module mit definiertem Aufgabenbereich und definierten Schnittstellen zu angrenzenden Modulen durchgeführt. Der grundsätzliche Aufbau der Anwendung sollte dem Model-View-Controller Muster folgen. Das Model bezeichnet hier das Datenmodell der Anwendung. Die Views sind die Benutzeroberflächen, in denen der Benutzer die Anwendungsdaten angezeigt bekommt und Änderungen an ihnen vornimmt. Der Controller ist für die Steuerung der Anwendung zuständig. Dieser nimmt die eingegebenen Änderungen entgegen und führt entsprechend der Anwendungslogik die Änderungen am Datenmodell durch.

4.4.2. Abstraktion der Kommunikationskomponente

Die Interprozesskommunikation wird von einer eigenen Komponente durchgeführt. Für diese Kommunikationskomponente ist eine eigene Abstraktionsschicht einzuführen, welche die identifizierten logischen Nachrichten von den tatsächlichen ausgetauschten Nachrichten trennt. Durch diese Trennung werden eine Reihe wichtiger Qualitätsmerkmale gesichert. Die Anwendungslogik ist von der Kommunikationsschicht entkoppelt, dadurch verringert sich die Komplexität der gesamten Anwendung auf die Komplexität der einzelnen Module. Die Wartbarkeit der einzelnen Module wird entsprechend erhöht. Durch die Definition von Schnittstellen, welche unabhängig von der Implementierung der Kommunikation sind, ist

die Kommunikationskomponente generell austauschbar, ohne dass dabei Änderungen am Controller durchzuführen sind.

Die Abstraktion der Kommunikationskomponente kann nicht nur aus der Definition einer Schnittstelle z. B. in Form eines Java-Interfaces bestehen. Parameter, dessen Inhalte technologieabhängig sind, müssen gekapselt werden und in technologieunabhängigen Klassen übergeben werden. Ein technologieabhängiger Parameter ist der Empfänger einer Nachricht. Der verwendete Schlüssel hierfür kann der Computername, eine Netzwerkadresse oder eine Referenz auf ein technologieabhängiges Objekt sein. Der Schlüssel zur Identifikation des Empfängers wird von der Kommunikationskomponente in einem allgemeinen Objekt für Nachrichtenempfänger verpackt. Die Abstraktion des Nachrichtenempfängers wird durch die Klasse `KoraParticipant` realisiert. Weitere Klassen der Abstraktionsschicht werden in Kapitel 4.4.3 identifiziert.

4.4.3. Empfang asynchroner Nachrichten

Aus den identifizierten Nachrichten aus Kapitel 4.2.1 geht hervor, dass Interaktions- und Informationsnachrichten generell asynchrone Nachrichten sind. Die Anwendung muss auf diese Nachrichten warten, darf hierfür jedoch nicht blockieren. Der Empfang asynchroner Nachrichten wird mit Hilfe von Threads umgesetzt. Beim Warten auf eine Nachricht wird der entsprechende Thread blockiert bis eine Nachricht eingetroffen ist. Das Warten auf eine Nachricht liegt im Aufgabenbereich der Kommunikationskomponente, welche jedoch nicht für die Verarbeitung von Nachrichten zuständig ist. Zu diesem Zweck werden Callbackobjekte eingeführt, welche als Schnittstelle zwischen Kommunikationskomponente und Anwendungslogik fungieren. Callbackobjekte, realisiert durch die Klasse `KoraCallback`, beinhalten keine Anwendungslogik, halten aber eine Referenz zu dieser, um entsprechende Methoden aufzurufen. Damit ein Callbackobjekt entscheiden kann, welche Methoden der Applikation aufzurufen sind, müssen diese in der Lage sein, die empfangende Nachricht zu verstehen. Callbackobjekte müssten hierzu eng an die Kommunikationskomponente gekoppelt werden. Um dieses zu vermeiden wird, die Klasse `KoraMessage` zur Abstraktion von Nachrichten eingeführt. Die Kommunikationskomponente reicht die abstrahierte Nachricht an das Callbackobjekt weiter, welches anhand der abstrahierten Nachricht erkennen kann, welche Methoden der Anwendungslogik aufzurufen sind.

Bei der Verwendung von Threads ist besonders auf die Vermeidung von Seiteneffekte, ausgelöst durch die nebenläufige Programmierung, zu achten. Bei Threadübergreifenden Programmabschnitten, besteht die Gefahr, dass einzelne Variablen während der parallelen Ausführung der Methoden durch einen anderen Thread ungewollt verändert werden und es so zu schwer identifizierbaren Programmfehlern kommt. Threadübergreifende

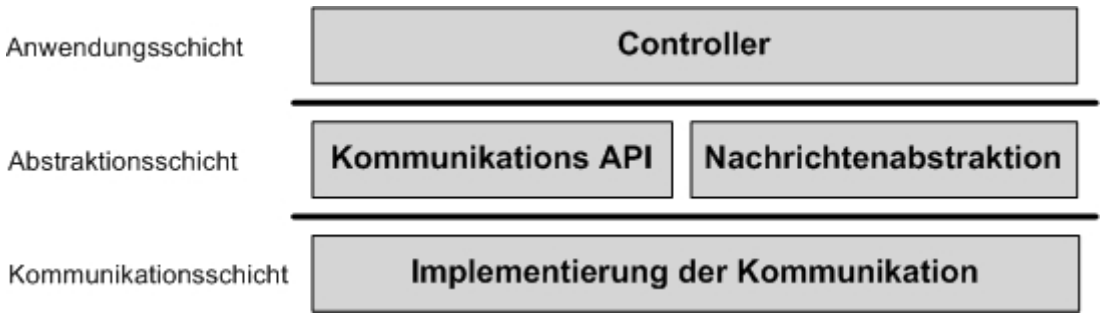

Abbildung 4.4.: Aufteilung der Schichten der Clientanwendung

Programmabschnitte sind vor allem statische Klassenmethoden und mit dem Singleton-Pattern[3] realisierte Klassen. Entsprechende Abschnitte müssen mit dem Java Schlüsselwort `synchronized` markiert werden, damit diese nicht von mehreren Threads gleichzeitig ausgeführt werden können.

