

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	SMARTBoard	1
1.3	Ziel	1
1.4	Übersicht	2
1.4.1	Funktionalitätsbeschreibung	2
1.4.2	Anforderungsanalyse	2
1.4.3	Recherche	2
1.4.4	Entwurf	2
1.4.5	Implementierung	2
1.4.6	Test und Bewertung	3
1.4.7	Zusammenfassung und Ausblick	3
1.5	Quellenangabe	4
2	Funktionalitätsbeschreibung	5
2.1	Grundfunktionen	5
2.2	Komponenten	5
2.2.1	Eingabegeräte	5
2.2.2	Ausgabegeräte	6
2.2.3	Serversoftware	6
2.2.4	Der Client	7
2.2.5	Übersicht	8
3	Anforderungsanalyse	9
3.1	Spezielle Anforderungen	9
3.1.1	Anmeldungsunterstützung	9
3.1.2	Teilnehmerverwaltung	9
3.1.3	Dateitransfer	9
3.1.4	Applikationssteuerung	10
3.1.5	Anmerkungen	10
3.1.6	Zeigemöglichkeit	10

3.1.7	Floor Control	10
3.1.8	Mehrpunktkonferenzen	10
3.2	Allgemeine Anforderungen	11
3.2.1	Offenheit	11
3.2.2	Plattformen	11
3.2.3	Verfügbarkeit	11
3.2.4	Konsistenz	11
3.2.5	Stabilität	11
3.2.6	Sicherheit	11
3.2.7	Firewalltauglichkeit	12
4	Eignungsanalyse von Komplettsystemen	13
4.1	Auswahl und Testablauf	13
4.2	Eignungsanalyse NetMeeting	13
4.2.1	Spezielle Anforderungen	13
4.2.2	Allgemeine Anforderungen	19
4.2.3	Fazit	21
4.3	Eignungsanalyse MBone-Tools	22
4.3.1	Spezielle Anforderungen	22
4.3.2	Allgemeine Anforderungen	23
4.3.3	Fazit	24
4.4	Eignungsanalyse JETS	25
4.4.1	Spezielle Anforderungen	26
4.4.2	Allgemeine Anforderungen	27
4.4.3	Fazit	29
4.5	Eignungsanalyse JaTeK	30
4.5.1	Einleitung	30
4.5.2	Bewertung	31
4.6	Zusammenfassung	32
4.6.1	Untersuchte Systeme	32
4.6.2	Kriterien	32
4.6.3	Ergebnisse	33
4.6.4	Fazit	35
4.7	Quellenangabe	36

5	Eignungsanalyse von Einzelkomponenten	37
5.1	Präsentations-Fernsteuerung	37
5.1.1	Aufbau	37
5.1.2	Bewertung	38
5.2	Java Shared Data Toolkit	38
5.2.1	Systembeschreibung	38
5.2.2	Bewertung anhand spezieller Anforderungen	40
5.2.3	Bewertung anhand allgemeiner Anforderungen	41
5.2.4	Fazit	42
5.3	Designempfehlung	42
5.4	Quellenangabe	44
6	Entwurf einer Generische Floor Control	45
6.1	Definition "Floor Control"	45
6.2	Grundbegriffe	45
6.2.1	Dienst	45
6.2.2	Teilnehmer	46
6.2.3	Recht	46
6.2.4	Mechanismus	46
6.2.5	Rollen	47
6.3	Abstimmungen	47
6.3.1	Beispielhafte Szenarien	48
6.3.2	Stimmen und Stimmgewichtung	49
6.3.3	Gegenstand von Abstimmungen	50
6.3.4	Optionen bei einer Abstimmung	50
6.3.5	Abschätzung der Kompliziertheit des Verfahrens	52
6.4	Quellenangabe	54
7	Implementierung der Floor Control	55
7.1	Kommunikationssystem	55
7.2	Benutzerdatenbank	55
7.3	Programmierschnittstellen	56
7.3.1	Server	56
7.3.2	Client	57
7.4	Systemstruktur	57

7.4.1	Server	58
7.4.2	Client	59
7.5	Quellenangabe	60
8	Test und Bewertung	61
8.1	Test	61
8.1.1	Chat-Server	61
8.1.2	Chat-Client	62
8.1.3	Testergebnisse	63
8.2	Bewertung	63
9	Zusammenfassung und Ausblick	64
9.1	Zusammenfassung	64
9.2	Ausblick	64
9.3	Quellenangabe	65
10	Anhang	66
10.1	Projektplan	66
10.1.1	Einleitung	66
10.1.2	Beschreibung der Arbeitspakete	67
10.1.3	Zeitplan	69
10.1.4	Dokumente und Meilensteine	70
10.2	NetMeeting	71
10.3	CommServer-API	71
10.4	CommClient-API	73
10.5	CommChannel-API	74
10.6	FloorServer-API	76
10.7	FloorClient-API	78

Abbildungsverzeichnis

Abbildung 2-1: Übersicht über die organisatorische Systemstruktur	8
Abbildung 4-1: Konferenzanmeldung per Verzeichnisdienst	14
Abbildung 4-2: Kopier-Artefakte bei der Zeigerdarstellung	17
Abbildung 4-3: Deaktivierte Werkzeugleiste	17
Abbildung 4-4: Das Fenster zur Vergabe des Rederechts (Floor Control)	27
Abbildung 4-5: Das JaWoS-Whiteboard	31
Abbildung 6-1: Rederechtvergabe durch Vorsitzenden	48
Abbildung 6-2: Rederechtvergabe nach dem first-come-first-served-Prinzip .	49
Abbildung 6-3: Beispielhafte Prioritätsreihenfolge	49
Abbildung 6-4: Prinzip der Stimmübergabe	50
Abbildung 6-5: Beispiel für einen Abstimmungsantrag	51
Abbildung 7-1: Datenfluß innerhalb des Floor-Control-Servers	58
Abbildung 7-2: Datenfluß innerhalb des Floor-Control-Clients	59
Abbildung 8-1: Benutzungsoberfläche des Chat-Servers	61
Abbildung 8-2: Auszug aus dem Programmcode des Chat-Servers	61
Abbildung 8-3: Benutzungsoberfläche des Chat-Clients	62
Abbildung 8-4: Auszug aus dem Programmcode des Chat-Clients	62
Abbildung 10-1: Abhängigkeiten der Arbeitspakete	68
Abbildung 10-2: Liste der von NetMeeting benutzten Ports	71

1 Einleitung

1.1 Motivation

Vorlesungen für bis zu 20 Studierende werden in zunehmendem Maße mit Hilfe von SMARTBoards gehalten, an denen computergestützte Präsentationen sowohl angezeigt als auch durch Texte und Zeichnungen ergänzt werden können. Für mehr Interaktivität wäre es wünschenswert, wenn auch die Studierenden direkt von ihrem Platz aus die Möglichkeit hätten, bei entsprechender Berechtigung zeigend oder sogar modifizierend auf die SMARTBoard-Anzeige einzuwirken.

1.2 SMARTBoard

Bei einem SMARTBoard handelt es sich um eine "elektronische Tafel" der Firma SMART Technologies Inc. [SMART]. Eine derartige Tafel besteht aus einer weißen Projektionsfläche, auf der per Projektor der Bildschirminhalt eines Präsentationscomputers abgebildet wird. Zudem ist diese Fläche berührungssensitiv, so daß eine Steuerung des Präsentationscomputers direkt an der Tafel erfolgen kann.

1.3 Ziel

Die Arbeit zielt darauf ab, die Interaktivität zwischen Lehrenden und Lernenden im Rahmen von Vorlesungen zu fördern, indem sie die kontrollierte Fernsteuerung von Bildschirmpräsentationen von mobilen Endgeräten aus unterstützt. Dazu wird insgesamt ein System mit folgender Funktionalität benötigt:

- Fernsteuerung von Bildschirmpräsentationen (z.B. PowerPoint-Folien)
- Anschluß mehrerer Endgeräte (bis zu 20)
- Annotation des Bildschirminhaltes von allen Endgeräten aus
- Steuerung eines Zeigersymbols
- Protokollierung der Eingaben um ein erneutes Anzeigen zu ermöglichen
- Koordination des Zugriffs

Der eigentliche Schwerpunkt dieser Arbeit besteht aus dem Entwurf und der Implementierung einer "Floor Control", d.h. einer Komponente, die den Zugriff auf die Fernsteuerungsfunktionen des Gesamtsystems regelt. Dabei sollen dem Vortragenden besondere Rechte eingeräumt werden, um den geregelten Fortgang von Veranstaltungen (z.B. Vorlesungen) gewährleisten zu können.

Es ist denkbar, diese Komponente in die Fernsteuerung mit PowerPoint einzubauen, wie sie im Rahmen einer bereits abgeschlossenen Studienarbeit [NGUY 00] für ein einzelnes mobiles Endgerät implementiert wurde. Im Hinblick auf ein hohes Maß an Stabilität und breiter Verfügbarkeit sind jedoch auch andere Systeme dahingehend zu analysieren, ob sie als Basis für das gewünschte Zielsystem dienen können.

1.4 Übersicht

Im folgenden wird die Vorgehensweise beschrieben, anhand derer die in der Zielbeschreibung aufgeführten Aufgaben bearbeitet wurden.

1.4.1 Funktionalitätsbeschreibung

Zu Beginn wurde die gewünschte Funktionalität des Zielsystemsgenauer definiert, um eine objektive Grundlage für die Bewertung existierender Systeme sowie von eigenen Neuentwicklungen bereitzustellen.

1.4.2 Anforderungsanalyse

Aus der Funktionalitätsbeschreibung wurden Kriterien zur Bewertung der Eignung von bestehenden Systemen abgeleitet, die in der Anforderungsanalyse zusammengefaßt wurden.

1.4.3 Recherche

Während der Recherche-Phase wurden verschiedene Softwaresysteme, die ein Whiteboard (eine Software-Komponente mit der Funktionalität einer verteilten, elektronischen Tafel) realisieren, auf ihre Eignung für die Verwendung im Zielsystem hin überprüft.

Als sich herausstellte, daß keines der untersuchten Komplettsysteme alle notwendigen Anforderungen erfüllt, wurde in einer zweiten Phase nach einzelnen Komponenten gesucht, die zur Realisierung des Systems verwendet werden könnten.

1.4.4 Entwurf

Da die Ergebnisse der Recherche negativ ausfielen und keines der untersuchten Systeme alle Grundanforderungen erfüllte, wurde beschlossen, ein entsprechendes System neu zu entwerfen. Hierzu wurde ein Entwurf für die Floor-Control-Komponente erarbeitet.

1.4.5 Implementierung

Im Anschluß an den Entwurf der Floor-Control-Komponente erfolgte deren Implementierung in der Programmiersprache JAVA.

1.4.6 Test und Bewertung

Während der Bewertungsphase wurde die Grundfunktionalität der Floor Control anhand eines einfachen Chat-Programms mit dezentraler Rederechtvergabe demonstriert und getestet.

1.4.7 Zusammenfassung und Ausblick

Die vorliegende Ausarbeitung schließt mit der Zusammenfassung der Ergebnisse dieser Studienarbeit sowie einem Ausblick auf das weitere Vorgehen im Hinblick auf das in der Einleitung beschriebene Gesamtsystem.

1.5 Quellenangabe

SMART

SMART Technologies Inc.
Webseite der Firma "SMART Technologies Inc."
<http://www.smarttech.com>

NGUY 00

Nguyen-Salamanis, Khan-Loan
Adhoc-Verwendung einer elektronischen Tafel unter Verwendung der Jini-Technologie
Universität Stuttgart, Fakultät Informatik, Studienarbeit Nr. 1778
ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/STUD-1778/

2 Funktionalitätsbeschreibung

Die im Projektplan umrissene Funktionalität des Gesamtsystems (siehe Abschnitt 10.1, "Projektplan") wird in diesem Kapitel weiter präzisiert und ausformuliert, um eine Grundlage sowohl für die Bewertung bestehender Systeme, als auch für den Entwurf von Neuentwicklungen bereitzustellen.

Hinweis: Da sich die Aufgabenstellung im Verlauf der Arbeit zugunsten einer allgemeineren Betrachtung geändert hat, wurden die im ursprünglichen Projektplan aufgeführten Arbeitspakete, die sich speziell auf die Erweiterung eines bereits bestehenden Systems beziehen, nicht in der geplanten Form umgesetzt.

2.1 Grundfunktionen

Im Projektplan wird der geforderte Funktionsumfang wie folgt beschrieben:

- Automatische Übertragung von Präsentationsdateien (PowerPoint) auf einen Präsentationsserver
- Steuerung der Präsentation von mobilen Endgeräten aus
- Austausch von Anmerkungen zwischen Smartboard und mobilen Endgeräten
- Zeigemöglichkeit von mobilen Endgeräten aus
- Koordination des Zugriffes mehrerer Endgeräte per Floor Control, steuerbar durch Dozent/in

2.2 Komponenten

Es lassen sich die folgenden organisatorischen Komponenten unabhängig von der gewählten Implementierung abgrenzen:

2.2.1 Eingabegeräte

Das Smartboard dient in dieser Betrachtung als reines Eingabegerät. Es handelt sich dabei um ein Positioneingabegerät, wobei zusätzlich Informationen über das gerade verwendete Werkzeug (Stift, Zeiger oder Schwamm) mitgeliefert werden.

Es ist denkbar, zu einem späteren Zeitpunkt weitere Positionierungsgeräte (z.B. ein Grafiktablett, etc.) in das System einzubinden. Hierbei ist die Offenheit des Systems entscheidend für den dafür notwendigen Entwicklungsaufwand.

2.2.2 Ausgabegeräte

Der an den Präsentationsserver und an das SMARTboard angeschlossene Projektor dient als Ausgabegerät. Auf ihm wird der Bildschirm des Servers ausgegeben.

Auf der Seite der Clients ist die Ausgabe in graphischer Form als Teil der Benutzungsoberfläche integriert.

2.2.3 Serversoftware

Der Server dient als zentrale Stelle im System. Er stellt folgende Dienste zur Verfügung:

Anmeldungsunterstützung

Alle teilnehmenden Personen müssen die Möglichkeit haben, mit dem Server in Kontakt zu treten, bzw. diesen Kontakt wieder zu beenden. Dem System muß die (innerhalb des Vortrags eindeutige) Identität sowie die Rolle jeder Person bekannt sein.

Es wäre wünschenswert, daß die Kontaktaufnahme mit dem Server schnell und unkompliziert erfolgt, d.h. die dazu notwendige Netzwerk-Adresse sollte z.B. anhand der Kenntnisse über den Ort des Vortrags ermittelbar sein.

Denkbar wäre auch eine Funktionalität, mit der - manuell oder automatisch - der Sitzplatz eines Teilnehmers bestimmt werden kann, um die Zuordnung von Benutzernamen und Personen im Raum für den Dozenten zu erleichtern.

Übertragung von Präsentationsdaten

Eine wichtige Anforderung ist die Möglichkeit, Präsentationsdateien von einem teilnehmenden Endgerät auf den Präsentationsserver zu übertragen. Denkbar wäre auch, daß die Präsentationsdateien während des Vortrags auf die anderen teilnehmenden Endgeräte übertragen werden können, wobei diese Möglichkeit steuerbar sein sollte, um Bedenken im Hinblick auf den Schutz des Urheberrechts zu vermeiden.

Zu beachten ist zudem, daß bei der Verwendung von Präsentationen in einem Format, welches die Ausführung von Programmanweisungen ermöglicht (z.B. in Form von "Visual Basic for Applications" bei Microsoft PowerPoint Präsentationen), die Gefahr der Verbreitung von Computer-Viren besteht.

Diese Gefahr läßt sich im Fall der Übertragung von Dateien auf den Präsentationsrechner dadurch umgehen, daß die Verwendung dieser Anweisungen deaktiviert wird. Bei der Übertragung von Präsentationen auf ein mobiles Endgerät ist ein ausreichender Schutz nur durch die Verwendung von Anti-Viren-Programmen mit einer aktuellen Viren-Datenbank möglich, was jedoch bei der Wartung einen nicht zu vernachlässigen Aufwand verursacht.

Präsentationssteuerung

Es muß die Möglichkeit bestehen, Präsentationen von einem Endgerät oder vom SMARTBoard aus zu steuern.

Annotation der Anzeige

- Sowohl vom SMARTboard, als auch von einem beliebigen Endgerät aus muß es möglich sein, Anmerkungen in Form von Freihand-Zeichnungen und einfachen Textstrings *über* die dargestellte Präsentation zu zeichnen, bzw. diese wieder zu entfernen ("Schwamm-Funktion").
- Bereits erstellte Anmerkungen sollen gespeichert werden und bei Wiederanzeige einer bestimmten Präsentationsseite ebenfalls wieder angezeigt werden, wobei die Möglichkeit des Ausblendens gegeben sein sollte.
- Die während eines Vortrags erstellten Anmerkungen sollen auf die Endgeräte rückübertragen werden können.
- Zusätzlich zur Möglichkeit, öffentliche, für alle Teilnehmer sichtbare Anmerkungen vorzunehmen, wäre es denkbar, rein private Annotationen anzubringen, wobei das entsprechende Dokument in diesem Fall in irgendeiner Form auf dem Endgerät vorhanden sein muß.

Zeigerfunktionalität

Teilnehmern sollte ein fernsteuerbares Zeigersymbol zur Verfügung stehen, durch welches das Verweisen auf bestimmte Teile der Bildschirmdarstellung erleichtert wird. Zur Verbesserung der Übersichtlichkeit bei einer derartigen Aktion, sollte eine Zuordnung von Zeiger zu steuerndem Teilnehmer möglich sein.

Verwaltung des Eingaberechts

Die vortragende Person soll jederzeit die Möglichkeit haben, den Studierenden die einzelnen Rechte der Präsentationssteuerung und -annotation zu gewähren und wieder zu entziehen.

2.2.4 Der Client

Die Client-Komponente ermöglicht den Zugriff auf die Dienste des Servers, wobei zwischen den beiden Rollen Vortragender und Zuhörer unterschieden wird. Es wäre wünschenswert, die Client-Software zu Beginn eines Vortrags vom Server herunterzuladen, um Komplikationen bei Versionsunterschieden der - teilweise hardwareabhängigen - Serverkomponente zu vermeiden. Dadurch würde gewährleistet werden, daß trotz unterschiedlicher Hard- und Software je Vorlesungsraum immer die richtige Version geladen wird.

2.2.5 Übersicht

Das Zusammenwirken der einzelnen Komponenten ist in Abbildung 2-1 graphisch dargestellt.

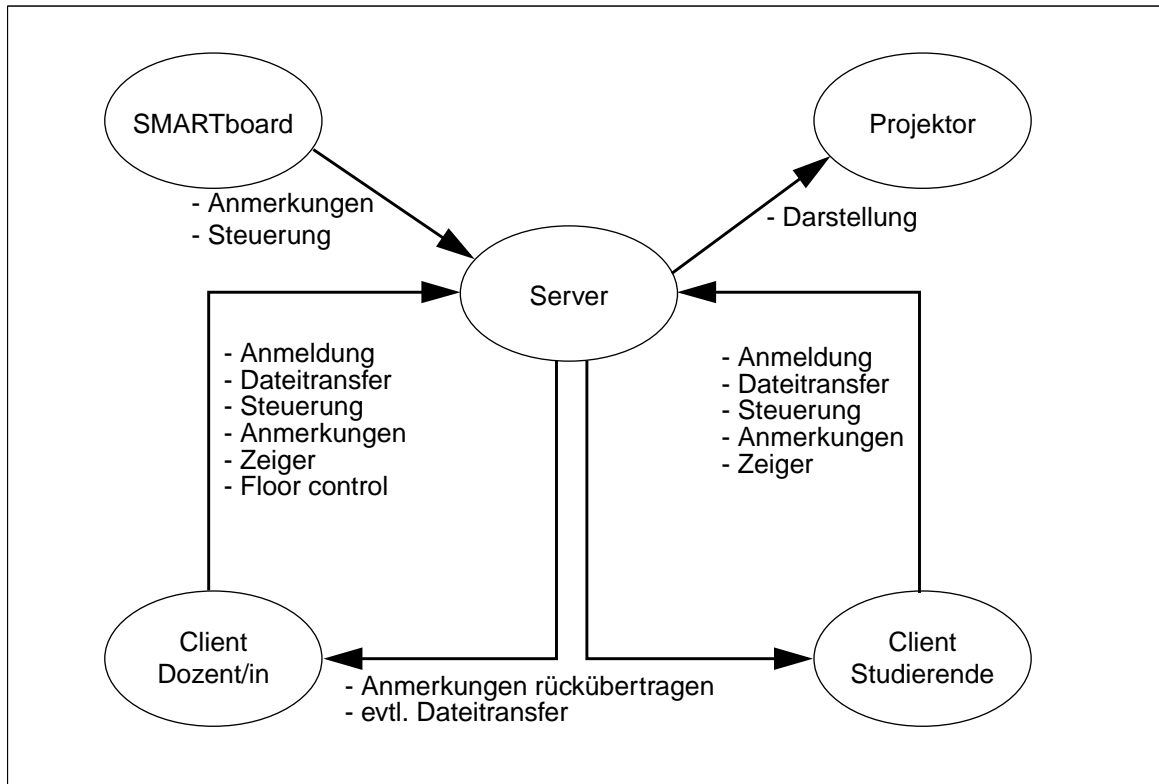


Abbildung 2-1: Übersicht über die organisatorische Systemstruktur

3 Anforderungsanalyse

Die im vorhergehenden Kapitel beschriebene Funktionalität entspricht zu großen Teilen der eines Konferenzsystems, wobei die Verwendung der Audio/Video-Komponente solcher Systeme in diesem Anwendungsfall nicht sinnvoll ist. In diesem Kapitel wird untersucht, welche Anforderungen ein Konferenzsystem erfüllen muß, um im Rahmen dieser Studienarbeit verwendet werden zu können.