4.4.4. Übersicht über die Komponenten und Schnittstellen

Durch die Kapselung der Kommunikationskomponente von der Anwendungslogik sowie der Einführung einer Abstraktionsschicht für die Kommunikation ergibt sich für die Clientanwendung ein Schichtenmodell bestehend aus drei Ebenen, wie Abbildung 4.4 grafisch darstellt. An oberster Stelle steht die Anwendungsschicht, welche aus Datenmodell, Benutzeroberfläche und Anwendungslogik besteht. Die Abstraktionsschicht beinhaltet ein Interface mit den Methoden, die von der Middleware zur Verfügung gestellt werden müssen. Die in diesem Interface definierten Methoden folgen den funktionalen Anforderungen bzw. den daraus identifizierten logischen Nachrichten. Zur Parameterübergabe an die darunterliegende Kommunikationsschicht werden auf Ebene der Abstraktionsschicht die drei in den Kapiteln 4.4.2 und 4.4.3 identifizierten Klassen `KoraMessage`, `KoraParticipant` und `KoraCallback` definiert.

Die Middleware bietet eine Reihe von Methoden, die speziell den Anforderungen zur Steuerung des Konferenzraumes nachkommen. Zusätzlich muss eine Schnittstelle für weitere zukünftig entwickelte Anwendungen definiert werden. Diese allgemeine Schnittstelle benötigt vor allem Methoden zum Senden von Nachrichten und dem Empfangen von asynchronen Nachrichten. Hierzu wird die in Kapitel 4.4.3 eingeführte Nachrichtenabstraktion `KoraMessage` verwendet. Mit Hilfe dieser Nachrichtenabstraktion können weitere Anwendungen die Middleware nutzen, ohne dass Änderungen an der Kommunikationskomponente notwendig sind. Zum Empfang von Nachrichten dient ebenfalls die Klasse

[3] Das Singleton-Pattern sichert ab, dass es zu einer Klasse nur eine einzige Instanz gibt, welche über einen globalen Zugriffspunkt erreichbar ist. [Gamma u. a. 1995]

`KoraMessage`. Die darin abgebildete Nachricht dient als Vorlage für die zu empfangende Nachricht.

Die für die Abstraktion der Kommunikation notwendigen Komponenten sind in Abbildung 4.5 mit ihren Eigenschaften und Methoden abgebildet. Die Klassen `KoraMessage` und `KoraParticipant` werden als JavaBeans[4] umgesetzt. Auf die Darstellung der entsprechenden Get- und Set-Methoden wird in diesem Modell verzichtet.

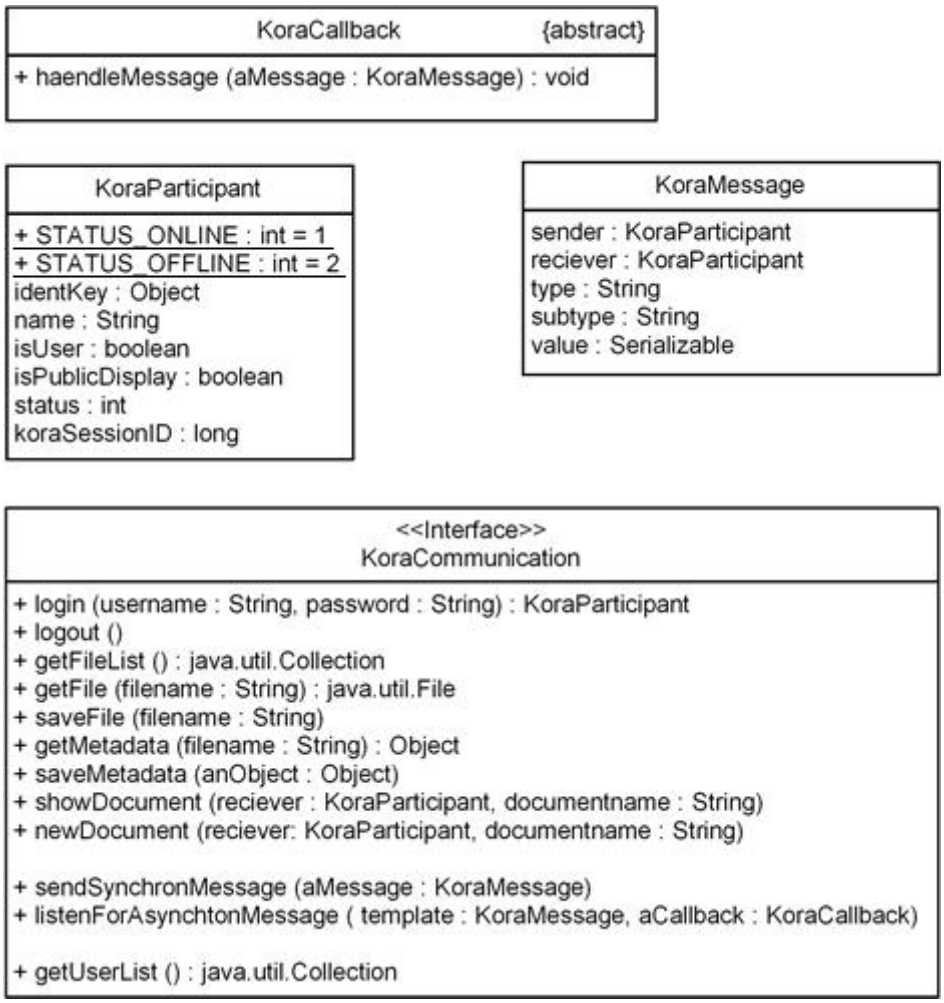

Abbildung 4.5.: UML-Darstellung der Kommunikationsabstraktion

[4] JavaBeans sind Klassen, die als Datencontainer dienen. Die Instanzvariablen einer JavaBean sind nur über die öffentlichen Zugriffsmethoden `getProperty()` und `setProperty(value)` erreichbar.

Generische Methoden zum Anbieten und Aufrufen von Diensten

Die Schnittstelle verfügt über zwei generische Methoden, mit denen sich weitere Nachrichten erstellen lassen, ohne die Kommunikationsschnittstelle erweitern zu müssen.
Die Methode `sendSynchronMessage(KoraMessage):KoraMessage` nimmt eine Nachricht entgegen und versendet diese. Wenn die Nachricht einer direkten Antwort bedarf, wartet die Methode bis eine direkte Antwortnachricht eingetroffen ist. Mit Hilfe dieser Nachricht lässt sich der Aufruf von neuen Diensten realisieren. Mit der Methode `listenForAsynchronMessage(KoraMessage, KoraCallback):void` lassen sich neue Dienste anbieten. Hierzu wird eine Vorlagenachricht übergeben, die beschreibt, auf welche Nachrichten dieser Dienst reagiert. Das Callbackobjekt registriert die auszuführende Operation bei Aufrufen dieses Dienstes.

4.5. Entwicklung der Serveranwendung

Die Serveranwendung ist eine zentrale Anwendung, welche die klassischen Aufgaben eines Servers ausführt. Die Anwendung stellt eine Reihe von Diensten bereit, welche den Zugang zum System und den Zugriff auf Dokumente bzw. Dokumentmetadaten ermöglicht. Der Server ist dabei in der Verantwortung, die Zugriffsrechte zu wahren, indem jede Nutzung eines Dienstes eine Berechtigungsprüfung vorausgeht. Die angebotenen Dienste entsprechen den in Kapitel 4.2.1 identifizierten Systemnachrichten.

4.5.1. Schichtentrennung

Der Aufbau der Anwendung erfolgt, wie in Abbildung 4.6 gezeigt, in drei logischen Schichten. Der Aufruf eines Dienstes wird von der Kommunikationsschicht empfangen. Diese ist abhängig von der verwendeten Middlewaretechnologie und muss daher als eigenständiges Modul realisiert werden. Von der Kommunikationsschicht wird der Aufruf an die Serviceschicht, welche die einzelnen Dienste zusammenfasst, weitergeleitet. Die Dienste nehmen Zugriff auf die Datenschicht, in der die Dokumente liegen und die Benutzerrechte abgebildet sind. Grundsätzlich sind die einzelnen Dienste unabhängig voneinander, wodurch das Hinzufügen neuer Dienste erleichtert wird, es werden jedoch auch Dienste definiert, die nicht öffentlich zugänglich sind, sondern nur von den öffentlichen Diensten genutzt werden. Ein Autorisierungsdienst zur Überprüfung, ob der Benutzer, der einen Dienst aufgerufen hat, auch tatsächlich am System angemeldet ist und auch das Recht zur Ausführung dieses Dienstes hat, muss von allen öffentlichen Diensten genutzt werden. Nach Ausführung des Dienstes ist die Kommunikationsschicht dafür verantwortlich, das Ergebnis des Dienstes an den aufrufenden Client zurückzusenden.

Abbildung 4.6.: Aufteilung der Schichten der Serveranwendung

Die nachfolgende Liste beschreibt die vom Server angebotenen Dienste. Alle angebotenen Dienste werden dabei mit dem Suffix *Service* versehen

- *LoginService* führt die Authentifizierung am System durch und startet eine Session.
- *LogoffService* beendet eine Session.
- *GetFileListService* gibt eine Liste mit allen Dateien des Benutzers zurück.
- *GetFileService* führt für eine bestimmte Datei den Dateitransfer zum Client durch.
- *SaveFileService* führt den Dateitransfer zum Server durch.
- *GetDocumentMetadataService* liefert Metadaten zu einer gegebenen Datei.
- *SaveDocumentMetadataService* speichert Metadaten zu einer gegebenen Datei.

Alle vom Server angebotenen Dienste werden durch den Wert Systemnachricht im Pflichtfeld *Eventtype* identifiziert. Im Feld *Subtype* steht der Name des Dienstes. Es entfällt die Notwendigkeit, für Systemnachrichten einen expliziten Empfänger angeben zu müssen. Hierdurch entsteht zusätzlich eine Lokationstransparenz für die einzelnen Dienste. Dieses wiederum ermöglicht eine Lastverteilung durch Verteilung der einzelnen Dienste auf verschiedene Server. Die Definition der Tupelfelder, die für die Umsetzung der hier beschriebenen Dienste notwendig sind, befinden sich in Anhang B.

4.5.2. Transfer von Dokumenten mittels Socketverbindungen

Für den Dateitransfer zwischen Client und Server wurde in Kapitel 4.2.5.2 eine Blackboardarchitektur für ungeeignet erklärt. Dieses wirft die Notwendigkeit auf, die Datei mit Hilfe einer Socketverbindung direkt zwischen Client und Server zu übertragen. Im Gegensatz zu den anderen Diensten ist der *GetFileService* und *SaveFileService* somit direkt technologieabhängig. Um eine Socketverbindung zu dem Client aufzubauen, muss diese zwischen den

Parteien koordiniert werden. Für den Client bedeutet das, dass er vom Server eine Portnummer übermittelt bekommt, auf welcher die Datei übertragen wird. Durch die Asynchronität der Aufrufe von Diensten muss der Server darauf achten, die Portnummern zu keinem Zeitpunkt mehrfach zu belegen. Hierzu wird seitens des Servers ein Pool mit freien Portnummern erstellt. Für jede Verbindung muss der Server eine freie Portnummer aus diesem Pool reservieren.

4.6. Integration weiterer Anwendungen

Die Clientanwendung Kora ist für den Benutzer die Hauptanwendung während des Aufenthaltes im Konferenzraum. Zum Betrachten bzw. Bearbeiten von Dokumenten sind weitere Anwendungen nötig. Besonders zum Öffnen und Speichern von Dokumenten müssen diese Anwendungen auf der hier beschriebene Middleware aufsetzen. Hierfür werden vier Varianten vorgestellt:

1. **Neuentwicklung von Anwendungen:**

   Für das Öffnen und Speichern von Dokumenten nimmt diese direkten Zugriff auf die vorhandene Middleware. Durch die direkte Anbindung an die Middleware kann diese Anwendung auch als verteilte Anwendung umgesetzt werden. Die Anwendung nutzt dabei den Event Heap, um mit derselben Anwendung auf einem anderen Rechner zu kommunizieren. Mögliche Anwendungsbeispiele für neu entwickelte Anwendungen: Texteditor, verteiltes Brainstorming, Metadateneditor.

2. **Modifizierung von Anwendungen die im Quellcode vorliegen:**

   Wie bei der Neuentwicklung einer Anwendung, nimmt eine modifizierte Anwendung direkten Zugriff auf die Middleware. Hierfür müssen vor allem die Teile der Anwendung umgeschrieben werden, die mit der Dateiein- und ausgabe zusammenhängen. Die Modifizierung kann auch soweit gehen, dass die Anwendung mit weiteren Anwendungen direkt interagiert.