Die Grundlagen für die Erarbeitung der Anforderungen sind dabei sowohl die ausformulierte Funktionalitätsbeschreibung aus Kapitel 2, als auch allgemeine Überlegungen, die sich aus dem Anwendungsszenario einer Vorlesung ergeben.

3.1 Spezielle Anforderungen

3.1.1 Anmeldungsunterstützung

Sowohl das Auffinden, als auch das Anmelden an eine existierende Sitzung des Konferenzsystems muß unterstützt werden. Dabei sollte insbesondere auch das verspätete Anmelden sowie das verfrühte Verlassen des Vortrags möglich sein. Zusätzlich ist es von Vorteil, wenn zum Auffinden des für einen Vortrag zuständigen Servers keine implementierungsinternen Informationen wie z.B. IP-Adressen notwendig sind, sondern das reine Anwenderwissen über Vortragsort oder -inhalt ausreicht.

3.1.2 Teilnehmerverwaltung

Es muß die Möglichkeit bestehen, die teilnehmenden Personen innerhalb des Systemablaufs eindeutig zu identifizieren und ihnen eine Rolle (Dozent/Zuhörer) zuzuweisen. Ohne eine derartige Möglichkeit ist die Vergabe der Zugriffsrechte und damit der geordnete Ablauf eines Vortrags nicht gewährleistet.

Desweiteren sollte für Teilnehmer ersichtlich sein, welche anderen Personen an einer Sitzung teilnehmen ("Team Awareness"), wobei eventuell auch die Möglichkeit gegeben sein sollte, Sitzungen anonym zu besuchen.

3.1.3 Dateitransfer

Für eine unkomplizierte Handhabung ist eine integrierte Dateiübertragungsmöglichkeit notwendig. Diese wird zum Beispiel benutzt, um die Präsentationsdateien bei Beginn des Vortrags auf den Server und eventuell von dort auf die Endgeräte zu übertragen. Diese letztgenannte Rückübertragung muß dabei steuerbar sein, um Probleme im Zusammenhang mit den Urheberrechten der Präsentationsdateien zu vermeiden.

3.1.4 Applikationssteuerung

Eine integrierte Schnittstelle zur Fernsteuerung der auf dem Präsentationsserver ablaufenden Software ist erforderlich.

3.1.5 Anmerkungen

Es ist von besonderer Wichtigkeit, daß das Konferenzsystem eine *“Anmerkungsfunktionalität”* unterstützt, beziehungsweise daß eine vorhandene *Whiteboard*-Funktion entsprechend erweitert werden kann, um die folgenden Aktionen abzudecken:

- Anmerkungen in Form von Freihandzeichnungen und einfachen Textstrings auf dem momentan angezeigten Bildschirminhalt anbringen
- Verwalten und wiederholtes Anzeigen von Anmerkungen zu früheren Inhalten
- Ausblenden von Anmerkungen
- Export der Anmerkungen in ein gängiges Format, um ein nachträgliches Verarbeiten zu ermöglichen
- Möglichkeit der öffentlichen **und** privaten Annotation

3.1.6 Zeigemöglichkeit

Das System sollte die Fernsteuerung eines Zeigers auf dem Präsentationsbildschirm ermöglichen, bzw. dessen Integration unterstützen.

3.1.7 Floor Control

Die Rechte an der Benutzung der erwähnten Dienste müssen zentral und rollenabhängig gesteuert werden können, um die Umsetzung der *Floor Control* zu ermöglichen.

3.1.8 Mehrpunktkonferenzen

Das System muß Konferenzen mit mindestens 20 teilnehmenden Endgeräten unterstützen, um im Rahmen von Vorlesungen sinnvoll einsetzbar zu sein.

3.2 Allgemeine Anforderungen

3.2.1 Offenheit

Die Integration von neuen Tools, sowie die Erweiterung bestehender Funktionalitäten muß durch das Konferenzsystem unterstützt werden. Insbesondere muß der Zugriff auf systeminterne Informationen, wie beispielsweise die Teilnehmerverwaltung, für Erweiterungen zugänglich sein.

Unter diesem Gesichtspunkt ist ebenfalls zu untersuchen, ob bei der Definition von Systemschnittstellen auf Standards zurückgegriffen wird.

3.2.2 Plattformen

Die Software muß auf den unterschiedlichen beteiligten Plattformen verfügbar sein. Dies umfaßt sowohl die mobilen Endgeräte, als auch den Präsentationsserver, der mit dem Smartboard verbunden ist.

3.2.3 Verfügbarkeit

Die in den Vorträgen verwendete Konferenzsoftware muß für alle Teilnehmer verfügbar sein. In diesem Zusammenhang ist das Vorhandensein und der Aufwand für den Erwerb von Softwarelizenzen ein zentrales Kriterium. Zudem ist auch der Aufwand zur Vermeidung von Versionsinkompatibilitäten zwischen Client- und Server-Software zu bewerten.

3.2.4 Konsistenz

Die Benutzungsschnittstelle des Konferenzsystems sollte im Erscheinungsbild und in der Bedienung konsistent sein, um die Akzeptanz des Endsystems zu verbessern. Neu zu integrierende Komponenten müssen unter diesem Gesichtspunkt ebenfalls nahtlos integriert werden können.

3.2.5 Stabilität

Die Stabilität des Konferenzsystems ist für den Erfolg eines rechnerunterstützten Vortrags, und somit auch für die Akzeptanz durch die Benutzer, von entscheidender Bedeutung. Dabei spielt die Robustheit gegenüber zufälligen, aber auch mutwilligen Fehlbedienungen eine Rolle.

3.2.6 Sicherheit

Bei diesem Punkt ist zu untersuchen, inwiefern die Authentifizierung der Teilnehmer und der Dozenten manipulierbar ist und als wie sicher sich die Basis für die Implementierung der *Flour Control* erweist.

3.2.7 Firewalltauglichkeit

Es soll außerdem untersucht werden, ob es beim Betrieb des Konferenzsystems in einem durch eine Firewall abgeschotteten Netzwerk zu Problemen kommen kann.

4 Eignungsanalyse von Komplettsystemen

4.1 Auswahl und Testablauf

Im diesem Abschnitt werden die Ergebnisse der Untersuchung von vier verschiedenen Konferenzsystemen anhand der Kriterien der Anforderungsanalyse (siehe Kapitel 3) dokumentiert. Es wurden die folgenden Systeme untersucht:

1. Microsoft NetMeeting
2. MBone-Tools
3. JETS
4. JaTek

Die Auswahl der untersuchten Systeme geschah vornehmlich nach den folgenden Kriterien:

- Verfügbarkeit. Es wurden keine kommerziellen Systeme getestet, da deren Verwendung im Widerspruch zur Forderung nach einfacher Verfügbarkeit steht (siehe Abschnitt 3.2.3, "Verfügbarkeit").
- Vorhandensein einer elektronischen Tafel
- Verfügbarkeit auf Unix- oder MS Windows Plattformen. Diese Einschränkung ergab sich aus der Hardwarekonfiguration der Testumgebung.

Im Laufe der Tests wurden diese Anwendungen installiert und ausgeführt, wobei jeweils nur die Komponenten benutzt wurden, die eine elektronische Tafel realisieren. Insbesondere wurden die Audio/Video-Systeme nicht verwendet, da diese für den primären Anwendungsfall der Präsentation innerhalb einer Vorlesung nicht obligatorisch sind. Die relevanten Ergebnisse der Tests wurden protokolliert und in diesem Kapitel zusammengefasst.

4.2 Eignungsanalyse NetMeeting

Bei diesem Test wurde die Konferenzsoftware "*Microsoft NetMeeting V3.01*" unter dem Betriebssystem MS Windows NT 4.0 untersucht.

4.2.1 Spezielle Anforderungen

Anmeldungsunterstützung

Die Lokalisierung des zuständigen Servers erfolgt entweder direkt über die Angabe des Rechnernamens oder über einen Verzeichnisserver (*Microsoft Internet Locator Service, ILS*, siehe Abbildung 4-1).

Verbindungswünsche von Clients können entweder explizit für jeden Benutzer gewährt werden (PopUp-Meldung auf Server-Rechner), oder alle Anfragen werden automatisch zugelassen, wobei optional ein Konferenzpasswort angegeben werden kann.

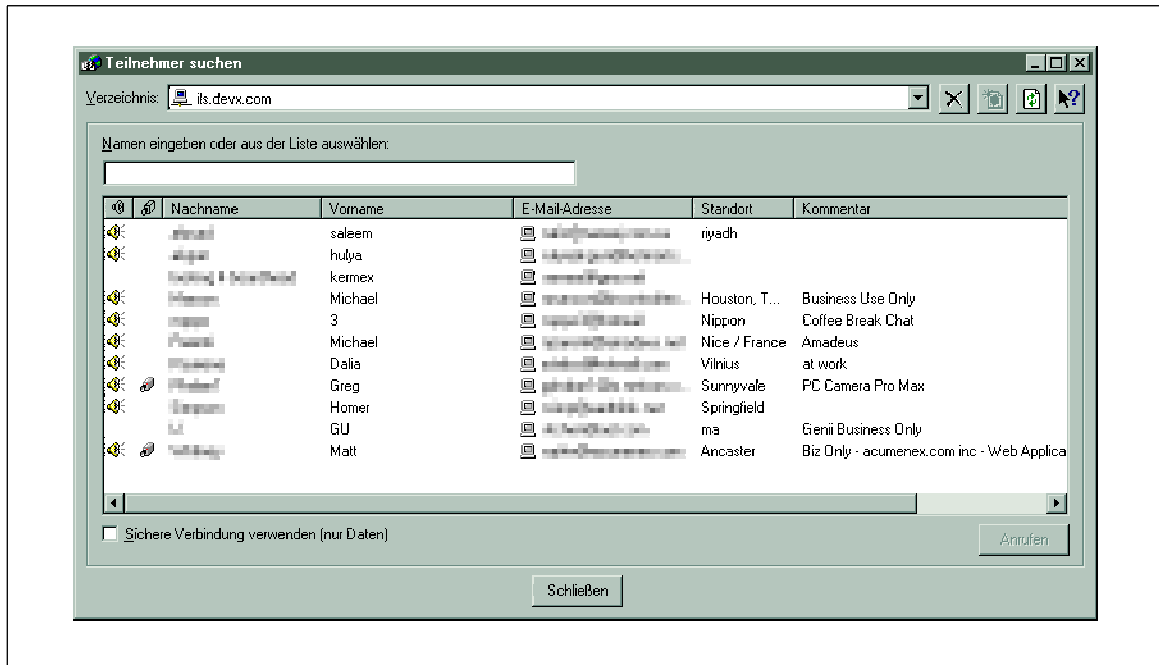


Abbildung 4-1: Konferenzanmeldung per Verzeichnisdienst

Das Verlassen sowie das Anmelden an eine Konferenz ist für die Clients jederzeit möglich. Im Fall von erweiterten, nicht sternförmigen Konferenzstrukturen entsteht ein Problem, wenn ein Client, welcher weitere Clients mit der Konferenz verbindet, die Sitzung frühzeitig verläßt. In einem solchen Fall werden alle untergeordneten Clients ebenfalls von der Konferenz getrennt.

Vorteile:

- Anmelden an Konferenz kann mit Hilfe eines eigenen Verzeichnisseservers unabhängig von Rechnernamen erfolgen
- ILS basiert auf LDAP, dadurch Standard-Konformität

Nachteile:

- Keine vollständige LDAP-Kompatibilität [MSRES 99]

Teilnehmerverwaltung

Die Identität der Konferenzteilnehmer kann bei Bedarf über Zertifikate festgestellt werden, die in einem *Windows Certificate Store* verwaltet werden.

Über eine Teilnehmerliste ist für alle Teilnehmer während der Konferenz ersichtlich, welche Personen anwesend sind.

Rollen werden von NetMeeting nur bei der Sitzungsverwaltung unterstützt, d.h. der Teilnehmer, der die Konferenz gestartet hat (der Konferenzleiter), kann Teilnehmer jederzeit entfernen. Den Zugriff auf einzelne Komponenten des Systems kann der Konferenzleiter jedoch nicht dynamisch regeln.

Dateitransfer

NetMeeting bietet die Möglichkeit des integrierten Transfers von Dateien. Die Datenübertragung basiert auf ITU-Standard T.127 [ITU-T 96].

Vorteile:

- Integrierter Dateitransfer
- Adressierung von einzelnen Teilnehmern und Gruppen

Nachteile:

- Datentransfer per Unicast kann bei mehr als 20 Teilnehmern zu Netzengpässen führen, besonders bei der Anbindung mobiler Endgeräte

Applikationssteuerung

NetMeeting ermöglicht die Fernsteuerung von beliebigen Anwendungen auf Windows Systemen über ITU-Standard T.128.

Vorteile:

- Fernsteuerung von kompletten Anwendungen

Nachteile:

- Evtl. sehr hohe Netzlast, da die Ausgabe von Anwendungen teilweise als BitMap übertragen wird
- Sicherheitsrisiko durch Mächtigkeit der Steuerungsfunktion
- Keine Vergabe der Steuerung an **einen** bestimmten Teilnehmer
- Eine **direkte** Steuerung, wie sie in NetMeeting realisiert wurde, ist für das geplante präsentationsunterstützende System nicht uneingeschränkt vorteilhaft, da unter anderem anwendungsexterne Daten (z.B. Annotationen) konsistent mit dem Zustand der Anwendung gehalten werden müssen (z.B. Wiederanzeigen von Anmerkungen zu einer bestimmten Folie in PowerPoint)

Anmerkungen

NetMeeting enthält ein elektronisches Whiteboard-Werkzeug, welches auf dem ITU Standard T.126 basiert. Es unterstützt Freihandzeichnungen, einfache Text-Eingaben und die Erstellung anderer graphischer Objekte auf der gemeinsamen Arbeitsfläche. Die Zeichnungen sind seitenweise organisiert, wobei zwischen den Seiten umgeschaltet werden kann. Die Synchronisation des Hin- und

Herblätterns kann deaktiviert werden, so daß ein privates Blättern möglich wird. Das Erstellen von privaten Annotation ist innerhalb des Systems dagegen nicht möglich.

Bei einem weiteren Test hat sich herausgestellt, daß bei der gleichzeitigen Verwendung von NetMeeting und dem aktuellen SMARTboard-Treiber die Eingaben des SMARTboards direkt an das NetMeeting-Whiteboard gesendet werden (sog. SMARTaware-Technologie), was dazu führt, daß keine Annotation des momentanen Bildschirminhalts mehr möglich ist. Für den geforderten Verwendungszweck ist diese Kombination deshalb ungeeignet.

Vorteile:

- Unterstützung der geforderten graphischen Operationen
- Unterstützung mehrerer Seiten
- Netzwerkkommunikation basierend auf T.126-Standard
- Einfügen von BitMap-Grafiken
- Speichern und Laden von früheren Inhalten möglich

Nachteile:

- Speicherung erfolgt ausschließlich in speziellem Format (.nmv)
- Kein Ausblenden der Annotationen
- Keine privaten Annotationen möglich
- Keine gleichzeitige Bildschirmannotation möglich
- Alle Objekte sind frei beweglich, d.h. die zu annotierende Bildschirmkopie kann ebenfalls bewegt werden, was im gewünschten Anwendungsfall zu inkonsistenten und unsinnigen Situationen führen kann.

Zeigemöglichkeit

Das Whiteboard-Werkzeug ermöglicht es jedem Teilnehmer, ein Zeigersymbol innerhalb des gemeinsamen Arbeitsbereichs zu plazieren, wobei die einzelnen Zeiger individuell gefärbt werden können, um eine Unterscheidung zu ermöglichen.

Vorteile:

- Unterschiedliche Zeiger
- Färbung möglich (laut Dokumentation)

Nachteile:

- Färbung nicht möglich/nicht ersichtlich! (Testergebnis)
- Zeiger können nur vom jeweiligen Besitzer wieder entfernt werden

- Keine direkte Zuordnung von Zeiger zu Besitzer. Bei bis zu 20 Teilnehmern fällt das Unterscheiden der Farben schwer.
- Graphischer Fehler in der aktuellen Version 3.01 (siehe Abbildung 4-2)

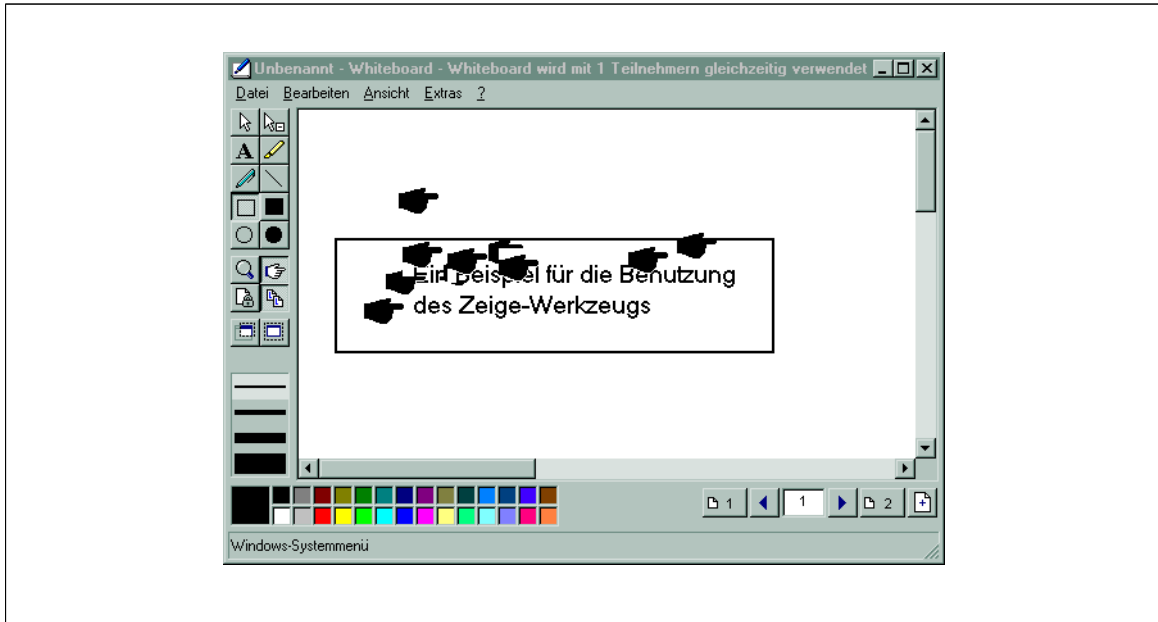


Abbildung 4-2: Kopier-Artefakte bei der Zeigerdarstellung

Floor Control

Das Whiteboard-Tool ermöglicht es, den modifizierenden Zugriff auf den Arbeitsbereich für alle Teilnehmer außer dem Konferenzleiter zu sperren (vgl. Abbildung 4-3).