3. **Verwendung anwendungseigener Skriptsprachen und eines externen Adapters:**

   Es gibt Anwendungen, die direkt über Skriptsprachen steuerbar sind. Hierdurch kann die Anwendung zwar weder mit anderen Anwendungen interagieren, noch kann das Öffnen und Speichern über die Middleware umgesetzt werden, dafür besteht zumindest die Möglichkeit, eine Interaktion zwischen der Anwendung und dem Adapter mit Hilfe der Skriptsprache zu realisieren. Beim Speichern eines Dokumentes kann der Adapter dieses direkt an die Middleware weitergeben, wodurch die neue Version des Dokumentes auch serverseitig gespeichert werden kann. Dieses Modell ist vor allem für Büroanwendungen wie Microsoft Office und OpenOffice.org umsetzbar.

4. **Integration mit Hilfe eines externen Adapters:**

   Anwendungen, die weder modifizierbar noch direkt steuerbar sind, müssen von einem externen Adapter aus aufgerufen werden. Diesem Aufruf wird ein Dateipfad als Parameter übergeben. Die Steuerung der Anwendung reduziert sich dabei auf Starten der Anwendung und Öffnen eines Dokumentes. Hierbei entsteht besonders das Problem, dass die Speicherung von Dokumenten mit Hilfe der Anwendung Kora oder mit Hilfe des Adapters durchgeführt werden muss. Diese Form der Integration ist für Anwendungen die keinen Zugriff auf Dokumente nehmen, z. B. Webbrowser, absolut ausreichend.

Das hier vorgestellte Modell wurde in ähnlicher Form bereits in [Bartnik 2006] beschrieben. Bartniks Modell unterschied zwei Fälle: Anwendungen die direkten Zugriff auf die Middleware nehmen und Anwendungen, die mit Hilfe eines Adapters mit der Middleware kommunizieren. Der direkte Zugriff wurde hier in Fall 1 und Fall 2 unterteilt, die Verwendung des Adapters ist in Fall 3 und Fall 4 abgebildet. Die beschriebenen vier Varianten weitere Anwendungen zu integrieren sind in Abbildung 4.7 noch einmal dargestellt.

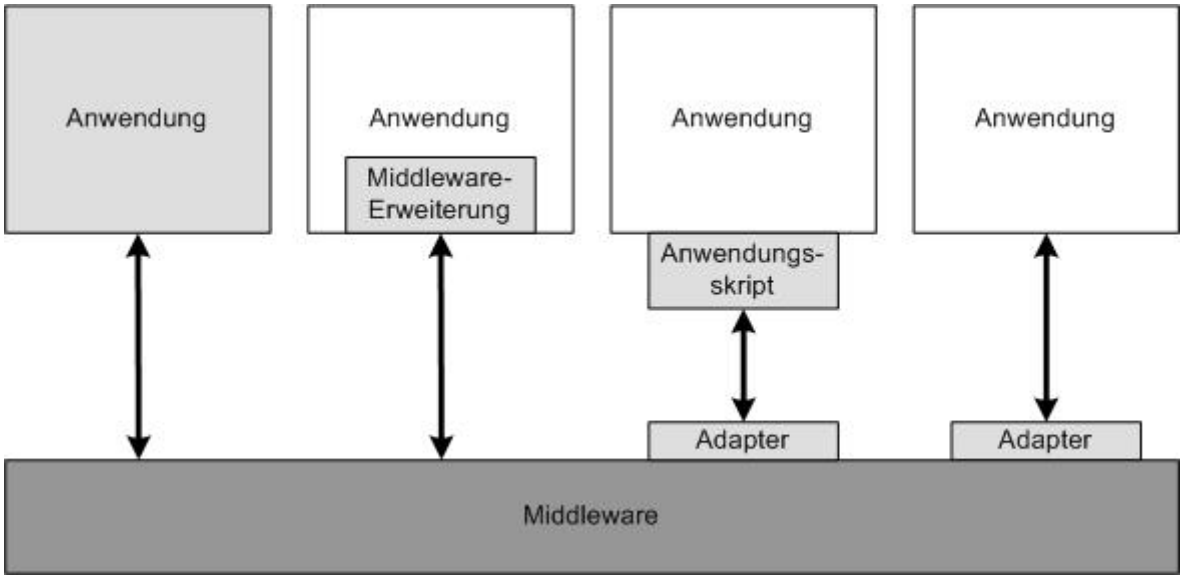

Abbildung 4.7.: Vier Varianten der Integration weiterer Anwendungen

5. Zusammenfassung und Ausblick

5.1. Zusammenfassung

Das Ziel dieser Arbeit ist es, eine Middleware für Gruppenarbeitsräume in ubiquitären Umgebungen zu konstruieren. Hierzu wurde ein konkretes Szenario, in diesem Fall die Stadtplanung, zur Hilfe genommen. Aus einer Beschreibung der Tätigkeit der Stadtplaner, einer kurzen Analyse des Ablaufs von Stadtplanungsprojekten und den verwendeten Informationstechnologien wurde ein Modell entwickelt, wie stadtplanerische Arbeit in einer kollaborativen Umgebung aussehen würde. Besonders prägend für dieses Szenario ist dabei, dass für die identifizierten Benutzer der Umgang mit solchen Systemen nicht unbedingt alltäglich ist. Aus der Analyse der Benutzer wird besonders deutlich, wie wichtig eine intuitive Bedienung für die Akzeptanz des Systems ist. Vor allem beim Vergleich mit Softwareentwicklungsprojekten ist hier ein deutlicher Unterschied zu erkennen. Weiter kommt für dieses Szenario erschwerend hinzu, dass in der Stadtplanung sehr spezielle Programme wie GIS- und CAD-Anwendungen verwendet werden. Um das Szenario Stadtplanung auf die kollaborative Arbeitsumgebung anwenden zu können wurde schließlich eine Vision des Systems entwickelt und die fachlichen Komponenten dargestellt. Um im späteren Verlauf eine konkrete Anwendung für den Betrieb des Gruppenarbeitsraumes zu entwickeln, wurden hierfür die notwendigen Anforderungen identifiziert. Diese Anforderungen gelten für das gewählte Szenario Stadtplanung, sind darüber hinaus aber allgemeingültig für den Betrieb des Arbeitsraumes und somit auch auf weitere Szenarien anwendbar.

In der Beschreibung der Systemkonfiguration wurden die einzelnen Hardwareelemente identifiziert. Es erfolgte dabei eine logische Unterteilung in die vier Teilbereiche Server, Netzwerkinfrastruktur, öffentliche Bildschirme und mobile Endgeräte. Bei den Servern wurde dabei explizit offen gelassen, auf wie viele Rechner diese aufgeteilt werden. Da bei dem Betrieb des Konferenzraumes die Anzahl der angeschlossenen Clients bei etwa 10 bis 20 liegen wird, ist die zu erwartende Last des Servers vergleichsweise gering. Durch die logische Trennung der serverseitigen Komponenten (Datenbank, Dateiserver, Systemdienstserver) ist das System grundsätzlich sehr gut skalierbar.

Im Abschnitt Interprozesskommunikation wurde eine umfangreiche Diskussion über die Verwendung einer RPC-basierten oder einer tupelraumbasierten Middlewaretechnologie

durchgeführt. Die Analyse der auszutauschenden Nachrichten hat dabei ergeben, dass diese in die drei verschiedenen Typen System-, Interaktions- und Informationsnachrichten unterteilt werden können. Diese unterscheiden sich in der Art des Inhalts und der Art ihrer Verteilung. Systemnachrichten stellen den Aufruf serverseitiger Dienste dar und werden nur für die Kommunikation zwischen Server und genau einem Client benutzt. Interaktionsnachrichten werden für direkte Interaktionen zwischen einzelnen Clients verwendet. Informationsnachrichten stellen allgemeine Informationen für die Clients dar und werden als Broadcastnachrichten versendet. In der Diskussion über eine RPC- oder tupelraumbasierte Technologie wurden für die drei Nachrichtentypen die jeweiligen Konsequenzen für die Umsetzung erläutert. Dabei stellte sich heraus, dass sich mit einer RPC-basierten Technologie die Systemnachrichten sehr gut umsetzen lassen, es aber zu Schwierigkeiten bei der Umsetzung der Interaktions- und Informationsnachrichten kommt. Für tupelraumbasierte Technologien ist der klassische Ablauf von Request und Response, wie es bei den Systemnachrichten üblich ist, etwas umständlich. Dieser lässt sich aber durch die Definition einheitlicher Tupelfelder umsetzen. Für Interaktions- und Informationsnachrichten ist eine tupelraumbasierte Lösung in jedem Fall vorzuziehen. Auch wenn der im Rahmen dieser Arbeit erstellte Prototyp fast ausschließlich Systemnachrichten verwendet, so wird sich bei einer Weiterentwicklung des Projektes die Zahl der Informations- und Interaktionsnachrichten erhöhen. Aus diesem Grunde wurde beschlossen, dass das System von Beginn an auf einer tupelraumbasierten Technologie aufgebaut wird. Eine hybride Lösung wäre aber ebenfalls denkbar, besonders wenn die Anforderungen an die Sicherheit als höher eingestuft werden sollten. Für die Realisierung wurde dabei die an der Stanford University entwickelte Event Heap Technologie gewählt.

Für das Design der Server-, wie der Clientanwendung wurde eine deutliche Trennung zwischen den technologieabhängigen und technologieunabhängigen Teilen vorgeschlagen. Dieses verringert die Komplexität des Quellcodes und schafft die Möglichkeit der Erweiterung. Zudem ermöglicht dieses einen Technologiewechsel, ohne dabei den Anwendungscontroller oder die Schnittstellen zur Kommunikationsschicht ändern zu müssen. Für die einheitliche Verwendung des Event Heaps wurden dabei eine Reihe von Designprinzipien festgelegt. Diese beziehen sich besonders auf die Tupelfelder und deren Inhalte.

Der entwickelte Prototyp hat gezeigt, dass die Umsetzung mit Hilfe einer tupelraumbasierten Technologie grundsätzlich möglich ist. Die geringe Größe des Clients zeigt dabei, wie schlank der Event Heap ist. Dadurch wird auch die Ausführung auf mobilen Geräten unterstützt, ohne hierfür besondere Einschränkungen vornehmen zu müssen. Für die Realisierung im Rahmen einer ubiquitären Umgebung sind tupelraumbasierte Technologien auf Grund ihrer geringen Komplexität und der Tatsache, dass die Kommunikation nur indirekt und mit Hilfe eines unabhängigen Servers läuft, gut geeignet.

Bei dem gewählten Szenario Stadtplanung besteht derzeit noch die Gefahr, dass die Be-

nutzer nicht die technische Ausstattung besitzen, um die Vorteile einer ubiquitären Umgebung zu nutzen.

5.2. Ausblick

Diese Arbeit hat gezeigt, dass die Entwicklung einer Middleware ein komplexes Thema ist. Die grundlegenden Konzepte wurden hierfür entwickelt und die notwendigen Dienste wurden identifiziert. Der nächste Schritt wird nun die Umsetzung dieser Konzepte sein. Dabei muss auch das in [Mund 2006] entworfene rollenbasierte Benutzermodell bedacht werden. Die Ergebnisse bezüglich der Dokumentmetadaten und ihrer Anzeige bzw. Bearbeitung, mit der sich Petra Ehlers beschäftigt, werden für dieses Projekt ebenfalls interessant sein.

Die Middleware ist bei der Realisierung des Konferenzraumes jedoch nur ein Teilaspekt. Weitere Fragen, besonders bezüglich der Interaktionsformen zwischen Menschen und Computer gilt es zu klären. Wie lassen sich die großen Bildschirme tastaturlos bedienen? Das IBM-Blueboard [Russell und Gossweiler 2001] realisiert eine intuitive Bedienung an einem öffentlichen Bildschirm, in Verbindung mit einer personalisierten Umgebung. Wie dieses Bedienungsmodell auf das Collaborative Workplace angewendet werden kann, müssen weitere Untersuchungen ergeben.