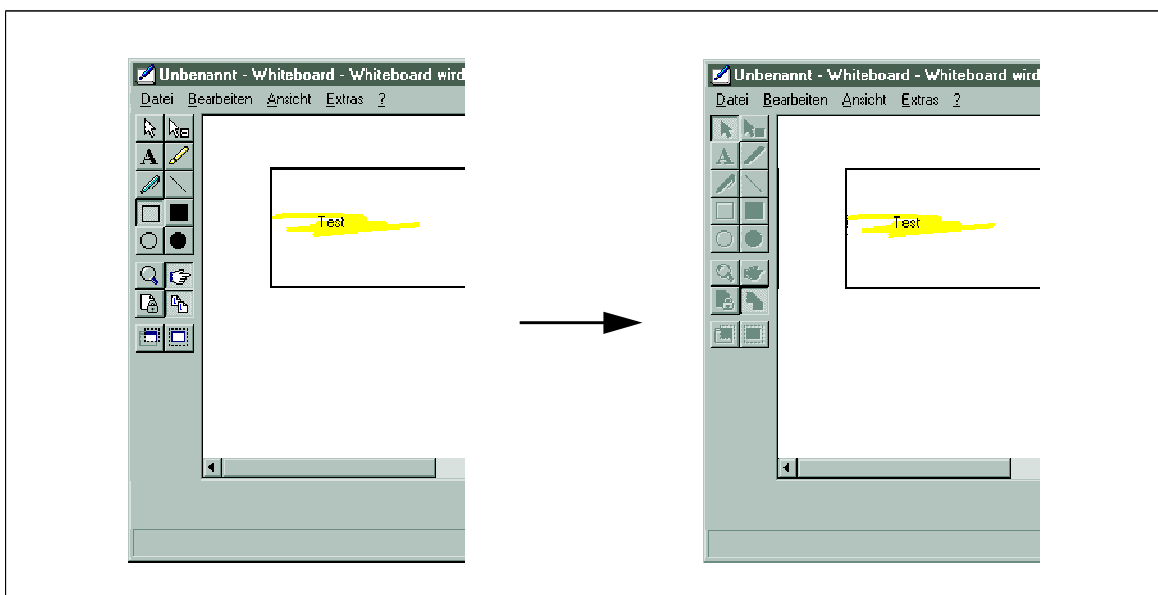


Abbildung 4-3: Deaktivierte Werkzeugleiste

Vorteile:

- Sperren der Eingabe möglich

Nachteile:

- Keine Vergabe von Rechten für einzelne Dienste (wie z.B. nur Zeiger, nur Freihandzeichnung) vorhanden
- Keine Vergabe von Rechten für *einzelne* Teilnehmer oder *Gruppen*, dadurch **keine sinnvolle *Floor Control*-Implementierung** möglich!

Mehrpunktконференzen

Während NetMeeting *Audio / Video*-Konferenzen nur als reine Punkt-zu-Punkt-Verbindungen unterstützt, ist laut Microsoft [MSSUP 00] die Zahl der Teilnehmer bei *Daten*-Konferenzen ohne A/V nur durch die verfügbare Bandbreite begrenzt.

In der Praxis zeigt sich, daß die Standardvorgabe für die Zahl der durch das Microsoft-Betriebssystem unterstützten, maximal gleichzeitig verfügbaren TCP/IP-Verbindungen manuell erhöht werden muß (Anpassung eines entsprechenden *Registry*-Eintrags).

Zudem kann bei einer größeren Zahl von Teilnehmern die sternförmige Ausgangsstruktur in eine baumartige Verbindungsstruktur umgewandelt werden, indem Benutzer nicht den Verbindungsleiter, sondern andere Teilnehmer der Konferenz kontaktieren. Bei diesem Ansatz hängt die Verbindung der folgenden Teilnehmer jedoch komplett von der Verbindung des ersten Teilnehmers ab. Diese Struktur empfiehlt sich normalerweise bei einer drohenden Überlastung des Rechners des Konferenzleiters.

Über die tatsächliche, *leistungsbedingte* Grenze der Teilnehmerzahl läßt sich ohne direkten Test mit mehr als 20 beteiligten Endgeräten jedoch keine Aussage treffen.

Vorteile:

- Theoretisch unbegrenzte Teilnehmerzahl
- Verschiedene Topologien möglich

Nachteile:

- Eigenheiten der Topologien sowie Probleme bei der Umsetzung in der Vorlesungspraxis
- Anpassung von Betriebssystem-Parametern notwendig
- Keine Daten über leistungsbedingte Grenzen

4.2.2 Allgemeine Anforderungen

Offenheit

Microsoft bietet ein kostenloses *Software Development Kit* an, mit dessen Hilfe die NetMeeting-Benutzungsschnittstelle in Webseiten und eigene Anwendungen eingebunden werden kann. Zudem besteht die Möglichkeit der Steuerung von grundlegenden Konferenzfunktionen wie dem Beitreten oder Verlassen einer Konferenz, dem Anwählen eines bestimmten Rechners und andere Dienste durch die *Scripting*- und *COM*-Schnittstellen.

Eine Integration von neuen Tools bzw. die Erweiterung der Funktionalität bestehender Tools (wie z.B. der Einführung einer *Floor Control*) ist aufgrund mangelnder *interner* Schnittstellen nicht möglich. Denkbar ist hingegen die Integration der Gesamtfunktionalität in ein eigenes System entlang der T.120-Kommunikationsprotokoll-Schnittstelle, wobei auch hier mit Problemen durch proprietäre Erweiterungen, bzw. durch Standardvorgriffe seitens Microsoft zu rechnen ist.

Plattformen

Microsoft NetMeeting ist ausschließlich für Microsoft Windows Betriebssysteme erhältlich.

Verfügbarkeit

NetMeeting 3.01 ist kostenlos von der Microsoft-Website zu beziehen. Es existieren Versionen für MS Windows 9x und NT 4.0. Der für den Verzeichnisdienst notwendige *MS Internet Locator Server* ist ebenfalls in der Version 2.0 kostenlos per Download zu beziehen. Im Hinblick auf Versionskonflikte läßt sich bisher sagen, daß NetMeeting 3 mit Version 2 kompatibel ist, wobei das Anwendungsprotokoll der vorhergehenden Version nicht vollständig auf dem ITU Standard T.120 basiert.

Konsistenz

Die Benutzungsoberfläche der NetMeeting-Software ist im Hinblick auf die Anforderungen in sich konsistent. Alle Werkzeuge lassen sich zentral starten. Eine öffentliche Schnittstelle zur Integration neuer Komponenten in die bestehende GUI ist nicht vorhanden.

Stabilität

Die Stabilität gegenüber Fehlbedienungen kann nur durch entsprechende Tests in Szenarien mit Teilnehmerzahlen von 20 und mehr ermittelt werden.

Im Hinblick auf die mutwillige Störung des Betriebs der Software ist anzumerken, daß im WWW Hinweise auf Sicherheitsprobleme mit NetMeeting zu finden sind. So weist zum Beispiel die unmodifizierte Version 3.01 einen Schwachpunkt auf, der durch einen DoS-Angriff ausgenutzt werden kann [MSDOS 00].

- Keine Daten über das Systemverhalten bei Konferenzen mit mehr als 20 Teilnehmern
- Als Microsoft Produkt ist NetMeeting ein bevorzugtes Ziel von Angriffen böswilliger Benutzer
- Bei Bekanntwerden einer neuen Sicherheitslücke ist man im Gegensatz zu Systemen, bei denen der Quellcode vorliegt, auf die Reaktion der Microsoft Corporation angewiesen, was in Anbetracht der zu erwartenden Attraktivität der Beeinflussung eines Systems, welches in Vorlesungen Anwendung finden soll, als bedenklich zu bewerten ist

Sicherheit

Die Authentifizierung der NetMeeting-Benutzer kann durch die Verwendung von Zertifikaten abgesichert werden.

Firewalltauglichkeit

Der Betrieb von NetMeeting *innerhalb* abgeschirmter Netze ist unproblematisch. Verzeichnisserver, die sich außerhalb des abgeschirmten Netzes befinden, können durch Öffnen eines einzigen TCP-Ports genutzt werden. Die Kommunikation mit Teilnehmern außerhalb des Netzes ist in Form einer reinen Daten-Konferenz ebenfalls ohne größere Anpassungen möglich.

Allein das Durchführen von Audio/Video-Konferenzen über Firewall-Grenzen hinweg erweist sich als problematisch, da dynamisch ausgehandelte TCP und UDP-Ports verwendet werden (vgl. Anhang, Abbildung 10-2).

4.2.3 Fazit

Anforderung	+/o/-	Kommentar
Anmeldung	+	Auswahl aus Verzeichnis (ILS)
Teilnehmer/Rollen	+	
Dateitransfer	+	Aber: nur Unicast!
Fernsteuerung	+	Sicherheitsproblem wg. mächtiger Funktion
Anmerkungen	o	Spezielles Format, keine private Annotation
Zeiger	+	Führt zu Darstellungsfehlern
Floor Control	--	Keine Rechtevergabe pro Dienst/Benutzer
Mehrpunktconf.	+/?	(Keine Informationen über große Konferenzen)
Offenheit	--	Interne Schnittstellen nicht offen!
Plattformen	-	Nur für MS Windows-Betriebssysteme
Verfügbarkeit	+	Kostenlos
Konsistenz	+	Zentrale Bedienung
Stabilität	+/?	(Keine Informationen über große Konferenzen)
Sicherheit	+	Authentifizierung per Zertifikat
Firewalltauglichkeit	+	Kaum Probleme mit nur-Daten-Konferenzen

Tabelle 4-1: Bewertung NetMeeting

Microsoft NetMeeting stellt den Großteil der für das Projekt erforderlichen, grundlegenden Funktionalitäten zur Verfügung (vgl. Tabelle 4-1). Das Produkt hat jedoch im Hinblick auf die erweiterten Anforderungen, die im Rahmen der Anforderungsanalyse zusammengestellt wurden, unüberbrückbare Defizite.

Eines der größten Hindernisse ist die fehlende Offenheit der internen Schnittstellen der Software, die eine echte Erweiterung der Werkzeuge nicht erlaubt. Besonders im Bereich der Rechtevergabe ist die Erweiterung der bisher nur unzureichend unterstützten Funktionalität der *Floor Control* jedoch eine zwingende Voraussetzung für den erfolgreichen Einsatz im Vorlesungs-/Präsentationsbetrieb.

Was sich zudem besonders unter dem Aspekt der Unterstützung mobiler Endgeräte negativ bemerkbar machen würde, ist die Tatsache, daß der Datentransfer ausschließlich per Unicast stattfindet. Bei bis zu 20 per drahtlosem Netzwerk verbundenen Teilnehmern würde die Übertragung von Bildschirmkopien (wie sie zur Umsetzung der Anmerkungs-funktionalität notwendig sein könnte) mit einem Datenaufkommen von mehreren 100 Kilobyte pro Folie mit

ziemlicher Sicherheit zu unvertretbar langen Verzögerungszeiten führen. Hinzu kommt das Fehlen von Kenntnissen über das Systemverhalten von Net-Meeting unter besagter Belastung durch bis zu 20 Teilnehmer.

Für den Fall, daß man den Einsatz des Endsystems nicht auf Präsentationen in einem *garantiert* homogenen Rechnerumfeld beschränken möchte, muß die ausschließliche Verfügbarkeit der Software für Microsoft Windows Betriebssystemen ebenfalls als Negativpunkt gewertet werden.

Man kann zusammenfassend sagen, daß eine Empfehlung für die Entwicklung des geforderten Systems basierend auf der Konferenzsoftware Microsoft Net-Meeting aufgrund der erwähnten Analyseergebnisse nicht vertretbar ist. Denkbar wäre jedoch die Einbindung des Programms als *zusätzlichen* Client entlang der standardisierten T.120-Schnittstelle. Eine derartige Schnittstelle könnte auch nachträglich in das System eingebunden werden, was sich angesichts der mangelnden Verfügbarkeit von frei erhältlichen Implementierungen des besagten Standards auch anbieten würde.

4.3 Eignungsanalyse MBone-Tools

Gegenstand dieser Analyse ist die Untersuchung der MBone Tools *Session Directory, SDR Version 3.0* in Verbindung mit dem *UCL Shared Whiteboard, WBD Version 1.0ucl4*, wobei in beiden Fällen ausschließlich die Win32-Version getestet wurde.

4.3.1 Spezielle Anforderungen

Anmeldungsunterstützung

Die Anmeldung an eine Konferenz erfolgt mit Hilfe des SDR-Tools durch einfaches Auswählen aus einer Liste. Verspätet eintreffende Teilnehmer werden ebenfalls unterstützt, da die Dauer der Sichtbarkeit der Auswahl allein vom Konferenzleiter abhängt. Zudem werden die im Whiteboard bereits erstellten Zeichnungen automatisch auf die Rechner von Neuankömmlingen übertragen.

Teilnehmerverwaltung

Das SDR-Werkzeug ermöglicht die Identifikation von Benutzern durch einen PGP-Schlüssel. Allerdings scheint es keine Verbindung zwischen der Teilnehmerverwaltung von SDR und WBD zu geben. Eine genaue Identifizierung der WBD-Teilnehmer ist demnach nicht direkt möglich.

Bei reinen Datenkonferenzen ohne Verwendung von Audio/Video-Komponenten fehlt zudem eine Rückmeldung über die Anzahl und die Namen der Konferenzteilnehmer, was sich im Hinblick auf die im CSCW-Umfeld wichtige *Team Awareness* negativ auswirkt.

Dateitransfer

Die Möglichkeit des integrierten Dateitransfers wird nicht unterstützt.

Applikationssteuerung

Eine Möglichkeit zur Fernsteuerung von Anwendungen wird von den beiden betrachteten Werkzeugen nicht unterstützt.

Anmerkungen

Das Whiteboard unterstützt die geforderten Anmerkungsoperationen. Es werden mehrere Seiten verwaltet, die das wiederholte Anzeigen von zuvor erstellten Zeichnungen möglich macht. Eine Synchronisation der aktuell sichtbaren Seite wird zwischen den Arbeitsbereichen der Teilnehmer nur bei Änderungen durchgeführt. Es gibt nur einen öffentlichen Arbeitsbereich, private Annotationen sind nicht möglich. Es macht sich zudem negativ bemerkbar, daß Seiten nicht komplett gelöscht werden können. Lediglich eine objektbasierte Löschfunktion ist vorhanden.

Externe Texte und Postscript-Dateien können importiert werden, wobei eine direkte Unterstützung für ein Bitmap-Format vollständig fehlt. Zudem zeigt sich, daß der erwähnte Import sehr langsam abläuft. Die Möglichkeit des Speicherns von Inhalten in Dateien fehlt völlig.

Zeigemöglichkeit

Eine Möglichkeit zur Anzeige eines ferngesteuerten Zeigersymbols ist nicht vorhanden.

Floor Control

Das dynamische Sperren von Zugriffsrechten ist nicht möglich.

Mehrpunktконференzen

Angesichts der auf große Gruppen ausgerichteten Architektur der MBone-Tools ist nicht anzunehmen, daß in Konferenzen mit 20 Teilnehmern technische Probleme auftreten.

4.3.2 Allgemeine Anforderungen

Offenheit

Der Quellcode der beiden Werkzeuge SDR und WBD ist per Download im Internet erhältlich. Anpassungen an der Funktionalität sind grundsätzlich möglich. Da WBD nicht auf einem standardisierten Protokoll wie z.B. der ITU-T.120-Standardfamilie aufbaut, ist die Integration mit anderen Produkten, die diese Schnittstelle unterstützen, nicht ohne erhebliche Änderungen möglich.

Plattformen

Es werden vorkompilierte Programmpakete für diverse Unix-Derivate und für Microsoft Windows-Betriebssysteme angeboten. Eine Anpassung an noch nicht unterstützte Plattformen ist dank des erhältlichen Quellcodes denkbar.

Verfügbarkeit

Die untersuchten Werkzeuge sind per Download aus dem Internet [UCL 00] zu beziehen. Die Benutzung ist für den nicht-kommerziellen Gebrauch kostenlos.

Konsistenz

Durch das SDR-Werkzeug werden die einzelnen Programme, die innerhalb einer Konferenz eingesetzt werden, automatisch gestartet. Die Bedienung von WBD ist teilweise etwas umständlich und könnte nach ergonomischen Aspekten optimiert werden.

Stabilität

In den Tests mit den aktuellen Versionen konnten keine offensichtlichen Instabilitäten entdeckt werden.

Sicherheit

Da die Authentifizierung der Teilnehmer in SDR und WBD nicht einheitlich erfolgt, beziehungsweise im letzteren Fall scheinbar nicht vorhanden ist, ist die Möglichkeit einer sicheren Authentifizierung nicht gegeben.

Firewalltauglichkeit

Aus Mangel an Informationen und Test-Möglichkeiten ist es nicht möglich, eine Aussage über die Probleme beim Zusammenspiel des SDR-Protokolls und handelsüblicher Firewall-Systeme zu machen.

4.3.3 Fazit

Die grundlegende Whiteboard- und Konferenz-Funktionalität wird mit einigen Abstrichen unterstützt (vgl. Tabelle 4-2). Als besonders negativ ist das Fehlen der Möglichkeit zu sehen, die Eingaberechte des gemeinsamen Arbeitsbereiches während einer Konferenz dynamisch zu ändern. Für den Fall, daß keine A/V-Komponenten verwendet werden, besteht außerdem keine Rückmeldung über die Zahl und die Identität der Konferenzteilnehmer, was im Hinblick auf Team Awareness als negativ zu werten ist.

Auf der anderen Seite sind die MBone-Tools vorteilhaft in größeren Konferenzen einsetzbar. Die freie und plattformübergreifende Verfügbarkeit ist ein weiterer Pluspunkt.

Anforderung	+/o/-	Kommentar
Anmeldung	+	Auswahl aus Verzeichnis
Teilnehmer/Rollen	o	Fehlt in WBD, keine Team Awareness
Dateitransfer	-	
Fernsteuerung	-	
Anmerkungen	-	Nur öffentlich, keine Bitmaps, kein Export
Zeiger	-	
Floor Control	-	Kein dynamisches Sperren!
Mehrpunktconf.	+	
Offenheit	+	Quellcode verfügbar
Plattformen	+	Unix, Windows, weitere denkbar
Verfügbarkeit	+	Kostenlos
Konsistenz	o	Teilweise umständliche Bedienung
Stabilität	+	Keine Anzeichen von Instabilitäten
Sicherheit	-	Authentifizierungsunterschiede SDR/WBD
Firewalltauglichkeit	?	(Keine Informationen vorhanden)

Tabelle 4-2: Bewertung MBone-Tools

Abschließend ist zu sagen, daß eine Implementierung des Systems auf Basis der behandelten MBone-Werkzeuge dank des verfügbaren Quellcodes möglich ist. Die Anpassung und die notwendige Erweiterung der Funktionalität würde aber mit großer Wahrscheinlichkeit den zeitlichen Rahmen dieser Studienarbeit sprengen. Zudem könnte sich der verteilte Charakter der MBone-Anwendung als Hindernis bei der Realisierung einer zentralen *Floor Control* erweisen.

4.4 Eignungsanalyse JETS

Es folgen die Analyseergebnisse für die Konferenzsoftware *Java Enabled Telecollaboration System, JETS Version 2.0 bzw. JETS 2000 Alpha 1*, die im Multimedia Communications Research Laboratory der University of Ottawa entwickelt wurde.

4.4.1 Spezielle Anforderungen

Anmeldungsunterstützung

Die Anmeldung an eine bestehende Konferenz erfolgt über die Anwahl der WWW-Adresse des zuständigen JETS-Servers. Verspätet ankommende Teilnehmer können sich ebenfalls durch diese Prozedur anmelden, eine Übertragung bereits vorgenommener Whiteboard-Zeichnungen wird jedoch nicht unterstützt.

Teilnehmerverwaltung

Teilnehmer werden durch einen Benutzernamen sowie ein simples, fünfstelliges, numerisches Passwort identifiziert. Die Liste der Mitglieder ist für alle Teilnehmer sichtbar (*Team-Awareness*). Die Rollenentscheidung Zuhörer/Leiter erfolgt durch die Anwahl unterschiedlicher WWW-Seiten.

Dateitransfer

Die Funktionalität eines integrierten Dateitransfers wird nicht unterstützt.

Applikationssteuerung

Die Funktionalität der Applikationssteuerung wird nicht unterstützt. Obwohl auf der Webseite von JETS die Rede von einer Unterstützung für interaktive PowerPoint-Präsentationen ist, konnte diese weder in der Dokumentation noch in Tests entdeckt werden. Durch die Bereitstellung einer API zur Entwicklung eigener Java-Applets und -Applications ist jedoch die Möglichkeit der Fernsteuerung selbstentwickelter Komponenten gegeben.

Anmerkungen

Das Whiteboard unterstützt Freihandzeichnungen aber keine Texteingaben. Das Importieren von Bitmaps und ganzen Folgen von Bitmaps (steuerbare *Slideshows*) ist möglich. Eine Speicherung der Zeichnungen findet nicht statt, demnach ist auch eine vorübergehende Ausblendung nicht möglich. Es existiert keine Funktionalität, um den Inhalt des Whiteboards abzuspeichern. Die Möglichkeit der privaten Annotation ist nicht gegeben.

Zeigemöglichkeit

JETS unterstützt keine Zeiger.

Floor Control

Es ist bereits eine umfassende *Floor Control* in JETS implementiert, welche die Eingaberechte für jeden Benutzer in drei Stufen regelt:

1. Kein Zugang
2. Nur Anzeigen
3. Sowohl Anzeigen, als auch Verändern

Die Vergabe der Rechte wird durch den Leiter (*Chairperson*) geregelt (siehe Abbildung 4-4), eine Änderung wird dem jeweiligen Teilnehmer durch eine entsprechende Färbung der GUI-Elemente seiner Kontrollkonsole mitgeteilt (rot, gelb, grün). Ein Teilnehmer kann jederzeit das Eingaberecht für ein bestimmtes Werkzeug beantragen, wobei dies der Chairperson per Pop-Up-Fenster mitgeteilt wird.

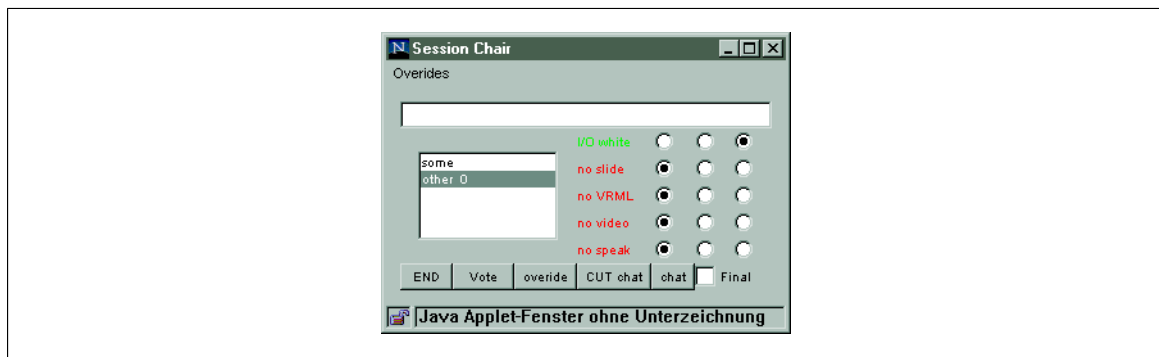


Abbildung 4-4: Das Fenster zur Vergabe des Rederechts (Floor Control)

Mehrpunktконференzen

Das System ist als sternförmige Client-Server-Architektur realisiert. In der Zahl der Teilnehmer gibt es keine erkennbaren designbedingten Beschränkungen.

4.4.2 Allgemeine Anforderungen

Offenheit

JETS bietet Entwicklern eine API, mit deren Hilfe eigene Erweiterungen in Form von Java-Applets oder -Applications integriert werden können. Die API besteht dabei im wesentlichen aus zwei Komponenten:

- Ein Broadcast-Kanal, über den Daten an alle anderen Teilnehmer geschickt werden können.
- Eine Lock-Funktion, durch die der exklusive Zugriff auf die Anwendung sichergestellt wird.

Insbesondere fehlt die Möglichkeit, direkt auf die Kernkomponenten des Sessionmanagements (Benutzerverwaltung) zuzugreifen.

Plattformen

Das System wurde komplett in Java entwickelt. Die Software-Voraussetzung der Teilnehmerseite besteht lediglich aus einem handelsüblichen Web-Browser mit Java-Unterstützung. Der Einsatz auf Handheld-Rechnern ist angesichts der Systemvoraussetzungen ebenfalls denkbar.

Verfügbarkeit

Der nicht-kommerzielle Einsatz zu Evaluationszwecken ist kostenlos.

Konsistenz

Die Benutzungsschnittstelle besteht aus einer Menge von Java-Applet-Fenstern, deren Bedienung sich teilweise als umständlich erweist und die einige Optimierung im Hinblick auf Ergonomie nötig hätte, falls sie im Lehrbetrieb Einsatz finden sollte.

Stabilität

Bei der Verwendung von JETS traten die folgenden Probleme auf:

- Bei Tests auf unterschiedlichen Systemen (Windows NT 4.0 mit Internet Explorer 5.0 und Netscape 4.75 sowie SunOS in Verbindung mit Netscape 4.76) kam es gelegentlich zu unbehandelten Exceptions.
- Während der Tests ist es außerdem häufig vorgekommen, daß Teilnehmer einer Konferenz nicht beitreten konnten, obwohl eine *Chairperson* anwesend war.

Eine Lokalisierung der genauen Fehlerursache konnte nicht durchgeführt werden.

Sicherheit

Die Authentifizierung erfolgt durch ein einfaches numerisches Passwort, das vom Sitzungsleiter bestimmt und an die Teilnehmer verteilt wird. Zumindest für ein prototypisches System bietet dieses Vorgehen ein ausreichendes Maß an Sicherheit. Zum Problem der Umgehung der Authentifizierung gibt es keine Erkenntnisse.

Firewalltauglichkeit

Der Betrieb innerhalb eines abgeschotteten Netzwerks sollte unproblematisch durchführbar sein, über Firewallgrenzen hinweg ist ein Betrieb wahrscheinlich nicht, oder nur eingeschränkt möglich, da dynamisch ausgehandelte TCP-Ports Verwendung finden.

4.4.3 Fazit

Anforderung	+/o/-	Kommentar
Anmeldung	+	Einfach per Webseite
Teilnehmer/Rollen	+	Personen und Rollen (Chairperson) unterstützt
Dateitransfer	-	
Fernsteuerung	-	
Anmerkungen	-	Kaum Funktionalität, keine Speicherung
Zeiger	-	
Floor Control	+	Komplette Rechtevergabe vorhanden
Mehrpunktconf.	+	Keine Informationen über Grenzen vorhanden
Offenheit	+	Umfangreiche API
Plattformen	+	Komplett in JAVA, Steuerung als Applets realisiert
Verfügbarkeit	+	Kostenlos
Konsistenz	-	Viele Fenster, Mangel in der Ergonomie
Stabilität	-	Anmeldungsprobleme, unbehandelte Exceptions
Sicherheit	+	Für Prototyp ausreichend
Firewalltauglichkeit	-/?	(Probleme über Firewallgrenzen hinweg)

Tabelle 4-3: Bewertung JETS

Für das JETS-System spricht die bereits vorhandene Floor Control und die Plattformunabhängigkeit. Als negativ ist die beschränkte Whiteboard-Funktionalität zu werten, die außerdem keine Speicherung der gezeichneten Objekte zuläßt. In diesem Bereich und an der Benutzungsoberfläche müßten einige Änderungen und Erweiterungen vorgenommen werden (vgl. Tabelle 4-3).

Als stärkstes Argument gegen den Einsatz von JETS sprechen allerdings die während der Tests aufgetretenen Instabilitäten. Die Tatsache, daß die Probleme bei der Konferenzanmeldung und die unbehandelten Ausnahmen auf unterschiedlichen Rechner- und Softwaresystemen aufgetreten sind, läßt vermuten, daß es sich dabei um ein Problem seitens des JETS-Systems handelt. Eine uneingeschränkte Empfehlung für den Einsatz im alltäglichen Lehrbetrieb ist aus den beschriebenen Gründen nicht vertretbar.

4.5 Eignungsanalyse JaTeK

Das an der Technischen Universität Dresden durchgeführte Teleteaching-Projekt *Java Based Teleteaching Kit, JaTeK* umfaßt ein Modul namens *Java Based Workgroup Support, JaWos*, welches der Kommunikationsunterstützung innerhalb von Übungsgruppen dient. Gegenstand der folgenden Analyse ist das in diesem Modul enthaltene Whiteboard.

4.5.1 Einleitung

Leider war es nicht möglich, die Funktionalität des Whiteboards anhand direkter Tests zu untersuchen, da die Software in der zur Verfügung stehenden Hardware-Umgebung nicht benutzbar war. Während der Zugriff auf die eigentlichen Lerninhalte über das Hauptmodul problemlos ablief, kam es beim Start des Whiteboards, sowie aller anderen Werkzeuge zur Gruppenunterstützung, zu langen, mehrminütigen Verzögerungszeiten bis hin zu überhaupt keiner Rückmeldung nach über 40 Minuten.

Eine entsprechende Anfrage beim JaTeK-Team der TU-Dresden ergab, daß das integrierte Whiteboard in der aktuellen Version nicht von einem Netzwerk aus benutzbar ist, welches durch eine Firewall abgeschirmt ist. Eine derartige Abschirmung ist jedoch in der lokal verfügbaren Netzwerkkonfiguration der Universität Stuttgart sowie der Fakultät Informatik gegeben.

Da die Lösung des Problems z.B. durch die Installation eines lokalen Servers im Rahmen dieser Studienarbeit als zu aufwendig erschien, wurde von weiteren Tests abgesehen. Die Bewertung basiert aus diesem Grund zum größten Teil auf den öffentlich verfügbaren Informationen über das System.

4.5.2 Bewertung

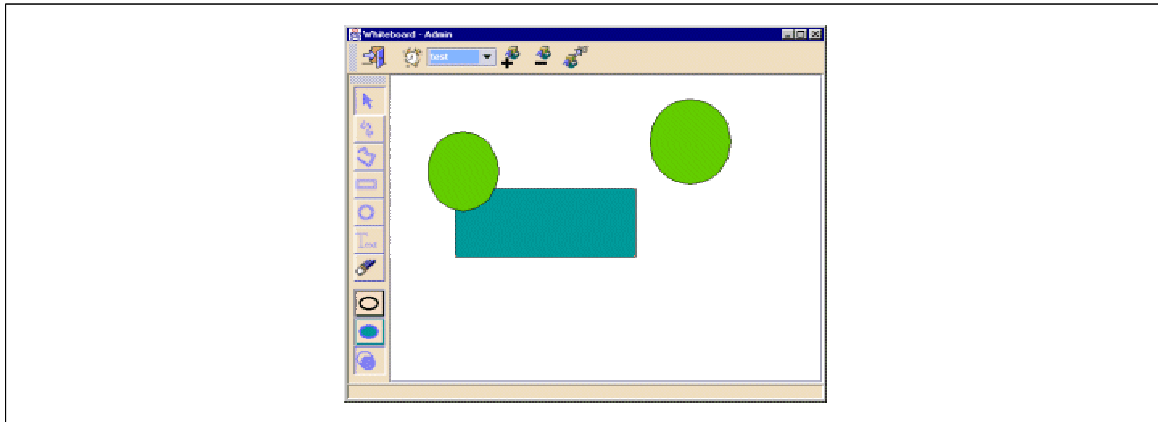


Abbildung 4-5: Das JaWoS-Whiteboard

Vorteile:

- Als positiv ist zu bewerten, daß die Software komplett in Java geschrieben wurde und damit prinzipiell plattformübergreifend eingesetzt werden kann
- Die Client-Software ohne Autorenfunktionalität ist frei verfügbar
- Benutzerauthentifizierung per Passwordeingabe
- Anmeldungsunterstützung über zentralen Server
- Mehrpunktkonferenz durch Client-Server-Architektur
- Konsistentes Bild der Benutzungsoberfläche, verschiedene Werkzeuge (Chat, Shared Text, Blackboard, Whiteboard) sind zentral integriert
- Geforderte Whiteboard-Zeichenoperationen werden unterstützt (siehe Abbildung 4-5)

Nachteile:

- Probleme bei der Verwendung in abgeschirmten Netzen
- Quellcode nicht direkt frei erhältlich
- Fehlen von Zusatzfunktionalität wie dem integrierten Dateitransfer, einer Möglichkeit der Applikationssteuerung oder der Floor Control
- Keine Unterstützung von Bitmaps im Whiteboard

4.6 Zusammenfassung

Es folgt eine abschließende Zusammenfassung der Untersuchungsergebnisse.

4.6.1 Untersuchte Systeme

Gegenstand der Untersuchungen waren:

1. Microsoft **NetMeeting** Version 3.01
2. Die **MBone-Tools** *Session Directory, SDR 3.0* und *UCL Shared Whiteboard, WBD Version 1.0ucl4*
3. Das *Java-Enabled Telecollaboration System, JETS Version 2.0* bzw. *Version 2000 Alpha 1*, entwickelt im MCRLab der University of Ottawa
4. Die Whiteboard-Komponente des *Java-Based Teleaching Kit, JaTeK* der TU-Dresden

4.6.2 Kriterien

Bei der Bewertung wurden die folgenden Kriterien angewandt:

- **Anmeldungsunterstützung:** Wie groß ist der Aufwand für das Auffinden von und das Anmelden an eine Konferenz?
- **Teilnehmerverwaltung:** Können Personen und Rollen (Vortragender/Teilnehmer) sicher und für alle Teilnehmer sichtbar identifiziert und unterschieden werden?
- **Dateitransfer:** Können Dateien innerhalb der Konferenz integriert kopiert werden?
- **Applikationssteuerung:** Besteht die Möglichkeit, Anwendungen fernzusteuern?
- **Anmerkungen:** Unterstützt das Whiteboard Freihandzeichnungen, Texteingaben und Bitmaps (Bitmaps zur Annotation von Bildschirmkopieren)? Werden Inhalte gespeichert? Können Zeichnungen exportiert werden? Kann man nicht-öffentliche Zeichnungen erstellen?
- **Zeigemöglichkeit:** Gibt es fernsteuerbare Zeigersymbole? Besteht die Möglichkeit einer Zuordnung zwischen Zeiger und steuerndem Teilnehmer?
- **Floor Control:** Können die Zugriffsrechte auf die einzelnen Dienste gesteuert werden?
- **Mehrpunktконференzen:** Werden Konferenzen mit 20 Teilnehmern unterstützt?
- **Offenheit:** Welche Schnittstellen zur Integration und Erweiterung bestehen?
- **Plattformen:** Welche Soft- und Hardwareplattformen werden unterstützt?
- **Verfügbarkeit:** Wieviel kostet die Anschaffung des Systems?

- **Konsistenz:** Wie konsistent sind das Erscheinungsbild und die Bedienung der Software?
- **Stabilität:** Wie robust ist das System im Alltagsbetrieb und gegenüber versehentlichen und mutwilligen Fehlbedienungen?
- **Sicherheit:** Wie sicher ist die Authentifizierung?
- **Firewalltauglichkeit:** Kommt es im Betrieb in einem abgeschirmten Netz zu Problemen?

4.6.3 Ergebnisse

Microsoft NetMeeting

Viele Features, aber:

- unzureichende Erweiterungsmöglichkeit
- fehlende Rechtevergabe
- Datenübertragung per Unicast
- Probleme mit SMARTboard (Whiteboard verhindert weitere Bildschirmnotation)

deshalb nicht zu empfehlen.

MBone-Tools

Vorteile:

- Quellcode verfügbar
- Plattformübergreifend

Nachteile:

- Keine dynamische Rechtevergabe
- Authentifizierung nur in SDR
- Umfangreiche Erweiterungen (Whiteboard, Authentifizierung, Rechtevergabe) notwendig
- Dezentraler Charakter der Anwendung erschwert Realisierung einer zentralen Rechtevergabe

Eine Implementierung auf Basis der MBone-Tools ist zwar denkbar, aber zeitlich wahrscheinlich zu aufwendig, da zu viele Änderungen notwendig sind. Deshalb ist dieses Vorgehen nicht zu empfehlen.

JETS

Vorteile:

- Vorhandene Unterstützung der Rechtevergabe
- Notwendige Erweiterungen der anderen Funktionalitäten

Eine Empfehlung für JETS kann jedoch aufgrund der Instabilität nicht gegeben werden. Unbehandelte Exceptions, Probleme bei manchen Konferenzanmeldungen und die dauernde Ausgabe von Debug-Informationen lassen Zweifel an der Ausgereiftheit des Systems aufkommen.

JaTeK

Vorteile

- Plattformunabhängig, da komplett in JAVA geschrieben
- Client frei erhältlich
- Authentifizierung per Benutzername und Passwort
- Konsistente Bedienung, zentrale Steuerung

Nachteile:

- Probleme mit Firewalls
- Quellcode nicht direkt frei erhältlich
- Fehlende Funktionalität: Dateitransfer, Applikationssteuerung, Floor Control!
- Whiteboard unterstützt keine Bitmaps!

Eine Implementierung auf Basis von JaTeK ist denkbar, aber aufgrund des zu erwartenden zeitlichen Aufwands für die notwendigen Anpassungen der Grundfunktionalität nicht zu empfehlen.

4.6.4 Fazit

Anforderung	NetMeeting	MBone	JETS	JaTeK
Anmeldung	+	+	+	?
Teilnehmer/Rollen	+	0	+	?
Dateitransfer	+	-	-	-
Fernsteuerung	+	-	-	-
Anmerkungen	0	-	-	0
Zeiger	+	-	-	?
Floor Control	-	-	+	-
Mehrpunktkonf.	+/?	+	+	?
Offenheit	-	+	+	?
Plattformen	-	+	+	+
Verfügbarkeit	+	+	+	0
Konsistenz	+	0	-	+
Stabilität	+/?	+	-	?
Sicherheit	+	-	+	+
Firewalltauglichkeit	+	?	-/?	-

Tabelle 4-4: Gesamtbewertung

Keines der betrachteten Systeme erweist sich im Hinblick auf die projektbedingten Anforderungen als empfehlenswert (vgl. Tabelle 4-4). Entweder mangelt es an der Erweiterbarkeit oder an der Stabilität, oder aber der Umfang der notwendigen Änderungen würde den zeitlichen Rahmen dieser Studienarbeit sprengen.

Da sich eine Entwicklung auf der Grundlage eines kompletten Konferenzsystems als nicht sinnvoll erwiesen hat, folgt im nächsten Kapitel eine Eignungsanalyse für Einzelkomponenten, welche als Basis für das gewünschte System verwendet werden können.

4.7 Quellenangabe

- MSRES 99** Microsoft Corporation
NetMeeting Resource Kit, Chapter 3: Finding People
Microsoft Website
<http://www.microsoft.com/windows/netmeeting/Corp/reskit/Chapter3/>
- ITU-T 96** International Telecommunication Union
Recommendation T.120 (07/96) - Data protocols for multimedia conferencing
ITU Electronic Bookshop
- MSSUP 00** Microsoft Corporation
Q243116 - Only a Certain Number of Clients Can Join a NetMeeting Conference
Microsoft Product Support Website
<http://support.microsoft.com/support/kb/articles/Q243/1/16.ASP>
- MSDOS 00** Microsoft Corporation
Q273854 - Denial of Service Can Occur with Microsoft NetMeeting
Microsoft Product Support Website
<http://support.microsoft.com/support/kb/articles/Q273/8/54.ASP>
- UCL 00** UCL Networked Multimedia Research Group
Mbone Conferencing Applications
<http://www-mice.cs.ucl.ac.uk/multimedia/software>
- MB 00** BZVD TU Dresden
MBone - Softwareliste
<http://bzvd.urz.tu-dresden.de/mbone/software.html>
- JETS 99** Multimedia Communications Research Laboratory, University of Ottawa
JETS: A Java-Enabled TeleCollaboration System
<http://www.mcrlab.uottawa.ca/jets/>
- JATEK 00** Professur für Rechnernetze, Technische Universität Dresden
Java Based Teleteaching Kit
<http://telet.inf.tu-dresden.de/JaTeK.htm>

5 Eignungsanalyse von Einzelkomponenten

Da die Analyseergebnisse des vorherigen Kapitels gegen die Verwendung eines kompletten Konferenzsystems sprechen, folgt in diesem Kapitel eine Untersuchung von Einzelkomponenten im Hinblick auf den Einsatz im gewünschten Zielsystem.

5.1 Präsentations-Fernsteuerung

Im Rahmen einer bereits abgeschlossenen Studienarbeit [NGUY 00] wurde auf der Basis von Java und Jini ein System zur Fernsteuerung einer PowerPoint-Präsentation entwickelt. Aufbau und Funktionsweise dieses Systems werden im folgenden Abschnitt zusammenfassend erläutert.

5.1.1 Aufbau

Das System zur Fernsteuerung wurde in einer Client-Server-Architektur entworfen. Die Komponenten sind im einzelnen:

- Jini-Lookup-Service. Dieser Service entspricht einer zentralen Anlaufstelle, durch die der Kontakt zwischen Client und Server vermittelt wird. Alle aktiven Server-Objekte melden sich an dieses System an, um bei entsprechenden Anfragen von Clients gefunden zu werden.
- Server-Objekt. Das Server-Objekt übernimmt die Kommunikation zwischen einem Client, dem Lookup-Service, einem evtl. angeschlossenen SMARTBoard sowie der PowerPoint-Anwendung.
- Proxy-Objekt. Die Kommunikation zwischen Client und Server erfolgt nicht direkt, sondern über das sog. Proxy-Objekt. Dieses Objekt wird bei Programmstart von einem HTTP-Server heruntergeladen. Es enthält die eigentliche Client-Anwendung, durch welche die Dienst-Funktionalität der Fernsteuerung realisiert wird. Durch dieses Vorgehen wird sichergestellt, daß immer eine zum Server kompatible Proxy-Version verwendet wird.
- Client-Objekt. Das Client-Objekt dient lediglich der ersten Kontaktaufnahme mit dem Server sowie dem Herunterladen und Aktivieren des Proxy-Objekts.
- Windows Scripting Host. Über das Windows Scripting Host-System wird der Kontakt zwischen Server-Objekt und PowerPoint-Anwendung hergestellt. Es dient der Übermittlung einfacher Befehle an PowerPoint, wie zum Beispiel Anzeigen der nächsten oder der vorhergehenden Seite.
- PowerPoint. Dies ist die eigentliche Anwendung, die ferngesteuert die Anzeige der Präsentationsdatei übernimmt.