Die in Stanford für den Event Heap entwickelte Anwendung PointRight [Johanson u. a. 2002b] hat gezeigt, wie sich ein Rechner mit Maus und Tastatur eines anderen Rechners steuern lässt. Diese Funktionalität sollte dieser Konferenzraum ebenfalls umsetzen. Besonders für das Ubiquitous Computing wäre auch eine Vereinfachung der Authentifikation sehr sinnvoll. Zur Zeit müssen Benutzername und Passwort bei einem extra durchzuführendem Login eingegeben werden. Dieses ist die Mindestanforderung eines jeden benutzerorientierten Systems, widerspricht aber dem Gedanken des Ubiquitous Computings, wonach die Verbindungen einfach, schnell und spontan hergestellt werden sollen. Hier gibt es verschiedene Alternativen, die es zu untersuchen gilt. So könnte die Authentifizierung mit Hilfe eines RFID-Chips oder durch Erkennung von biometrischen Merkmalen durchgeführt werden. In diesem Zusammenhang ließe sich die Authentifikation auch dynamisch, je nach Position im Raum gestalten. So wäre bei der Benutzung der öffentlichen Bildschirme automatisch zuzuordnen, wer diese gerade benutzt.

Literaturverzeichnis

Albers 2002 ALBERS, Gerd: *Stadtplanung: Eine praxisorientierte Einführung*. 2. Auflage. Darmstadt : Wissenschaftliche Buchgesellschaft, 2002. – ISBN 3-534-03266-0

Bartnik 2006 BARTNIK, Roman: *Weiterentwicklung einer Technologiebasis für interaktive Gruppenarbeitsräume*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2006

Coors und Zipf 2005 COORS, Volker ; ZIPF, Alexander: *3D-Geoinformationssysteme*. Heidelberg : Herbert Wichmann Verlag, 2005. – ISBN 3-87907-411-9

EDV-Stadtplanung 1999 AG EDV IN DER STADTPLANUNG: *Datenverarbeitung in der Stadtplanung - praxisorientierte Ansätze*. 1999. – URL http://www.ag-edv-stadtplanung.de/ag-main.html. – Zugriffsdatum: 2006-06-01

Ehlers 2006 EHLERS, Petra, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit (in Vorbereitung), 2006

Gamma u. a. 1995 GAMMA, Erich ; HELM, Richard ; JOHNSEN, Ralph ; VLISSIDES, John: *Design Patterns. Elements of Reusable Object-Oriented Software*. Reading, MA, USA : Addison-Wesley Publishing Company, 1995. – ISBN 0-201-63361-2

Gelernter 1985 GELERNTER: Generative communications in Linda. In: *ACM Transactions on Programming Languages and Systems* 7 (1985), Nr. 1, S. 80–112

GigaSpaces GIGASPACES TECHNOLOGY LTD. (Hrsg.): *GigaSpaces*. – URL http://www.gigaspaces.com. – Zugriffsdatum: 2006-07-28

Harris 1967 HARRIS, Britton: The Limits of Science and Humanism in Planning. In: *AIP Journal* (1967), September, S. 325

Jini SUN MICROSYSTEMS (Hrsg.): *The Jini Network Technology*. – URL http://www.sun.com/jini. – Zugriffsdatum: 2006-07-28

Johanson 2002 JOHANSON, Brad: *Application Coordination Infrastructure for Ubiquitous Computing Rooms*, Stanford University, Dissertation, 2002

Johanson und Fox 2002 JOHANSON, Brad ; FOX, Armando: The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In: *4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002)* (2002), Juni

Johanson und Fox 2004 JOHANSON, Brad ; FOX, Armando: Extending Tuplespaces For Coordination in Interactive Workspaces. In: *Journal of Systems and Software* 69 (2004), Nr. 3

Johanson u. a. 2002a JOHANSON, Brad ; FOX, Armando ; WINOGRAD, Terry: The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. In: *IEEE Pervasive Computing Magazine* 1 (2002), Nr. 2

Johanson u. a. 2002b JOHANSON, Brad ; HUTCHINS, Greg ; WINOGRAD, Terry ; STONE, Maureen C.: PointRight: experience with flexible input redirection in interactive workspaces. In: *UIST*, 2002, S. 227–234

Johanson und Salgar 2002 JOHANSON, Brad ; SALGAR, Satyajeet: *The Wire Protocol for the Event Heap*. 2002. – URL http://iwork.stanford.edu/docs/eheap/wire_protocol.html. – Zugriffsdatum: 2006-07-29

KAS 2004 KONRAD-ADENAUER-STIFTUNG E.V. (Hrsg.): *Grundlagen praktischer Kommunalpolitik*. 2004. – URL http://www.kas.de/kommunal/e-learning/. – Zugriffsdatum: 2006-05-17

Miese 1980 MIESE, Jörg: *Stadt- und Regionalplanung: Ein Methodenhandbuch*. Braunschweig : Friedrich Vieweg Sohn Verlagsgesellschaft, 1980. – ISBN 3-528-08672-6

Mörike 2003 MÖRIKE, Michael: *Entwicklungsplattformen - Java versus .NET*. Heidelberg : dpunkt-Verlag, 2003. – ISBN 3-89864-202-X

Mund 2006 MUND, Horst: *Berechtigungsstrukturen in kollaborativen Umgebungen*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2006

Neumann 2006 NEUMANN, Carola: *Effizienzsteigerung von Diskussionsprozessen in einem neu gestalteten Konferenzraum*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2006

OMG 2002 OBJECT MANAGEMENT GROUP (Hrsg.): *CORBA Specification*. 2002. – URL http://www.omg.org/technology/documents/spec_catalog.htm. – Zugriffsdatum: 2006-07-30

Piening 2006 PIENING, Andreas: *RESCUE - Der Leitstand für Desaster-Szenarien*, Hochschule für Angewandte Wissenschaften Hamburg, Fachvortrag, Mai 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/piening/folien.pdf. – Zugriffsdatum: 2006-07-29

Russell und Gossweiler 2001 RUSSELL, Daniel M. ; GOSSWEILER, Rich: On the Design of Personal & Communal Large Information Scale Appliances. In: *UbiComp 2001 Proceedings*, 2001, S. 354–360

Schümmer u. a. 2000 SCHÜMMER, Jan ; SCHÜMMER, Till ; SCHUCKMANN, Christian: *COAST - Ein Anwendungsframework für synchrone Groupware*, GMD-Forschungszentrum Informationstechnik GmbH, Forschungsbericht, 2000

Song u. a. 2003 SONG, Y.J. ; TOBAGUS, W. ; LEONG, D.Y. ; JOHANSON, B. ; FOX, A.: *iSecurity: A Security Framework for Interactive Workspaces Technical Report*, Stanford University, Forschungsbericht, September 2003