5.1.2 Bewertung

Die Funktionalität einer Präsentationsfernsteuerung wird von diesem System erfolgreich realisiert. Darüber hinaus existiert eine Schnittstelle, über die Eingaben auf dem SMARTBoard an den angeschlossenen Client übertragen werden. Ein Nachteil, daß keinerlei Sitzungsverwaltung und auch keine Möglichkeit der Authentifizierung vorhanden ist. Zudem ist die Annotation von Bildschirmhalten nicht möglich, da lediglich innerhalb eines eigenen Fensters Zeichnungen angebracht werden können.

Als weiteres Vorgehen empfiehlt es sich, die Kernfunktionalität dieser Arbeit direkt zu übernehmen und mit anderen Komponenten zu kombinieren. Die Bindung an JINI als Lookup-Service ist dabei evtl. zugunsten einer allgemeineren Lösung auf der Ebene des Kommunikationssystems zu überdenken.

5.2 Java Shared Data Toolkit

Das *“Java Shared Data Toolkit Version 2.0 (FCS)”* ist eine von Sun entwickelte Sammlung von Java-Objekten und -Funktionen, welche die Entwicklung von interaktiven Kooperationsanwendungen in Java unterstützt und für nicht-kommerzielle Zwecke frei erhältlich ist.

5.2.1 Systembeschreibung

Einführung

Die Hauptfunktion von JSDT besteht in der Verbreitung von Nachrichten und Daten innerhalb einer bestimmten Gruppe von Empfängern. Das System stellt dabei eine höhere, von der darunterliegenden Transportschicht unabhängige Schnittstelle zur Verfügung.

Es werden verschiedene Transportsysteme durch unterschiedliche Implementierungen der Kommunikationsobjekte von JSDT unterstützt. In der Standard-Ausführung werden Implementierungen für die folgenden Technologien mitgeliefert:

- TCP/IP-Sockets (TCP und UDP),
- HTTP-Tunnelled Sockets (zur Überbrückung von Firewalls) und
- LRMP¹ (*lightweight reliable multicast protocol*)

Die Auswahl der zu benutzenden Implementierung erfolgt dynamisch bei Programmstart durch die Übergabe eines Parameters. Innerhalb einer Sitzung kann gleichzeitig nur eines der verfügbaren Systeme verwendet werden. Ein Wechsel der Mechanismen ist nur nach dem Neustart der Anwendung möglich.

1. Die LRMP-Java-Library kann kostenlos aus dem Internet bezogen werden. Die Adresse ist <http://webcanal.inria.fr/lrmp/index.html>

Kommunikationsstruktur

Die Kommunikationsstruktur ist unterteilt in die Komponenten *Session*, *Channel* und *ByteArray*.

Jeder Austausch von Daten findet innerhalb einer **Session** statt, sie definiert den eigentlichen "Ort", an dem sich die beteiligten Partner treffen, um miteinander zu kommunizieren. Jede Session wird von einer Server-Anwendung gestartet und verwaltet. Anhand der Adresse des Servers (bestehend aus IP, Port und verwendeter Implementierungsart) können sich Client-Anwendungen an einer Session beteiligen.

Ein **Channel** in JSDT ist ein Kommunikationskanal, über den beliebige Daten zwischen mehreren Session-Teilnehmern ausgetauscht werden können. Eine Nachricht wird dabei entweder an alle Teilnehmer eines Channels oder nur an einen bestimmten Teilnehmer gesendet. Der Empfang kann sowohl synchron als auch asynchron erfolgen.

Es besteht ebenfalls die Möglichkeit, innerhalb einer Session ein **ByteArray** zu erstellen. Es handelt sich dabei um einen automatisch synchronisierten, verteilten Speicherbereich.

Verwaltungsstruktur

Zusätzlich zu den Kommunikationselementen sind Komponenten und Methoden vorhanden, durch die eine Verwaltung und Regelung des Datenaustauschs stattfinden kann. Dazu gehören *Tokens*, *Manageable Objects* und die Möglichkeit, Clients zu authentifizieren.

Tokens sind Marken innerhalb einer Session, die zur Synchronisation von Aktionen dienen. Diese Marken können beantragt, freigegeben oder weitergegeben werden. Man kann diese Funktionalität nutzen um z.B. eine Rederecht-Verwaltung (*Floor-Control*) zu implementieren oder den Zugriff auf eine Resource (z.B. Ausgabegerät) zu koordinieren. Die Kontrolle des Tokens ist dabei dezentral geregelt, d.h. es gibt keine Möglichkeit, einem Client ein Token zu entziehen.

Die meisten Komponenten der JSDT-Architektur sind optional "**Manageable Objects**", d.h. es können Funktionen installiert werden, die während der Laufzeit entscheiden, ob ein bestimmter Client eine bestimmte privilegierte Aktion auf diesem Objekt durchführen darf. Privilegierte Aktionen umfassen das Erstellen und Entfernen von und das Beitreten zu Sessions, Channels, ByteArrays und Tokens.

Bei der Entscheidung darüber, ob ein Client berechtigt ist, eine bestimmte Aktion auszuführen, findet die Möglichkeit der **Authentifizierung** Anwendung. Es handelt sich dabei um einen simplen Callback-Aufruf durch den eine Challenge-Response-Interaktion zwischen Client und Object Manager stattfindet. Diese Interaktion kann folgendermaßen beschrieben werden:

1. Ein Client führt eine privilegierte Operation aus
2. Der entsprechende Manager wird automatisch aufgerufen, wobei Informationen über das betroffene Objekt, die Art der Operation und den Anfragersteller bereitgestellt werden.
3. Der Manager formuliert eine Anfrage in beliebig definierbarer Form (z.B. "Passwort:") und übergibt diese dem anrufenden Client
4. Der Client antwortet auf diese Anfrage (z.B. mit dem entsprechenden Passwort) mittels einer anfangs definierten Callback-Funktion
5. Der Manager entscheidet anhand der Antwort, ob der Client die Aktion ausführen darf
6. Die Aktion des Clients wird ausgeführt oder abgewiesen

Wie zu erkennen ist, lassen sich hiermit beliebige Formen der Authentifizierung realisieren. Die eben angeführte Beschreibung bezieht sich dabei auf die programmtechnische Implementierung der Authentifizierung, die transparent im Hintergrund abläuft. Ein Benutzer würde sich natürlich in einem Programmablauf nur ein einziges Mal identifizieren (z.B. per Passwort, welches dann gespeichert werden würde).

5.2.2 Bewertung anhand spezieller Anforderungen

Es folgt eine Bewertung von JSMT anhand der Punkte der Anforderungsanalyse, die von diesem System abgedeckt werden.

Anmeldungsunterstützung

Es liegt keine globale Form der Anmeldungsunterstützung vor. Diese kann aber ohne weiteres mit den zur Verfügung stehenden Mitteln realisiert werden, z.B. in Form eines zentralen Verzeichnisseservers.

Teilnehmerverwaltung

Die Teilnehmerverwaltung stellt umfangreiche und selbstdefinierbare Möglichkeiten zur Verfügung, die einzelnen Teilnehmer zu identifizieren.

Dateitransfer

Der Transfer von Dateien ist ohne weiteres durch einen Kommunikationskanal realisierbar. Bei Verwendung der LRMP-Implementierung (siehe Abschnitt , "Einführung") würden hierbei lokal vorhandene Multicastmöglichkeiten ebenfalls ausgenutzt werden.

Mehrpunktконференzen

Laut Aussage von einem der beteiligten Entwickler im Online-Diskussionsforum [JSDTFORUM], wurde das System für Gruppengrößen von mehreren hundert Teilnehmern entworfen. Die Erfahrungsberichte von verschiedenen Anwendern im selbigen Forum lassen auf den problemlosen Betrieb mit mindestens 70 bis zu über 200 Teilnehmern schließen.

5.2.3 Bewertung anhand allgemeiner Anforderungen

Offenheit

JSDT ist als Objektbibliothek konzipiert und bietet durch umfangreiche Schnittstellen einen optimalen Grad an Offenheit. Zudem ist geplant, den Quellcode des Pakets in Zukunft öffentlich zugänglich zu machen.

Plattformen

JSDT ist durchgehend in plattform-unabhängigem Java-Code geschrieben und ist dadurch auf allen Plattformen verwendbar, auf denen eine *Java Virtual Machine* vorhanden ist.

Verfügbarkeit

Das Softwaresystem ist kostenlos per Download erhältlich [JSDTWEB] und darf mit selbstentwickelten Programmen in unveränderter Form verbreitet werden.

Stabilität

JSDT liegt in der Version 2.0 vor, was auf eine akzeptable Entwicklungsreife schließen läßt. Zudem wird mit dem Softwarepaket eine beispielhafte Implementierung eines Whiteboards mitgeliefert, welches bei Tests ohne Probleme lauffähig war.

Im entsprechenden Diskussionsforum für Entwickler [JSDTFORUM] ist dagegen gelegentlich die Rede von Problemen bei der Verwendung des Pakets. Meist lassen sich diese Probleme jedoch auf eine falsche Benutzung der Software zurückführen. Eine endgültige Aussage über die Stabilität des Systems ist jedoch ohne spezielle Tests nicht zu machen.

Die seit der letzten Version 2.0 (FCS) vom Oktober 1999 eingestellte Weiterentwicklung der Software seitens Sun Microsystems, Inc. läßt sich besonders im Hinblick auf die Stabilitätsfrage als einzigen großen Negativpunkt anführen. Aus verschiedenen Veröffentlichungen geht jedoch hervor, daß der Quellcode des Pakets zu einem noch nicht genau bestimmten Zeitpunkt der Öffentlichkeit zugänglich gemacht werden soll, um die Entwicklung als OpenSource-Projekt weiterzuführen.

Sicherheit

Da die Funktionen und Methoden zur Authentifizierung von Teilnehmern durch den Entwickler selbst bereitgestellt werden, ist ein selbstbestimmbares Maß an Sicherheit möglich. In der Socket-Implementierung wird zusätzlich die Verwendung von SSL-Sockets unterstützt, wodurch eine einfache Möglichkeit zur automatischen Verschlüsselung der übertragenen Inhalte bereitgestellt wird.

Firewalltauglichkeit

Der Betrieb innerhalb eines abgeschotteten Netzwerks dürfte problemlos funktionieren, da in diesem Fall keinerlei Kontakt mit der Firewall auftritt. Zur Ermöglichung der Kontaktaufnahme mit einer Session über derartige Grenzen hinweg gibt es die Möglichkeit mittels speziell dafür abgestellter Web-Server die Anfragen über eine HTTP-Verbindung zu "tunneln" und damit Probleme mit Firewall-Konfigurationen zu vermeiden.

5.2.4 Fazit

Das JSJT-System stellt eine Vielzahl von Funktionen zur Verfügung, mit deren Hilfe der Session-Management-Teil des zu entwickelnden Systems größtenteils abgedeckt werden kann. Die Funktionalität ist sehr umfangreich und speziell auf die Notwendigkeit von interaktiv-kooperativen Anwendungen abgestimmt. Die Token-Vergabe erfolgt dezentral, kann aber in modifizierter Form zur Realisierung einer Floor-Control herangezogen werden.

Als einzigen Negativpunkt läßt sich die erwähnte eingestellte Weiterentwicklung des Systems seitens Sun Microsystems nennen. Imentsprechenden Entwicklerforum wird deshalb von der Verwendung von JSJT zur Realisierung von Unternehmenslösungen abgeraten, da ein ausreichender technischer Support nicht gesichert ist.

Ein genauer Termin für die Veröffentlichung des Quellcodes der Software steht zudem nicht fest. Trotzdem deutet die Ankündigung dieses Schritts auf eine positive Zukunft des Systems hin.

5.3 Designempfehlung

Aufgrund der Ergebnisse der Eignungsanalyse von Kapitel 4 ist die Implementierung des Zielsystems auf der Grundlage eines der untersuchten Konferenzsysteme nicht zu empfehlen. Eine Neuentwicklung unter Einbeziehung der in diesem Kapitel betrachteten Einzelkomponenten stellt sich als sinnvolle Alternative dar. Dabei wäre das folgende Vorgehen denkbar:

- Entwicklung von Session-Management- und Kommunikationsmodulen auf Basis von JSJT
- Generische Floor Control entwerfen und implementieren

- Zusätzliche Whiteboard-Komponente entwickeln
- Restliche Komponenten entwickeln/erweitern (Dateitransfer, Präsentationssteuerung, Anmeldungsunterstützung, GUI)
- Zu Gesamtsystem integrieren

Java bietet sich im Hinblick auf eine plattformübergreifende Verfügbarkeit als Implementierungssprache an. Zudem basieren sowohl die Präsentationskomponente, als auch JSDT auf Java.

5.4 Quellenangabe

- NGUY 00** Nguyen-Salamanis, Khan-Loan
Adhoc-Verwendung einer elektronischen Tafel unter Verwendung der Jini-Technologie
Universität Stuttgart, Fakultät Informatik, Studienarbeit Nr. 1778
ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/STUD-1778/
- JSDTFORUM** Diskussionsforum für Entwickler
Archives of JSJT-INTEREST@JAVA.SUN.COM
<http://archives.java.sun.com/archives/jsdt-interest.html>
- JSDTWEB** Sun Microsystems, Inc.
Java (TM) Shared Data Toolkit Home Page
<http://java.sun.com/products/java-media/jsdt/>
- JSDTGUIDE** Rich Burrige, Staff Engineer, Sun Microsystems, Inc.
Java Shared Data Toolkit User Guide
Teil der JSJT-Distribution

6 Entwurf einer Generische Floor Control

Um den konkurrierenden Zugriff mehrerer Teilnehmer auf das SMARTBoard, beziehungsweise in einer allgemeiner Betrachtung auf einen beliebigen Dienst zu koordinieren, ist ein Mechanismus zur Vergabe des “Rederechts” notwendig. Im folgenden wird ein derartiger Mechanismus in einer möglichst allgemeinen, anwendungsunabhängigen Form beschrieben und entworfen.

6.1 Definition “Floor Control”

Unter einer *Floor Control* (engl. für “Rederecht auf einer Konferenz”) versteht man im Zusammenhang mit Konferenzsystemen einen Mechanismus, durch den festgelegt wird, welcher Teilnehmer einer Sitzung das Recht hat, einen bestimmten Dienst zu nutzen.

6.2 Grundbegriffe

Aus der vorhergehenden Definition lassen sich Grundbegriffe ableiten, deren Modellierung eine Voraussetzung für die Entwicklung der Floor Control darstellt.

6.2.1 Dienst

Bei einem Dienst handelt es sich um ein Programmobjekt, dessen Ein- und Ausgaben durch eine Floor Control koordiniert werden sollen. Der Dienst *bedient* sich dabei der Funktionen der Floor Control, weshalb innerhalb des Floor-Mechanismus keine Modellierung des Dienstes an sich stattfinden muß. Ein Dienst entspricht vielmehr einer konkreten Instanziierung der abstrakten Floor Control-Klasse.

Zum Zeitpunkt der Instanziierung einer Floor Control wird außerdem die Floor-Größe festgelegt. Diese Größe bestimmt, wieviele Teilnehmer gleichzeitig auf den Dienst zugreifen dürfen, bzw. ob der Zugriff exklusiv stattfinden soll (Floor-Größe=1).

Um den Dienst beziehungsweise die entsprechende Floor Control für die Teilnehmer einer Sitzung identifizierbar zu machen, muß zudem ein Name angegeben werden. Dieser Name wird von der Teilnehmerseite für die Verbindungsaufnahme mit der Floor Control verwendet.

6.2.2 Teilnehmer

Die Definition des Teilnehmer-Begriffs ist nicht Gegenstand der Floor Control. Sie wird vielmehr aus einer anderen Komponente (Teilnehmer-Datenbank) importiert.

Der Teilnehmerbegriff muß dabei die folgenden Anforderungen erfüllen:

- Jeder Teilnehmer kann eindeutig identifiziert werden
- Jedem Teilnehmer ist ein beschreibender Name zugeordnet (entweder Name des Teilnehmers oder Pseudonym)
- Jedem Teilnehmer ist eine Priorität in Form einer Ganzzahl zugeordnet

Zusätzlich zur *statischen* Priorität eines Teilnehmers kann bei Systemstart ein Teilnehmer bestimmt werden, welcher den Vorsitz übernimmt. Dadurch erhält dieser eine dynamische Priorität, welche größer ist als die aller anderen Teilnehmer.

6.2.3 Recht

Unter einem "Recht" versteht man in diesem Zusammenhang einen einfachen booleschen Wert, der von der zu kontrollierenden Anwendung benutzt und interpretiert wird. Die Hauptaufgabe der Floor Control ist eine dynamische, von den Teilnehmern beeinflussbare Zuordnung von Recht zu Teilnehmer.

6.2.4 Mechanismus

Der Mechanismus legt fest, nach welchen Regeln die Zuordnung von Recht zu Teilnehmer erfolgt. Man unterscheidet die folgenden Mechanismen [BURG97]:

- Implizit (*designation mode*): eine Zentrale Instanz vergibt und entzieht das Rederecht
- Explizit (*baton mode*): das Rederecht wird durch den aktuellen Besitzer abgegeben.

Im *baton mode* kann die Vergabestrategie (Ziel der Weitergabe) je nach Anforderung unterschiedlich gewählt werden:

1. Nur Rückgabe möglich (*first come, first served*-Reihenfolge)
2. Rückgabe oder Weitergabe an einen bestimmten Teilnehmer möglich.

6.2.5 Rollen

Eine Rolle ist die dynamische Eigenschaft eines Teilnehmers, die bestimmt, auf welche Operationen der Floor Control dieser Teilnehmer Zugriff hat. Man unterscheidet zwei Rollen:

1. **Vorsitzender.** Ein Teilnehmer in dieser Rolle kann das Rederecht vergeben und entziehen sowie die Vergabestrategie (siehe Abschnitt 6.2.4, "Mechanismus") einstellen. Zudem besitzt ein Vorsitzender alle Rechte, über die auch ein Zuhörer verfügt.
2. **Zuhörer.** Ein Zuhörer kann die folgenden Aktionen ausführen: Rederecht beantragen, Rederecht zurückgeben, Rederecht an einen bestimmten Teilnehmer weitergeben.

6.3 Abstimmungen

Wie in Abschnitt 6.2.5, "Rollen" ersichtlich ist, gibt es innerhalb des Rederecht-Systems bestimmte Operationen, die nur durch den Vorsitz ausgeführt werden können. Dies umfaßt z.B. die Möglichkeit, einem Teilnehmer das Rederecht zu entziehen. Gleichzeitig gibt es aber auch in den *eigentlichen* Diensten selbst Aktionen die einer besonderen Befugnis bedürfen, um chaotische Situationen zu vermeiden (zum Beispiel ist es sinnvoll, die Funktionen des Session Management wie das Entfernen von Teilnehmern durch eine derartige Befugnis vor Mißbrauch zu schützen).

Eine Möglichkeit, diese "Befugnisfrage" zu klären, wäre beispielsweise das Festlegen eines Vorsitzenden zur Startzeit durch eine zentrale Stelle. Mit Blick auf die unterschiedlichen denkbaren Anwendungsszenarien des zu entwickelnden Systems (z.B. Vorlesungen, Gruppenübungen mit und ohne Gruppenleiter) erweist sich diese Lösung jedoch schnell als zu unflexibel.

Eine Alternative dazu stellt die Verwendung eines "Abstimmungsmechanismus" dar, mit dessen Hilfe die Befugnisfrage während einer Sitzung durch die Beteiligung der Sitzungsteilnehmer geklärt werden kann. Der Vorteil dieser Vorgehensweise wird besonders bei *vorsitzlosen* Sitzungen offensichtlich. In derartigen Szenarien sind Situationen denkbar, in denen es nicht ausreicht, die Regelung der Rechtevergabe ausschließlich anhand eines sozialen, nicht systemgestützten Protokolls vorzunehmen. Ein Beispiel hierfür wäre ein uneinsichtiger Teilnehmer einer ungeleiteten Gruppenübung, welcher es gegen den Willen der Mehrheit ablehnt, das Rederecht abzugeben.

6.3.1 Beispielhafte Szenarien

Die folgenden Szenarien sind im Projektzusammenhang denkbar und sollten durch einen entsprechenden Mechanismus abgedeckt werden:

1. **Vortrag.** Ein einzelner Teilnehmer bestimmt über die Vergabe des Rederechts und über andere Operationen, die einer besonderen Berechtigung erfordern

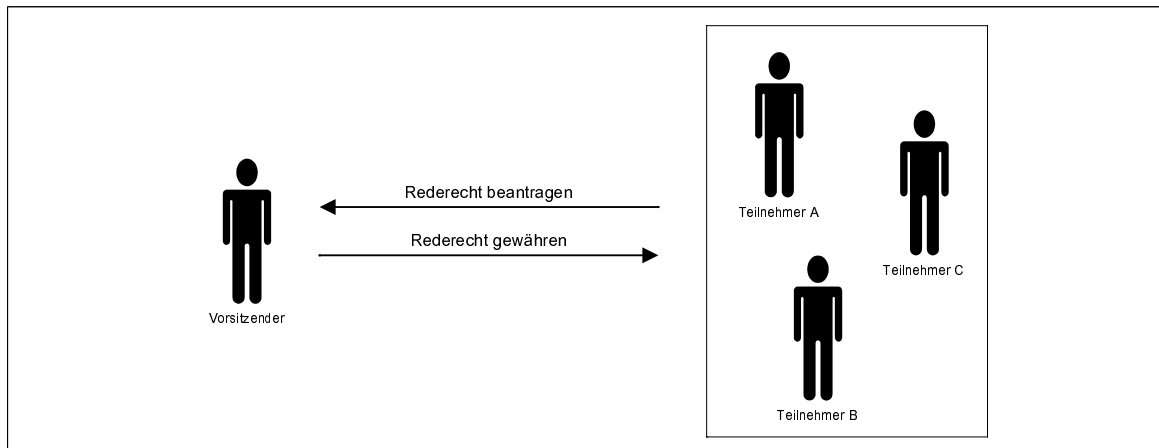


Abbildung 6-1: Rederechtsvergabe durch Vorsitzenden

2. **Gruppenübung.** Die Vergabe des Rederechts erfolgt nach dem *first come, first served* Prinzip ohne zentrale Kontrolle. Bei Bedarf sollte auf einzelne Vorsitzfunktionen zurückgegriffen werden können.

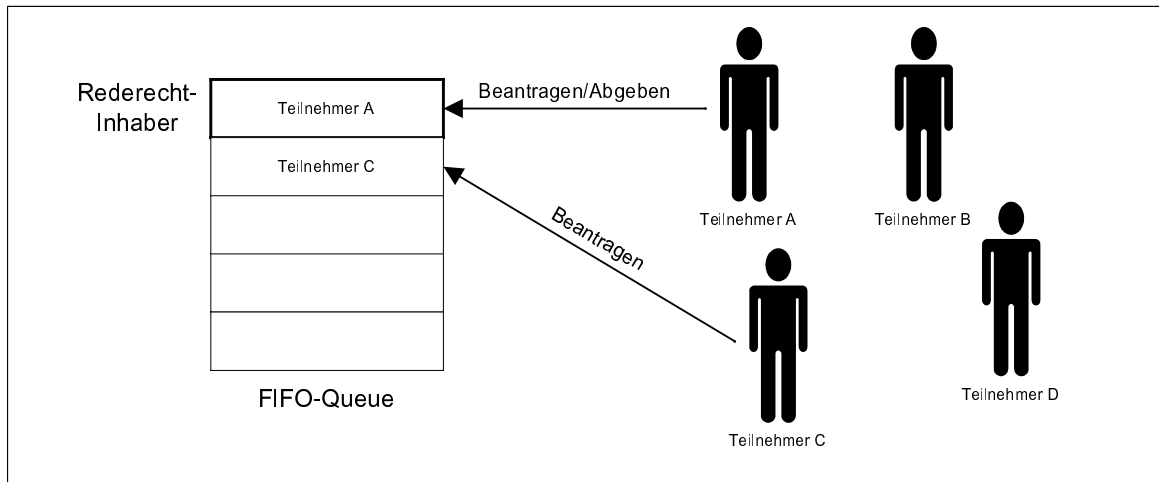


Abbildung 6-2: Rederechtsvergabe nach dem first-come-first-served-Prinzip

3. **Geleitete Gruppenübung.** Die Mehrheit der Teilnehmer wählt einen Übungsleiter, welcher den Vorsitz übernimmt. Bei Bedarf sollte diese Vorsitzstellung widerrufen werden können.

6.3.2 Stimmen und Stimmgewichtung

Um der besonderen Stellung des Vortragenden innerhalb einer Sitzung gerecht zu werden, erfolgt eine Gewichtung der Stimmen. Dabei gelten die folgenden Regeln:

- Jeder Teilnehmer hat eine Stimme
- Das Gewicht einer Stimme hängt von der Priorität des Teilnehmers ab (siehe Abschnitt 6.2.2, "Teilnehmer")

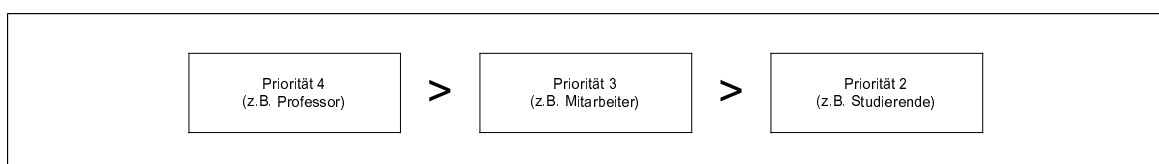


Abbildung 6-3: Beispielhafte Prioritätsreihenfolge

- Bei Systemstart kann einem bestimmten Teilnehmer eine "maximale Priorität" zugewiesen werden, wodurch dieser explizit die Rolle des Vorsitzenden übernimmt.
- Eine einzelne Stimme mit höherer Priorität zählt mehr als alle Stimmen mit niedriger Priorität

6.3.3 Gegenstand von Abstimmungen

Aus der einleitenden Ausführung über die “Befugnisfrage” lassen sich die folgenden Gegenstände von Abstimmungen ableiten:

- **Vorsitzfrage:** Wer soll den Vorsitz einnehmen?
- **Floor Control Antrag:** Soll eine bestimmte Funktion der Floor Control ausgeführt werden? (z.B. erzwungener Entzug des Rederechts)
- **Dienstspezifischer Antrag:** Soll eine bestimmte Funktion eines Dienstes ausgeführt werden? (z.B. Entfernen von Teilnehmern)

6.3.4 Optionen bei einer Abstimmung

Ein Teilnehmer reagiert auf eine Abstimmung, indem er seine Stimme vergibt. Die möglichen Ziele dieser Vergabe sind im einzelnen:

- Ein anderer Teilnehmer
- Zustimmung/Ablehnung einer beantragten Floor Control-Aktion
- Zustimmung/Ablehnung einer beantragten dienstspezifischen Aktion
- Enthaltung

Die Stimmvergabe kann dabei jederzeit widerrufen werden. Zu beachten ist außerdem, daß ein Teilnehmer seine Stimme nur **einer** dieser Alternativen gleichzeitig geben kann. Während diese Einschränkung keine nennbare Reduktion des Leistungsumfangs der Floor Control darstellt, dient sie als vorbeugende Maßnahme, eine übermäßig komplizierten Bedienung der Floor Control zu verhindern.

Stimmvergabe an andere Teilnehmer

Beispiel: Teilnehmer A vergibt seine Stimme an Teilnehmer B

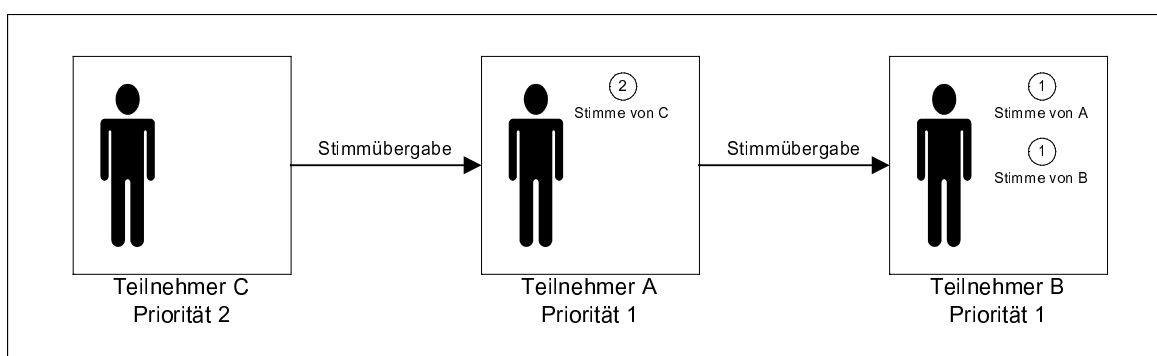


Abbildung 6-4: Prinzip der Stimmübergabe

Teilnehmer B verfügt dadurch bei Abstimmungen über die Summe der an ihn vergebenen Stimmen (inklusive deren Gewichtung). Diese Stimmvergabe funktioniert nur einstufig, d.h. ein Teilnehmer kann nur seine eigene Stimme weitergeben, nicht aber die Stimmen, die er zuvor von anderen Teilnehmern erhalten hat.

Erhält ein Teilnehmer mehr als die Hälfte der gewichteten Stimmen, so nimmt dieser Teilnehmer automatisch die Vorsitzfunktion ein, d.h. er bekommt das Recht, die entsprechenden Operationen der Floor Control auszuführen.

Dies gilt ebenfalls für den Fall, daß ein Teilnehmer als einziger Teilnehmer in einer Sitzung eine höhere Priorität besitzt, als alle anderen Teilnehmer (normaler Vortragsfall).

Antwort auf eine beantragte Floor Control-Funktion

Dieser Mechanismus findet dann Anwendung, wenn es keinen Vorsitz gibt, aber dennoch auf spezielle Funktionen zurückgegriffen werden soll, die nur für den Vorsitz verfügbar sind.

Beispiel: in einer Sitzung ohne Vorsitz soll einem Redner das Rederecht entzogen werden. Ein Teilnehmer beantragt diese Funktion. Alle anderen Teilnehmer werden benachrichtigt und können diesem Antrag ihre Stimme geben. Erreicht der Antrag mehr als die Hälfte der gewichteten Stimmen, so wird die Aktion ausgeführt.

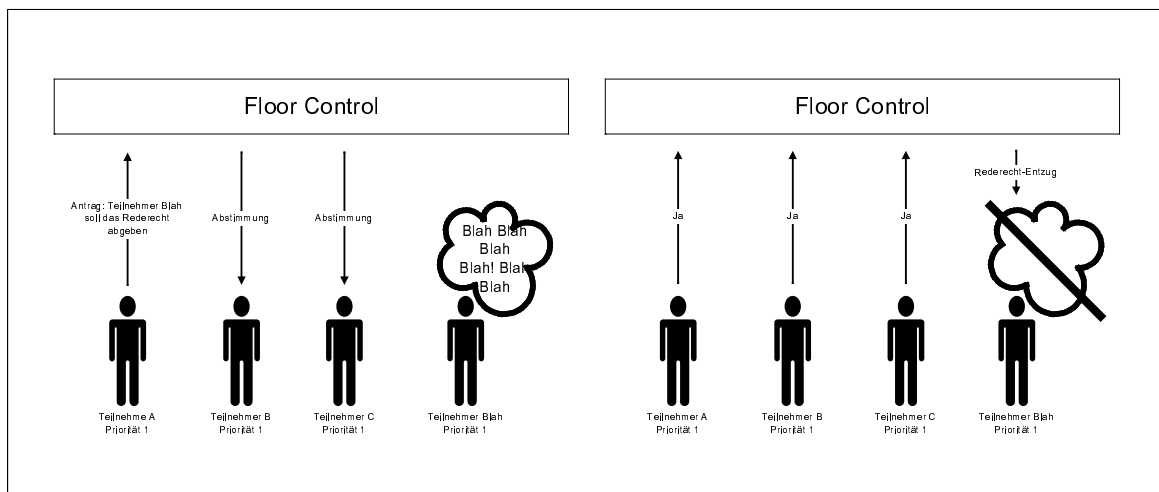


Abbildung 6-5: Beispiel für einen Abstimmungsantrag

Andererseits kann ein Antrag auch negativ beantwortet werden. Er wird abgelehnt, wenn mehr als die Hälfte der gewichteten Stimmen negativ auf diesen Antrag antworten.

Anträge an sich sind unbefristet, d.h. solange keine Mehrheit dafür oder dagegen stimmt, steht der Antrag weiterhin zur Abstimmung. Ein Antrag kann allerdings auch zurückgenommen werden, indem kein Teilnehmer, einschließlich des Antragsstellers, weder befürwortend noch ablehnend über den Antrag abstimmt (die Stimme des Antragsstellers wird zu Beginn automatisch auf der Seite der Befürworter verbucht).

Antwort auf eine beantragte dienstspezifische Funktion

Die Möglichkeit, über Anträge abzustimmen, sollte zudem auch den einzelnen Diensten zugänglich gemacht werden.

Beispiel: In einer Sitzung ohne Vorsitz soll ein Teilnehmer aus der Sitzung ausgeschlossen werden. Ein anderer Teilnehmer beantragt diese Aktion durch die Schnittstelle des entsprechenden Dienstes (in diesem Fall Session Management). Der Dienst leitet diesen Antrag an die Floor Control weiter.

Aus Sicht der Floor Control handelt es sich hierbei um eine generische Aktion, das heißt sie verfügt nicht über Wissen über den inhaltlichen Gegenstand der Aktion (hier: Entfernen eines Teilnehmers). Damit die Teilnehmer dennoch wissen, worüber sie abstimmen sollen, muß der Dienst eine allgemeine Identifizierung der Aktion angeben, z.B. eine Zeichenkette mit Inhalt "Entfernen von Teilnehmer X". Die Teilnehmer können so diesem Antrag ihre Stimme geben.

Hat eine Mehrheit positiv oder negativ über die Aktion abgestimmt oder wurde der Antrag zurückgezogen, so wird der entsprechende Dienst darüber verständigt, welcher wiederum die eigentliche Aktion ausführt oder nicht.

Stimmenthaltung

Jeder Teilnehmer kann sich bei allen Abstimmungen seiner Stimme enthalten, indem er diese aus dem "aktiven Stimmenpool" entfernt. Unter dem Aspekt der Konsistenz der Bedienung bietet es sich an, diese Aktion dadurch zu realisieren, daß ein Teilnehmer seine Stimme an einen *speziellen Antrag* "Enthaltung" übergibt. Dadurch würden alle stimmbezogenen Aktionen (Delegation, Abstimmung, Enthaltung) innerhalb eines einzigen gedanklichen Modells stattfinden (nämlich dem der Stimmvergabe wahlweise an Personen oder an Anträge).

Bei dieser Art der Enthaltung wird die entsprechende Stimmgewichtung bei der Bestimmung der Mehrheitsgrenze nicht berücksichtigt. Es besteht ein Unterschied zwischen der aktiven Enthaltung und der passiven Nichtbeantwortung von Abstimmungen.

Beispiel: wird in einer Sitzung mit 4 Teilnehmern ein Antrag gestellt, so ist dieser Antrag erst dann entschieden, wenn 3 Teilnehmer dafür oder dagegen gestimmt haben. Wenn sich einer dieser Teilnehmer seiner Stimme enthält, so wird ein Antrag bereits mit 2 Stimmen entschieden.

Diese Art der Stimmenthaltung findet zum Beispiel dann Anwendung, wenn ein Teilnehmer über eine höhere Priorität verfügt als alle anderen Teilnehmer, aber dennoch Entscheidungen über die Floor Control der Mehrheit der Teilnehmer überlassen möchte.

6.3.5 Abschätzung der Kompliziertheit des Verfahrens

Eine wichtige Randbedingung bei der Entwicklung der Floor Control besteht darin, die Benutzung des Systems für Anwender nicht unnötigkompliziert zu gestalten und insbesondere die Aufmerksamkeit der Benutzer nicht vom eigentlichen Dienst abzulenken.

Durch den zuvor beschriebenen Abstimmungsmechanismus zur Vergabe des Rederechts wird sowohl diese Randbedingung als auch die Forderung nach Flexibilität prinzipiell erfüllt. Die Anwenderakzeptanz hängt jedoch zum Großteil auch von der Gestaltung der Benutzungsoberfläche ab, die Zugriff auf die erwähnte Funktionalität gewährt. Unter diesem Gesichtspunkt empfiehlt sich die Staffelung der Eingabelemente in zwei Gruppen:

1. Elemente, die sich direkt auf das Rederecht beziehen. Hierzu gehören: Schaltfläche zum Beantragen und Abgeben des Rederechts sowie Informationen darüber, wer momentan Inhaber des Rederechts ist. Zudem sollte eine weitere Anzeige darüber Aufschluß geben, ob und wieviele Abstimmungsanträge ausstehen.
2. Alle übrigen Elemente, die für die Benutzung der verfügbaren Funktionen notwendig sind.

Es wäre hierbei wünschenswert, die erste Elementgruppe zusammen mit der eigentlichen Dienst-Oberfläche dauerhaft einzublenden, während die zweite Gruppe nur bei Bedarf sichtbar sein sollte.

6.4 Quellenangabe

BURG 97 Cora Burger
Groupware
dpunkt-Verlag ISBN 3-920993-60-8

7 Implementierung der Floor Control

Dieses Kapitel befaßt sich mit der Implementierung der in Kapitel 6, “Entwurf einer Generische Floor Control”, entworfenen Komponente zur Verwaltung des Rederechts.

7.1 Kommunikationssystem

Als Basis der Implementierung ist ein Kommunikationssystem notwendig, um Rederechtsanträge unter den Teilnehmern austauschen zu können. Hierzu wird das im Rahmen einer anderen Arbeit [OBER 01] entwickelte Comm-System¹ verwendet.

Dieses Kommunikationssystem ermöglicht die Verbreitung von Nachrichten in Form beliebiger Java-Objekte innerhalb einer Gruppe von Teilnehmern. Dabei wird sowohl das Authentifizieren von Teilnehmern, als auch die Verwendung eines Multicast-basierten Protokolls zur effizienteren Ausnutzung der vorhandenen Bandbreite verwendet. Das System ist in einer Client-Server-Architektur mit den folgenden Komponenten angelegt:

- CommServer (siehe Abschnitt 10.3, “CommServer-API”). Hierbei handelt es sich um die zentrale Stelle des Systems.
- CommClient (siehe Abschnitt 10.4, “CommClient-API”). Eine Instanz dieser Client-Klasse ermöglicht die Verbindungsaufnahme mit einem Server.
- CommChannel (siehe Abschnitt 10.5, “CommChannel-API”). Das Channel-Objekt ermöglicht die eigentliche Kommunikation. Auf einem Server können unterschiedliche Kanäle existieren, die durch Namen identifiziert werden. Es stehen Methoden für die synchrone sowie für die asynchrone Kommunikation zur Verfügung.

7.2 Benutzerdatenbank

In der Benutzerdatenbank sind alle statischen Teilnehmerinformationen gespeichert. Diese Datenbank wird unter anderem während des Authentifizierens eines Teilnehmers zum Zeitpunkt der Anmeldung an das Kommunikationssystem benutzt. Um bei ungeleiteten Sitzungen die Frage des Vorsitzes zu regeln, sind in der Benutzerdatenbank ebenfalls Informationen über die Priorität eines Teilnehmers gespeichert (siehe Abschnitt 6.3.2, “Stimmen und Stimmengewichtung”). Die Implementierung der Benutzerdatenbank erfolgt im Rahmen einer eigenen Arbeit [SOMM01].

1. Das erwähnte Kommunikationssystem entstand im Rahmen dieser Studienarbeit, wird jedoch erst in einer folgenden Arbeit, die sich mit der Integration des Gesamtsystems auseinandersetzt, behandelt.

7.3 Programmierschnittstellen

Aus dem Inhalt des Entwurfs in Kapitel 6 ergibt sich direkt der Umfang der öffentlichen Floor-Control-Schnittstellen, durch welche Anwendungsprogrammierer Zugriff auf die Funktionalität dieser Komponente erhalten. Es folgt eine Beschreibung dieser Schnittstellen:

7.3.1 Server

Während der Initialisierung des Floor-Control Servers besteht die Möglichkeit, einem Teilnehmer die Rolle des Vorsitzenden zuzuweisen. Falls kein Vorsitzender explizit bestimmt wird, erfolgt die Vergabe der Vorsitz-Rechte dynamisch anhand der Prioritäten der anwesenden Teilnehmer.