Streitz u. a. 1999 STREITZ, Norbert A. ; GEISSLER, Jörg ; HOLMER, Torsten ; KONOMI, Shin'ichi ; MÜLLER-TOMFELDE, Christian ; REISCHL, Wolfgang ; REXROTH, Petra ; SEITZ, Peter ; STEINMETZ, Ralf: i-LAND: an interactive landscape for creativity and innovation. In: *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 1999, S. 120–127. – ISBN 0-201-48559-1

Streitz und Nixon 2005 STREITZ, Norbert A. ; NIXON, Paddy: The Dissapearing Computer. In: *Communicitions of the ACM* 48 (2005), Nr. 3, S. 32–35

Tandler 2003 TANDLER, Peter: The BEACH Application Model and Software Framework for Synchronous Collaboration in Ubiquitous Computing Environments. In: *Journal of Systems & Software, special issue on Ubiquitous Computing* 69 (2003), Nr. 3, S. 267–296

Tandler u. a. 2002 TANDLER, Peter ; STREITZ, Norbert A. ; PRANTE, Thorsten: Roomware-Moving Toward Ubiquitous Computers. In: *IEEE Micro* 22 (2002), Nr. 6, S. 36–47

Tanenbaum und van Steen 2002 TANENBAUM, Andrew ; STEEN, Marten van: *Distributed Systems: Principles and Paradigms*. Upper Saddle River, NJ : Prentice Hall, 2002. – ISBN 0131217860

Tolksdorf und Glaubitz 2001 TOLKSDORF, Robert ; GLAUBITZ, Dirk: XMLSpaces for Coordination in Web-Based Systems. In: *WETICE '01: Proceedings of the 10th IEEE International Workshops on Enabling Technologies*. Washington, DC, USA : IEEE Computer Society, 2001, S. 322–327. – ISBN 0-7695-1269-0

TSpaces IBM (Hrsg.): *TSpaces - Intelligent Connectionware*. – URL http://almaden.ibm.com/cs/TSpaces/index.html. – Zugriffsdatum: 2006-07-28

Verkehrskonzept Hütteldorf 2002 STADT WIEN: *Verkehrskonzept Kerngebiet Hütteldorf*. 2002. – URL http://www.wien.gv.at/stadtentwicklung/02/40/huedf.htm. – Zugriffsdatum: 2006-07-30

Weiser 1991 WEISER, Mark: The Computer for the 21st Century. In: *Scientific American* 265 (1991), Nr. 3, S. 66–75

Wells u. a. 2004 WELLS, G.C. ; CHALMERS, Alan ; CLAYTON, P.G.: Linda implementations in Java for concurrent systems. In: *Concurrency and Computation: Practice and Experience* 16 (2004), August, Nr. 10, S. 1005–1022

XPlanung 2003 BUNDESMINISTERIUM FÜR WIRTSCHAFT UND FORSCHUNG (Hrsg.): *XPlanung - Standardisierung von Geodaten*. 2003. – URL http://www.mediakomm-transfer.de/Content/de/Homepage/__GemeinsameDokum%ente__de/XPlanung__Flyer.html. – Zugriffsdatum: 2006-07-30

Zimmermann 1980 ZIMMERMANN, Hubert: OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. In: *IEEE Transactions on Communications* 28 (1980), Nr. 4, S. 425–432

A. Träger öffentlicher Belange

Abfallentsorgung	Stadt-/Kreisverwaltung
Altlastensanierung	Bezirksregierung
Bauaufsicht	Stadt-/Kreis-/(Verbands-)Gemeindeverwaltung
Bau-/Kultur-/Boden-/Erdgeschichtliche Denkmäler	Abteilungen des Landesdenkmalamtes
Baumaßnahmen des Bundes und der Länder / Staatlicher Hochbau	Staatsbauamt
Bergbau / Rohstoffe, mineralische	Bergamt
Boden-/Baugrundverhältnisse	Geologisches Landesamt
Bodenordnung (Flurbereinigung, Umlegung, Grenzregelung)	Kulturamt, Stadtvermessungsamt, Katasteramt
Brandschutzwesen	Feuerwehr, Kreisdienststelle für Brand- und Katastrophenschutz
Fernmeldewesen	Regulierungsbehörde für Telekommunikation und Post
Forstwirtschaft	Forstamt
Gesundheitswesen (Siedlungs-/Umwelthygiene)	Kreisgesundheitsamt
Grundbesitz, öffentlicher	Bundesvermögensamt, Forstamt, Straßen-/Verkehrsamt
Immissionsschutz (ohne Verkehrsanlagen)	Gewerbeaufsichtsamt
Industrie, Gewerbe, Handwerk	Industrie-/Handels-/Handwerkskammern
Jugendförderung/-pflege/-fürsorge/ -hilfe/-bildung/-erholung	Jugendamt
Kirchen, Religionsgemeinschaften ö.R.	Pfarrämter, örtliche Gemeinden
Klimaschutz	Wetterdienst
Landschaftspflege, Naturschutz-/Artenschutzbereiche	Stadt-/Kreisverwaltung, Bezirksregierung
Landwirtschaft	Stadt-/Kreisverwaltung, Landwirtschaftskammer
Landwirtschaftliches Siedlungswesen	Kulturamt
Schulwesen, Hochschulen	Bezirksregierung, Schulträger, Universitäten, Fachhochschulen
Verkehr (auch Verkehrsemissionen/-immissionen)	Autobahnamt, Straßen-/Verkehrsamt, Straßenprojektamt, Bezirksregierung, Kreis-/Stadt-/(Verbands-)Gemeindeverwaltung Deutsche Bahn AG Netz, Eisenbahn-Bundesamt, SPNV-Zweckverband Öffentl. Nahverkehrsbetriebe/-unternehmen Wasser- und Schiffahrtsamt Deutsche Flugsicherung GmbH, Flughafengesellschaft, Landesaufsicht, Bezirksregierung
Ver-/Entsorgung (Energie, Rohr-(fern-)leitungen, Wasserversorgung, Abwasserbeseitigung)	Ver-/Entsorgungsträger, Gewerbeaufsichtsamt, Bergamt, Amt für Wasser- und Abfallwirtschaft
Verteidigung	Standortverwaltung, Wehrbereichsverwaltung
Wasserwirtschaft (Gewässerbenutzung, Gestaltung der Gewässer, Grundwasser-verhältnisse, Hochwasserabfluss)	Amt für Wasser- und Abfallwirtschaft, Kreis-/Stadt-/(Verbands-)Gemeindeverwaltung

Abbildung A.1.: Träger öffentlicher Belange [KAS 2004]

B. Definition der Tupelfelder

Die folgenden Tabellen enthalten die Tupelfelder der entworfenen Nachrichten. In der linken Spalte ist der Feldname angegeben, in der rechten der Feldwert. Werte in eckigen Klammern sind dynamisch. Dies sind entweder Benutzereingaben oder vom System gesetzte Werte. In Kursivschrift angegebene Felder sind optional.