Für einen Dienst-Server ist hauptsächlich die aktuelle Zugriffs-Berechtigung eines Teilnehmers interessant. Dementsprechend stellt die Server-Schnittstelle Funktionen zur Verfügung, um die Zugriffsfrage für einen gegebenen Teilnehmer zu beantworten, bzw. um die Liste der momentan zugriffsberechtigten Teilnehmer abzufragen.

Desweiteren wird die folgende Funktionalität bereitgestellt (vgl. Abschnitt 10.6, "FloorServer-API"):

- Abfrage des gesamten Floor-Zustandes (beteiligte Teilnehmer, Teilnehmer, die einen Antrag auf Rederecht gestellt haben, Teilnehmer, die das Rederecht besitzen).
- Abfrage und Änderung der Floor-Einstellungen (maximale Anzahl der gleichzeitig zugriffsberechtigten Teilnehmer, Festlegung, ob bei einer ungeleiteten Sitzung das Rederecht von Teilnehmern weitergegeben werden darf).
- Abfrage der momentanen Abstimmungssituation (siehe Abschnitt 6.3, "Abstimmungen") sowie Erstellung von neuen, anwendungsspezifischen Abstimmungen.
- Registrierung eines Listener-Objekts, das asynchron über Änderungen im Zustand der Floor-Control benachrichtigt wird.

7.3.2 Client

Die clientseitige Schnittstelle der Floor-Control umfaßt im wesentlichen die Funktionalität des Floor-Control-Servers. Zusätzlich werden die folgenden Aktionen unterstützt (vgl. Abschnitt 10.7, "FloorClient-API"):

- Antrag auf Rederecht stellen und zurückziehen, Rederecht zurückgeben
- Stimmabgabe für einen Teilnehmer oder eine Abstimmung oder Stimmenthaltung
- Vorsitz-Funktionen. Diese umfassen das Gewähren, das Entziehen und das Austauschen des Rederechts. Zudem besteht die Möglichkeit, den Namen des aktuellen Vorsitzenden zu erfragen. Teilnehmer, die den Vorsitz nicht innehaben, können Vorsitz-Funktionen beantragen, über die per Abstimmung entschieden wird.

7.4 Systemstruktur

Gegenstand dieses Abschnitts ist die Übersicht über die komponenteninterne Systemstruktur im Hinblick auf den Datenfluß zwischen den einzelnen Objekten.

7.4.1 Server

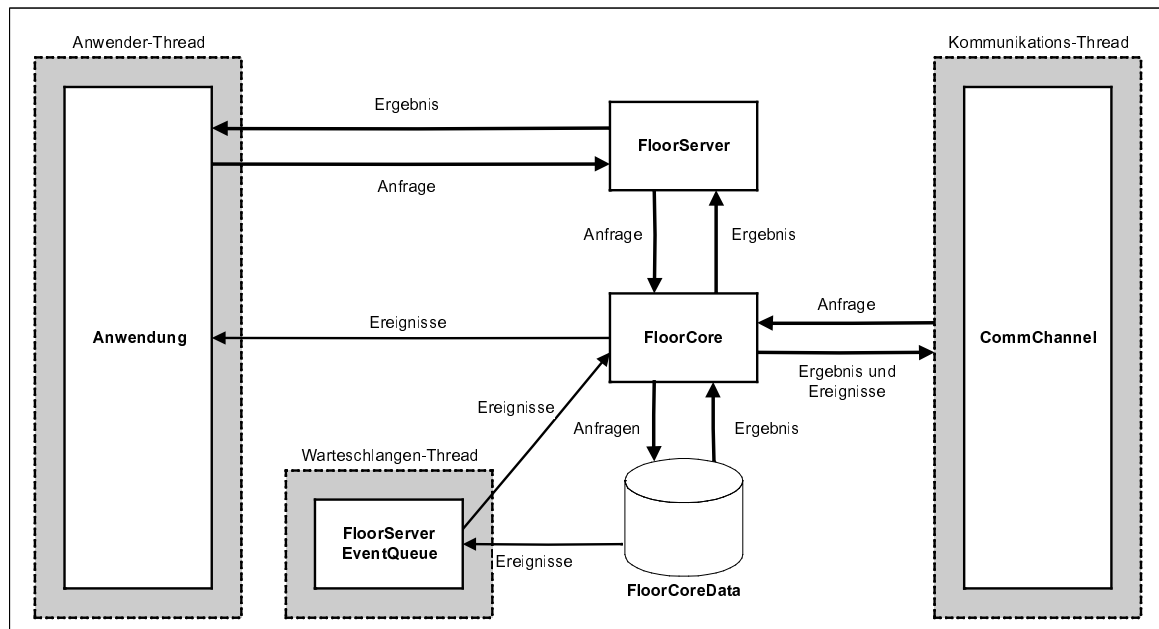


Abbildung 7-1: Datenfluß innerhalb des Floor-Control-Servers

Die Objekte des Servers lassen sich ihrer Funktion entsprechend einteilen in solche, die die interne Daten- und Ereignisverwaltung übernehmen (FloorCoreData und FloorServerEventQueue) und jene, die als Schnittstelle zu lokalen beziehungsweise entfernten Anwendungen dienen (FloorServer und FloorCore, vgl. Abbildung 7-1).

- **FloorServer:** Bereitstellung einer Schnittstelle für die serverseitige Anwendung.
- **FloorCore:** Bündelung der Anfragen von lokalen und entfernten Teilnehmern, Implementierung des Berechtigungstests für die Benutzung von Vorsitz-Funktionen, Ansteuerung der internen Datenbank.
- **FloorCoreData:** Verwaltung der Floor-Control-Daten, Generierung von Ereignissen bei Zustandsänderungen.
- **FloorServerEventQueue:** Verteilung von Floor-Ereignissen in einem eigenen Thread. Diese Warteschlange ist notwendig, um die Generierungsreihenfolge der Ereignisse bei der Auslieferung zu wahren. Des Weiteren verhindert sie das Blockieren des gesamten Systems im Falle von langsamen Listener- oder Kommunikationssystem-Funktionen.

7.4.2 Client

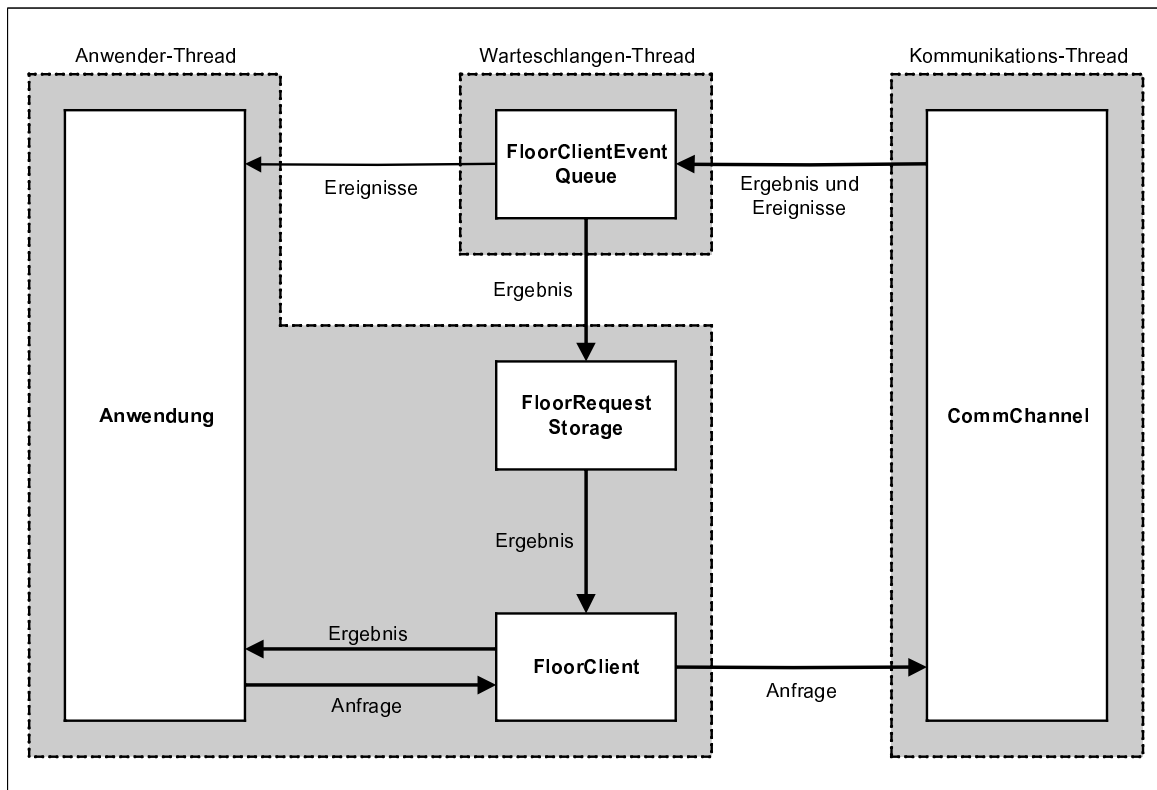


Abbildung 7-2: Datenfluß innerhalb des Floor-Control-Clients

Die Hauptfunktionalität der Client-Komponente besteht in der Bereitstellung einer Schnittstelle zum entfernten FloorServer.

- **FloorClient:** Leitet Anfragen der Anwendung an das Kommunikationssystem weiter.
- **FloorClientEventQueue:** Nimmt Floor-Ereignisse und Ergebnisse aus Client-Anfragen über das Kommunikationssystem entgegen und verteilt diese. Diese Warteschlange ist wiederum als eigener Thread implementiert, um die Blockierung des gesamten Systems bei langsamen Listener-Funktionen zu vermeiden. Da über den Kanal sowohl Ereignisse, als auch die Ergebnisse von Anfragen asynchron eintreffen, erfolgt die Synchronisation zwischen Warteschlangen- und Anwendungsthread über ein eigenes Objekt, der FloorRequestStorage.
- **FloorRequestStorage:** Stellt die Möglichkeit bereit, auf eingehende Anfrage-Ergebnisse zu warten.

7.5 Quellenangabe

- OBER 01** Peter Oberparleiter
Eine Architektur zur gemeinsamen Nutzung von Anwendungen in der Lehre mit prototypischer Realisierung von elektronischer Tafel und Leinwand
Universität Stuttgart, Fakultät Informatik, Diplomarbeit, 2001
- SOMM 01** Marcus Sommer
Prototypische Erstellung einer Sitzungsverwaltung für den Einsatz in Lehrveranstaltungen
Universität Stuttgart, Fakultät Informatik, Studienarbeit, 2001

8 Test und Bewertung

8.1 Test

Um die Grundfunktionalität der implementierten Floor-Control-Komponente zu testen, wurde eine Beispielanwendung entwickelt, die ein einfaches Chat-System mit Rederecht-Vergabe realisiert.

8.1.1 Chat-Server

Der Chat-Server bietet dem Benutzer die Möglichkeit, die Zahl der gleichzeitig redeberechtigten Teilnehmer (Floor-Größe) einzustellen sowie das System herunterzufahren (siehe Abbildung 8-1).

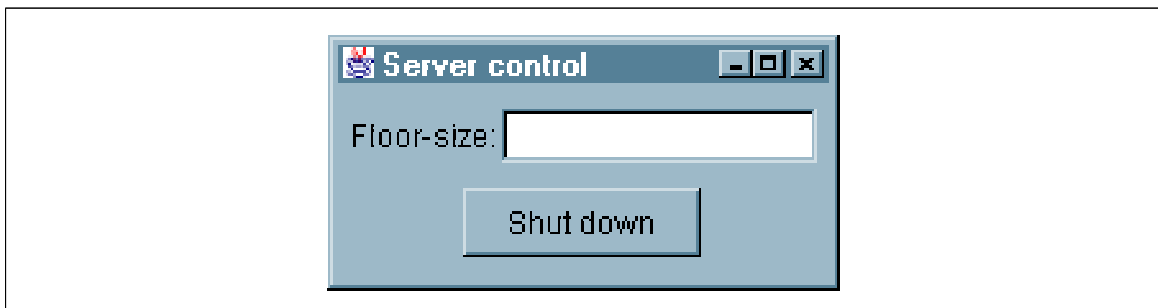


Abbildung 8-1: Benutzungsoberfläche des Chat-Servers

```

while (running)
{
    try
    {
        CommMessage msg = channel.receive(500);

        if (floor.isFloorOwner(msg.getSender()))
        {
            channel.sendToOthers( msg.getSender()+ ": " +
                msg.getAsString());
        }
    }
    catch (CommTimeOutException e)
    {
    }
    catch (Exception e)
    {
        /* Every other exception is an error */
        running = false;
    }
}

```

Abbildung 8-2: Auszug aus dem Programmcode des Chat-Servers

Der Auszug aus dem Programmcode des Chat-Servers in Abbildung 8-2 macht deutlich, daß die Benutzung der Grundfunktion der Floor-Control mit minimalem programmiertechnischem Aufwand möglich ist.

8.1.2 Chat-Client

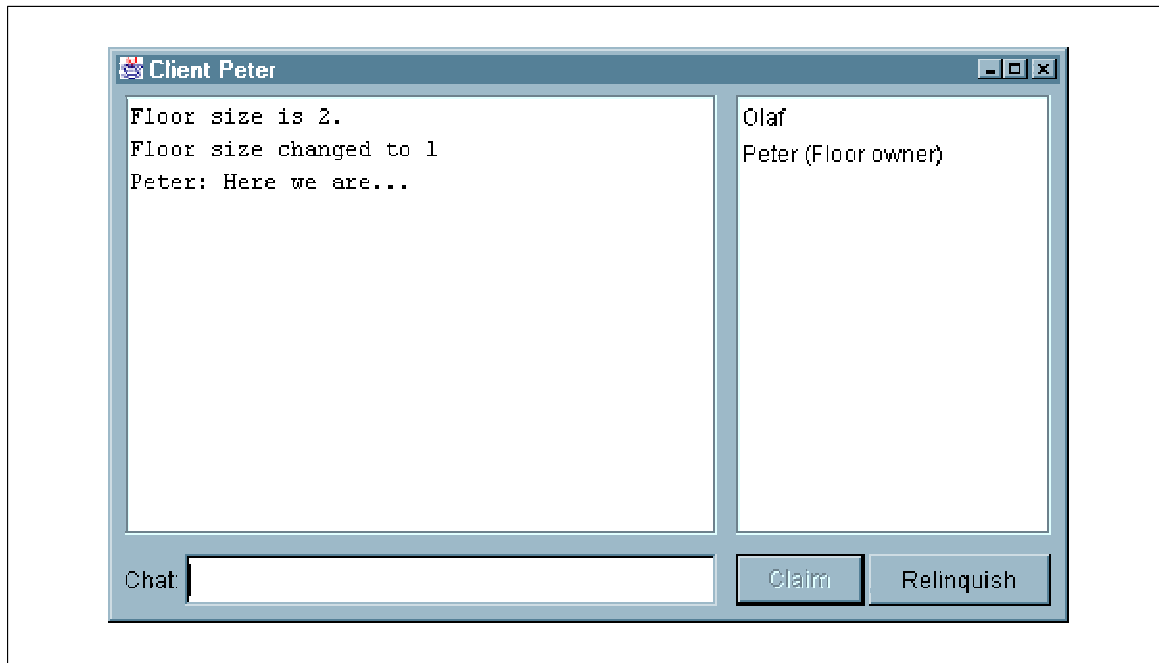


Abbildung 8-3: Benutzungsoberfläche des Chat-Clients

```
// This method is called when the relinquish-button was pressed
private void relinquishButtonActionPerformed(ActionEvent evt)
{
    try
    { floor.relinquishFloor(); }
    catch (FloorException e) {}
}

// This method is called when the claim-button was pressed
private void claimButtonActionPerformed(ActionEvent evt)
{
    try
    { floor.claimFloor(); }
    catch (SASCIAException e) {}
}
```

Abbildung 8-4: Auszug aus dem Programmcode des Chat-Clients

Die Benutzungsoberfläche des Chat-Clients (siehe Abbildung 8-3) beinhaltet eine Darstellungsfläche für übermittelte Nachrichten, sowie ein Eingabeschaltfeld für eigene Mitteilungen. Des weiteren werden in einer Liste die Namen aller an einer Sitzung beteiligten Teilnehmer aufgeführt, wobei die Rederecht-

inhaber besonders gekennzeichnet sind. Mittels zweier weiterer Schaltflächen besteht die Möglichkeit, das Rederecht zu beantragen oder abzugeben. Das Erstellen einer Mitteilung ist nur bei gewährtem Rederecht möglich.

Abbildung 8-4 demonstriert die Verwendung der beiden grundlegenden Methoden der Floor-Client-API.

8.1.3 Testergebnisse

- Die Grundfunktionalität der Rederechtvergabe wird durch die Floor-Control-Komponente korrekt erfüllt.
- Die Antwortzeiten des Systems sind ausreichend - die Zeitdauer zwischen Antragstellung und -gewährung blieb immer unter einer Sekunde.

8.2 Bewertung

- Die Implementierung der Floor-Control erwies sich im Hinblick auf die notwendige Grundfunktionalität als korrekt. Über die erweiterte Funktionalität der Abstimmungen gibt es noch keine Erkenntnisse.
- Wie aus dem Testergebnis ersichtlich, sind die Antwortzeiten des Systems ausreichend. Die Implementierung läßt hierbei noch Spielraum für Optimierungen. So wurde z.B. die serverseitige Ereignisgenerierung noch relativ ineffizient in einem Verfahren mit zur Teilnehmerzahl quadratischem Aufwand implementiert.

9 Zusammenfassung und Ausblick

9.1 Zusammenfassung

Zu Beginn dieser Arbeit wurde ein System zur ferngesteuerten Bildschirmpräsentation (das sog. "Zielsystem") definiert. Es folgte die Präzisierung der Funktionalität sowie die Erstellung einer Liste von Anforderungen an Systeme, die das Zielsystem implementieren sollen. Nachdem sich herausstellte, daß diese Anforderungen zum Großteil denen eines Konferenzsystems entsprachen, wurden 4 solcher Systeme auf ihre Eignung entsprechend der definierten Anforderungen untersucht.

Als Ergebnis der Untersuchung ergab sich, daß keines der Systeme allen notwendigen Anforderungen gerecht wurde. Daraufhin wurde beschlossen, Komponenten zu untersuchen, die für eine Neuentwicklung verwendet werden könnten. Das positive Ergebnis dieser Analyse führte zu einer Designempfehlung zugunsten einer Neuentwicklung auf Basis der untersuchten Komponenten.

Nach Abschluß der anfänglichen Recherche- und Untersuchungsphase lag der Schwerpunkt der weiteren Arbeit auf dem Entwurf und der Implementierung einer Komponente, die eine Rederechtverwaltung realisiert. Die entwickelte Komponente wurde anschließend einem Test unterzogen, an dessen Ende ein positives Ergebnis in Hinblick auf Funktionalität und Korrektheit der geforderten Grundfunktionalität stand.

9.2 Ausblick

Eine weitere Arbeit zu diesem Thema [OBER 01] wird sich mit der Integration der Floor-Control und anderer Komponenten mit dem Ziel der Erstellung eines Prototyps des Zielsystems befassen. Zum Zeitpunkt des Abschlusses dieser Arbeit befinden sich zwei weitere notwendige Komponenten in der Entwicklung. Dies ist zum einen eine Protokollierungsdatenbank [SCH 01] sowie eine Komponente zur Sitzungsverwaltung [SOMM 01]. Gleichzeitig wird an weiteren Einsatzgebieten des Zielsystems gearbeitet [BAI 01]

Sobald diese Entwicklungen abgeschlossen und zu einem Komplettsystem integriert sind, kann dieses testweise in Vorlesungen eingesetzt werden. Es ist zu erwarten, daß sich die Ergebnisse dieser Tests auf die weitere Entwicklung des Systems auswirken werden.

9.3 Quellenangabe

- OBER 01** Peter Oberparleiter
Eine Architektur zur gemeinsamen Nutzung von Anwendungen in der Lehre mit prototypischer Realisierung von elektronischer Tafel und Leinwand
Universität Stuttgart, Fakultät Informatik, Diplomarbeit, 2001
- SCH 01** Andreas Schmid
Prototypische Erstellung einer Protokollierung für den Einsatz in Lehrveranstaltungen
Universität Stuttgart, Fakultät Informatik, Studienarbeit, 2001
- SOMM 01** Marcus Sommer
Prototypische Erstellung einer Sitzungsverwaltung für den Einsatz in Lehrveranstaltungen
Universität Stuttgart, Fakultät Informatik, Studienarbeit, 2001
- BAI 01** Xue Bai
Gemeinsames Lernen von Kommunikationsprotokollen durch elektronisch unterstütztes Rollenspiel
Universität Stuttgart, Fakultät Informatik, Diplomarbeit, 2001

10 Anhang

10.1 Projektplan

Titel: Gemeinsame Nutzung von Smartboards durch Dozent/in und Studierende über mobile Endgeräte

Autor: Peter Oberparleiter

Datum: 8. November 2000

10.1.1 Einleitung

Der Projektplan dient der/dem Bearbeiter(in) und der/dem Betreuer(in) als Grundlage für die Projektüberwachung und -steuerung. Er legt den Projektablauf aber nicht ein für allemal fest, sondern soll im Laufe der Arbeit ergänzt und angepaßt werden. So ist es z.B. möglich, daß anfänglich spezifizierte Arbeitspakete weiter verfeinert oder Arbeitspakete zugunsten anderer aufgegeben werden.