B.1. Systemnachrichten

Login

Eventtype	Systemnachricht
Subtype	Login
Username	<Benutzername>
Password	<Passwort>
SourceIP	<IP-Addresse des Clients>

Tabelle B.1.: Tupeldefinition der Systemnachricht Login

Login Server-Antwort

Eventtype	Systemnachricht_response
Subtype	Login
Request_SessionID	<SessionID des Requests>
Request_SeqNr	<SeqNr des Requests>
ReturnValue	<KoraSessionID>
Exceptiontype	*<Typ der Exception>*

Tabelle B.2.: Tupeldefinition der Systemnachricht Login Server-Antwort

Logout

Eventtype	Systemnachricht
Subtype	Logout
KoraSessionID	<KoraSessionID>

Tabelle B.3.: Tupeldefinition der Systemnachricht Logout

GetFileList

Eventtype	Systemnachricht
Subtype	GetFileList
KoraSessionID	<KoraSessionID>

Tabelle B.4.: Tupeldefinition der Systemnachricht GetFileList

GetFileList Server-Antwort

Eventtype	Systemnachricht_response
Subtype	GetFileList
Request_SessionID	<SessionID des Requests>
Request_SeqNr	<SeqNr des Requests>
ReturnValue	<Liste mit Dateinamen>
Exceptiontype	*<Typ der Exception>*

Tabelle B.5.: Tupeldefinition der Systemnachricht GetFileList Server-Antwort

GetFile

Eventtype	Systemnachricht
Subtype	GetFile
KoraSessionID	<KoraSessionID>
Path	<Dateipfad>

Tabelle B.6.: Tupeldefinition der Systemnachricht GetFile

GetFile Server-Antwort

Eventtype	Systemnachricht_response
Subtype	GetFile
Request_SessionID	<SessionID des Requests>
Request_SeqNr	<SeqNr des Requests>
ReturnValue	<Portnummer für Dateitransfer>
Exceptiontype	*<Typ der Exception>*

Tabelle B.7.: Tupeldefinition der Systemnachricht GetFile Server-Antwort

SaveFile

Eventtype	Systemnachricht
Subtype	SaveFile
KoraSessionID	<KoraSessionID>
Path	<Dateipfad>

Tabelle B.8.: Tupeldefinition der Systemnachricht SaveFile

SaveFile Server-Antwort

Eventtype	Systemnachricht_response
Subtype	SaveFile
Request_SessionID	<SessionID des Requests>
Request_SeqNr	<SeqNr des Requests>
ReturnValue	<Portnummer für Dateitransfer>
Exceptiontype	*<Typ der Exception>*

Tabelle B.9.: Tupeldefinition der Systemnachricht SaveFile Server-Antwort

GetMetadaten

Eventtype	Systemnachricht
Subtype	GetMetadaten
KoraSessionID	<KoraSessionID>
Path	<Dateipfad>

Tabelle B.10.: Tupeldefinition der Systemnachricht GetMetadaten

GetMetadaten Server-Antwort

Eventtype	Systemnachricht_response
Subtype	GetMetadaten
Request_SessionID	<SessionID des Requests>
Request_SeqNr	<SeqNr des Requests>
ReturnValue	<Metadatenobjekt>
Exceptiontype	*<Typ der Exception>*

Tabelle B.11.: Tupeldefinition der Systemnachricht GetMetadaten Server-Antwort

SaveMetadaten

Eventtype	Systemnachricht
Subtype	SaveMetadaten
KoraSessionID	<KoraSessionID>
Path	<Dateipfad>
Metadaten	<Metadatenobjekt>

Tabelle B.12.: Tupeldefinition der Systemnachricht SaveMetadaten

SaveMetadaten Server-Antwort

Eventtype	Systemnachricht_response
Subtype	SaveMetadaten
Request_SessionID	<SessionID des Requests>
Request_SeqNr	<SeqNr des Requests>
Exceptiontype	*<Typ der Exception>*

Tabelle B.13.: Tupeldefinition der Systemnachricht SaveMetadaten Server-Antwort

B.2. Interaktionsnachrichten

ShowDocument

Eventtype	Interaktionsnachricht
Subtype	ShowDocument
KoraSessionID	<KoraSessionID>
Path	<Dateipfad>
Target	<SessionID des Nachrichtenempfängers>

Tabelle B.14.: Tupeldefinition der Interaktionsnachricht ShowDocument

ShowDocument Server-Antwort

Eventtype	Interaktionsnachricht_response
Subtype	ShowDocument
Request_SessionID	<SessionID des Requests>
Request_SeqNr	<SeqNr des Requests>
Exceptiontype	*<Typ der Exception>*

Tabelle B.15.: Tupeldefinition der Interaktionsnachricht ShowDocument Server-Antwort

B.3. Informationsnachrichten

UserStatus

Eventtype	Informationsnachricht
Subtype	UserStatus
KoraSessionID	<KoraSessionID>
Value	<KoraTeilnehmer-Objekt>

Tabelle B.16.: Tupeldefinition der Informationsnachricht UserStatus

C. Hinweis zum Prototypen

Im Rahmen dieser Arbeit wurde ein Prototyp erstellt, welcher die hier beschriebenen Konzepte der Middleware umsetzt. Der Quelltext und die zur Ausführung notwendigen Bibliotheken werden auf Anfrage herausgegeben. Zur Ausgabe steht der Author der Arbeit, Lars Burfeindt (L.Burfeindt@gmx.de) sowie der Erstgutachter, Prof. Dr. rer. nat. Kai von Luck (luck@informatik.haw-hamburg.de), zur Verfügung.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 17. August 2006	
Ort, Datum	Unterschrift