Der Projektplan kann in Absprache mit der/dem Betreuer(in) in beiderseitigem Einverständnis in den folgenden Fällen angepaßt werden:

- bei Verzug,
- bei vorzeitigem Abschluß eines Arbeitspakets oder
- bei Gewinn neuer Erkenntnisse.

Der aktualisierte Projektplan soll in den Anhang der Ausarbeitung einfließen.

Projektbeschreibung: Bei diesem Projekt handelt es sich um eine Studienarbeit an der Abteilung Verteilte Systeme des Instituts für Parallele und Verteilte Höchstleistungsrechner der Universität Stuttgart.

Diese Studienarbeit baut auf den Erfahrungen mit einer bereits durchgeführten Arbeit auf, welche die Steuerung von Powerpoint-Präsentationen auf einer Smartboard-Anzeige durch ein mobiles Endgerät zum Thema hatte. Im Rahmen dieser Arbeit soll ein System mit erweiterter Funktionalität entworfen und prototypisch realisiert werden, welches sich wenn möglich in ein gegebenes Konferenzsystem integrieren läßt. Die Funktionalität des Systems umfaßt die folgenden Punkte:

- Automatische Übertragung von Präsentationsdateien (Powerpoint) auf einen Präsentationsserver
- Steuerung der Präsentation von mobilen Endgeräten aus
- Austausch von Anmerkungen zwischen Smartboard und mobilen Endgeräten

- Zeigemöglichkeit von mobilen Endgeräten aus
- Koordination des Zugriffes mehrerer Endgeräte per floor control, steuerbar durch Dozent/in

Der Bearbeitungszeitraum erstreckt sich über 6 Monate, beginnend am Mittwoch, den 8. November 2000 bis Montag, 7. Mai 2001.

Die Studienarbeit wird betreut von Dr. rer. nat. Cora Burger.

In den nachfolgenden Abschnitten werden zunächst die Arbeitspakete identifiziert und beschrieben, dann wird ein Zeitplan zu deren Durchführung festgelegt. Schließlich erfolgt die Definition der Meilensteine und der dafür zu erstellenden Dokumente.

10.1.2 Beschreibung der Arbeitspakete

Dieser Abschnitt umfaßt die Definition der wesentlichen Arbeitspakete, die Abhängigkeiten zwischen einzelnen Paketen und die Identifikation von kritischen Punkten, die die Durchführung des Projektes gefährden könnten.

Definition der Arbeitspakete

Die im Rahmen der Studienarbeit durchzuführenden Tätigkeiten lassen sich in folgende, zusammenhängende Arbeitspakete aufgliedern:

- **AP1, Projektplan:** Erstellung eines initialen Projektplanes
Status: abgeschlossen
- **AP2, Einarbeitung:** Notebook- und Smartboard-Benutzung, Smartboard-API
Status: -
- **AP3, Anforderungsanalyse:** Spezifizieren der für die Erfüllung der geforderten Funktionalität notwendigen Anforderungen an ein Konferenzsystem
Status: -
- **AP4, Recherche:** Bewertung vorhandener Konferenzsysteme anhand des ermittelten Anforderungsprofils, anschließende Betrachtung der Designalternativen Neuimplementation und Integration in Konferenzsystem
Status: -
- **AP5, Anpassung:** Bei einer Designentscheidung zugunsten der Integration in ein Konferenzsystem wird es notwendig, Anpassungen am bestehenden Grundmodul zur Übertragung und Steuerung von Präsentationen vorzunehmen
Status: -
- **AP6, Mehrbenutzeranbindung:** Entwurf der Anbindung von mehreren Endgeräten an das Präsentationssystem
Status: -

- **AP7, Rechtevergabe:** Entwurf der Zugriffssteuerung
Status: -
- **AP8, Testszenarien:** Entwurf von Testszenarien zur Untersuchung der Skalierbarkeit
Status: -
- **AP9, Realisierung:** Prototypische Implementierung des entworfenen Systems
Status: -
- **AP10, Test:** Untersuchung der Implementierung anhand der entworfenen Testszenarien
Status: -
- **AP11, Ausarbeitung:** Dokumentation von Anforderungen, Umgebung, Spezifikation, Implementierung und Bewertung
Status: -
- **AP12, Vortrag:** Präsentation der Ergebnisse im Rahmen des VS-Kolloquiums
Status: -

Abhängigkeiten der Arbeitspakete

Zwischen den Arbeitspaketen der Studienarbeit bestehen die folgenden Abhängigkeiten:

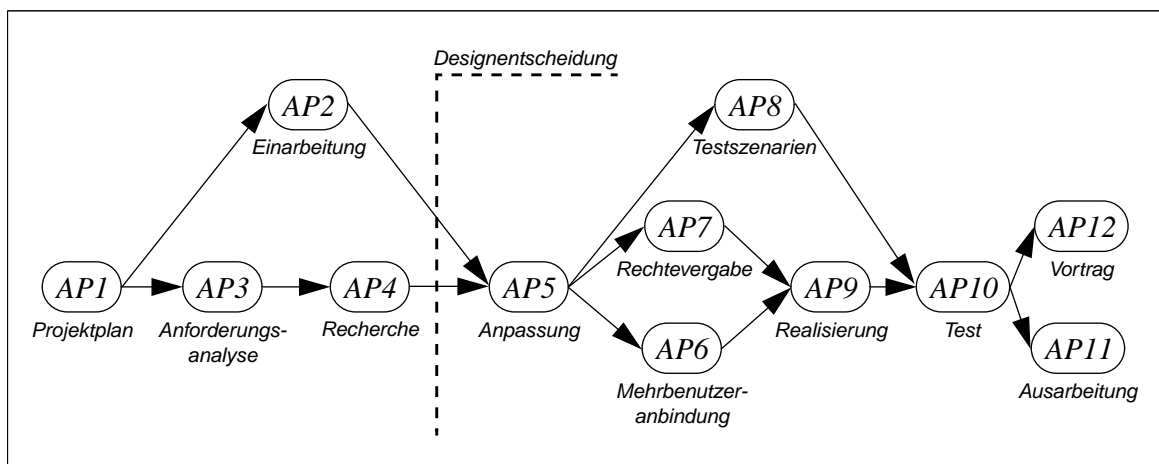


Abbildung 10-1: Abhängigkeiten der Arbeitspakete

Kritische Punkte / Risiken

Ein kritischer Punkt ist nach der Designentscheidung erreicht. Je nach Ausgang der vorhergehenden Analyse und Recherche wird entschieden, auf welcher Basis der weitere Entwurf stattfindet. Fällt die Entscheidung zugunsten der Integration in ein bestehendes Konferenzsystem, so müssen große Teile der bereits entwickelten Präsentationsplattform angepaßt werden, was abhängig von der Offenheit des Konferenzsystems zu Verzögerungen führen kann.

10.1.3 Zeitplan

In diesem Abschnitt erfolgt die zeitliche Planung der Arbeitspakete und Meilensteine. Der Bearbeitungszeitraum ist 6 Monate oder 26 Wochen. Wenn sich der Zeitraum mehrerer Arbeitspakete überlappt, werden diese parallel bearbeitet. Am Ende des Bearbeitungszeitraums ist ein Puffer eingeplant worden, der für Verzögerungen oder andere unvorhergesehene Tätigkeiten genutzt werden soll.

geplanter Zeitraum	Aufwand	Arbeitspakete und Meilensteine	tatsächlicher Zeitraum
1. Woche	1 Tag	AP1 Projektplan	
1.-6. Woche	2 PW	AP2 Einarbeitung	
2. Woche	1 PW	AP3 Anforderungsanalyse	
3.-6. Woche	3 PW	AP4 Recherche	
1. Meilenstein: Designentscheidung			
7.-8. Woche	2 PW	AP5 Anpassung der Grundfunktionalität	
9.-14. Woche	3 PW	AP6 Mehrbenutzeranbindung entwerfen	
9.-14. Woche	3 PW	AP7 Rechtevergabe entwerfen	
9.-20. Woche	1 PW	AP8 Testszenarien entwerfen	
15.-20. Woche	5 PW	AP9 Prototypische Realisierung	
2. Meilenstein: Prototyp			
21. Woche	1 PW	AP10 Test	
22.-24. Woche	2 PW	AP11 Ausarbeitung	
22.-24. Woche	1 PW	AP12 Vortrag	
25. Woche	1 PW	Revision/Korrekturlesen der Dokumente (Betreuerin)	
25. Woche	1 PW	Puffer	

(PW = Personenwoche)

November 2000

Mo	Di	Mi	Do	Fr	Sa	So	#
		1	2	3	4	5	
6	7	8	9	10	11	12	1
13	14	15	16	17	18	19	2
20	21	22	23	24	25	26	3
27	28	29	30				4

Dezember 2000

Mo	Di	Mi	Do	Fr	Sa	So	#
				1	2	3	4
4	5	6	7	8	9	10	5
11	12	13	14	15	16	17	6
18	19	20	21	22	23	24	7
25	26	27	28	29	30	31	

Januar 2001

Mo	Di	Mi	Do	Fr	Sa	So	#
1	2	3	4	5	6	7	
8	9	10	11	12	13	14	8
15	16	17	18	19	20	21	9
22	23	24	25	26	27	28	10
29	30	31					11

Februar 2001

Mo	Di	Mi	Do	Fr	Sa	So	#
			1	2	3	4	11
5	6	7	8	9	10	11	12
12	13	14	15	16	17	18	13
19	20	21	22	23	24	25	14
26	27	28					15

März 2001

Mo	Di	Mi	Do	Fr	Sa	So	#
			1	2	3	4	15
5	6	7	8	9	10	11	16
12	13	14	15	16	17	18	17
19	20	21	22	23	24	25	18
26	27	28	29	30	31		19

April 2001

Mo	Di	Mi	Do	Fr	Sa	So	#
						1	19
2	3	4	5	6	7	8	20
9	10	11	12	13	14	15	21
16	17	18	19	20	21	22	22
23	24	25	26	27	28	29	23
30							24

Mai 2001

Mo	Di	Mi	Do	Fr	Sa	So	#
	1	2	3	4	5	6	24
7	8	9	10	11	12	13	25
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
28	29	30	31				

10.1.4 Dokumente und Meilensteine

Als Ergebnis der jeweiligen Arbeitspakete sollen zu den Meilensteinen folgende Dokumente entstehen, die nach dieser Versionskontrolle unterliegen und nur in Absprache mit der/dem Betreuer(in) geändert werden sollen.

- **Anforderungsanalyse:** Beschreibt die Anforderungen, die zur Erfüllung der Funktionalität an ein Konferenzsystem gestellt werden
- **Recherche-Ergebnis:** Enthält das Resultat der Bewertung vorhandener Konferenzsysteme anhand des ermittelten Anforderungsprofils sowie eine Empfehlung im Hinblick auf die Designalternativen "Neuimplementierung" und "Integration in Konferenzsystem".
- **Prototyp:** Kurze Dokumentation der prototypischen Realisierung.
- **Ausarbeitung:** Dokumentation von Anforderungen, Recherche, Umgebung, Spezifikation, Implementierung und Bewertung

10.2 NetMeeting

Abbildung 10-2 zeigt die Liste der von Microsoft NetMeeting benutzten Netzwerk-Ports.

Port-Nummer	Verwendung
389	Internet Locator Server (TCP)
522	User Location Service (TCP)
1503	T.120 (TCP)
1720	H.323 call setup (TCP)
1732	Audio call control (TCP)
Dynamisch	H323 call control (TCP)
Dynamisch	H.323 streaming (RTP over UDP)

Abbildung 10-2: Liste der von NetMeeting benutzten Ports

10.3 CommServer-API

The CommServer-class can be used to create a communication server to which clients can be connected to and on which channels can be created on.

A server can be started and stopped again. Each time it starts, it generates an address by which clients can connect to it. The address can be queried using getAddress().

There are two lists determining which clients are allowed to join this server, allowed list and disallowed list. The following algorithm determines whether a particular participant is allowed to be connected to this server:

1. Both lists are empty (initial state)
2. The allowedList is not empty and the member in question is part of the allowedList
3. The disallowedList is not empty and the member in question is not part of the disallowedList

The member will be denied access to the session in any other case

Note that all participants will be denied access if the server is not open (see setOpen())

Author:

Peter Oberparleiter

Constructor Summary

[CommServer\(\)](#)

Create a new server using the default communication system implementation.

[CommServer\(boolean auth\)](#)

Create a new server using the default communication system implementation.

[CommServer\(java.lang.String implementation, boolean auth\)](#)

Create a new server with the given implementation type.

Method Summary

void [addToAllowList](#)(java.lang.String participantName)
Add a participant to the list of allowed participants

void [addToDisallowList](#)(java.lang.String participantName)
Add a participant to the list of disallowed participants

[CommChannel](#) [createChannel](#)(java.lang.String channelName)
Create a new channel on this server.

void [disconnectParticipant](#)(java.lang.String participantName)
Disconnect a participant from this server and all channels.

[CommAddress](#) [getAddress](#)()
Retrieve the address of this server.

java.lang.String[] [getAllowedList](#)()
Retrieve the list of allowed participants.

java.lang.String[] [getDisallowedList](#)()
Retrieve the list of disallowed participants

int [getParticipantCount](#)()
Retrieve the number of currently connected participants

java.lang.String[] [getParticipantList](#)()
Retrieve the list of currently connected participants.

boolean [isChannel](#)(java.lang.String channelName)
Check whether a channel with the given name exists on this server.

- boolean [isOpen\(\)](#)
Check whether clients are currently allowed to join this server.
- boolean [isRunning\(\)](#)
Check whether this server is currently running.
- void [remFromAllowList](#)(java.lang.String participantName)
Remove a participant from the list of allowed participants
- void [remFromDisallowList](#)(java.lang.String participantName)
Remove a participant from the list of disallowed participants.
- void [setOpen](#)(boolean open)
Determine whether new clients should be allowed to join this server.
- void [setSessionListener](#)([CommSessionListener](#) newListener)
Specify the listener to be notified of participant and server events.
- [CommAddress](#) [start](#)()
Start the server.
- void [stop](#)()
Stop the server.

10.4 CommClient-API

The CommClient-class serves as a means for clients to connect to aCommServer. It also provides a method to join channels on that server.

Author:

Peter Oberparleiter

Constructor Summary

[CommClient](#)(java.lang.String participantName, java.lang.String participantPassword)
Create a new client using the given name and password but do not connect to a server yet.

[CommClient](#)(java.lang.String participantName, java.lang.String participantPassword, [CommAddress](#) address)
Create a new client using the given name and password and connect it to the CommServer at the given address.

Method Summary

void [connect](#)([CommAddress](#) serverAddress)
Connect this client to the `CommServer` at the given address.

void [disconnect](#)()
Disconnect from the server.

java.lang.String [getClientName](#)()
Get the name of this client.

int [getParticipantCount](#)()
Retrieve the number of participants connected to this server.

java.lang.String[] [getParticipantList](#)()
Retrieve the list of currently connected participants on this server.

boolean [isChannel](#)(java.lang.String channelName)
Check

boolean [isConnected](#)()
Check whether this client is currently connected to a sever.

[CommChannel](#) [joinChannel](#)(java.lang.String channelName)
Join the channel with the given name.

void [setSessionListener](#)([CommSessionListener](#) newListener)
Specify an object which should be notified of participant and server events.

10.5 CommChannel-API

This class provides methods to transmit data among a group of participants.

A channel can be created using the `CommServer.createChannel()`-method. The creator of a channel is automatically the owner of that channel. Furthermore the owner will not show up in the result of a call to `getParticipantList` or `getParticipantCount`. Messages to the owner can be sent using the `sendToOwner()`-method. To determine whether a message originated from a floor owner, use the `CommMessage.isOwnerMessage()`-method. When the owner of a channel calls `leave()`, all other participants are disconnected and the channel is closed.

A channel can be joined using the `CommClient.joinChannel()`-method or `AppClientFrame.joinChannel()`.

Incoming events can be received either synchronously using the `receive()`-methods or asynchronously by registering a `CommChannelConsumer` and setting the current communication mode to asynchronous. To set the mode use `setSync()`.

Author:

Peter Oberparleiter

Constructor Summary[**CommChannel**](#)(sascia.Client newJSDTClient, sascia.Channel newJSDTChannel)

This is a constructor for internal use only.

Method Summary

- java.lang.String [**getClientName\(\)**](#)
Retrieve the name of this client.
- int [**getParticipantCount\(\)**](#)
Retrieve the current number of participants connected to this channel.
- java.lang.String [**getParticipantList\(\)**](#)
[] Retrieve an array of Strings containing the names of all currently connected participants.
- boolean [**getSync\(\)**](#)
Retrieve the current communication mode.
- boolean [**isMessage\(\)**](#)
Check whether there is a new message available.
- void [**leave\(\)**](#)
Leave this channel.
- [**CommMessage**](#) [**receive\(\)**](#)
Receive a message synchronously.
- [**CommMessage**](#) [**receive\(int timeout\)**](#)
Wait a specified amount of time for a new message to arrive.
- void [**sendToAll**](#)(java.lang.Object content)
Send a message to all participants on this channel including the sender.
- void [**sendToOthers**](#)(java.lang.Object content)
Send a message to all participants on this channel except for the sender.
- void [**sendToOwner**](#)(java.lang.Object content)
Send a message to the owner of this channel.
- void [**sendToParticipant**](#)(java.lang.String recipient, java.lang.Object content)
Send a message to a particular participants on this channel.

- void [setChannelConsumer](#)([CommChannelConsumer](#) consumer)
Set the consumer object which should be notified of new messages.
- void [setChannelListener](#)([CommChannelListener](#) listener)
Set the listener object which should be informed when participant or channel events occur.
- void [setSync](#)(boolean newSync)
Set the current communication mode of this channel.

10.6 FloorServer-API

This class represents a floor control server. It contains all the floor data and it allows floor control clients to join.

Author:

Peter Oberparleiter

Constructor Summary

[FloorServer](#)([CommServer](#) server, java.lang.String name)
Create a new floor control server.

[FloorServer](#)([CommServer](#) server, java.lang.String name, int size)
Create a new floor control server.

[FloorServer](#)([CommServer](#) server, java.lang.String name, int size, java.lang.String chair)
Create a new floor control server.

[FloorServer](#)([CommServer](#) server, java.lang.String name, int size, java.lang.String chair, [FloorStrategy](#) strategy)
Create a new floor control server.

Method Summary

- void [abortPoll](#)([PollID](#) pollID)
Abort a poll
- [PollID](#) [createPoll](#)(java.lang.String description)
Create a new poll
- java.lang.String [getChairName](#)()
Get the name of the current chair person
- int [getFloorApplicantCount](#)()
Get the number of floor participants

java.lang.String[] [getFloorApplicantList\(\)](#)
Get a list of all participants who currently apply for the floor

int [getFloorOwnerCount\(\)](#)
Get the number of floor owners

java.lang.String[] [getFloorOwnerList\(\)](#)
Get a list of all current floor owners

int [getFloorSize\(\)](#)
Get the current floor size

[FloorStrategy](#) [getFloorStrategy\(\)](#)
Get the current floor passing strategy

int [getParticipantCount\(\)](#)
Get the number of registered participants

java.lang.String[] [getParticipantList\(\)](#)
Get a list of all currently registered participants

[Poll](#) [getPoll\(PollID pollID\)](#)
Retrieve the data of a poll

int [getPollCount\(\)](#)
Retrieve the number of open polls in this floor control

[Poll\[\]](#) [getPollList\(\)](#)
Retrieve a list of all open polls

[VoteCount](#) [getTotalCount\(\)](#)
Get the total count of active votes

[VoteCount](#) [getVoteCount\(java.lang.String participant\)](#)
Get the vote count of a particular participant

boolean [isFloorApplicant\(java.lang.String participant\)](#)
Check whether a participant is currently applying for the floor

boolean [isFloorOwner\(java.lang.String participant\)](#)
Check whether a participant owns the floor

boolean [isParticipant\(java.lang.String participant\)](#)
Check whether a given participant is taking part in this floor control

void [setFloorListener\(FloorListener newListener\)](#)
Set the listener for events generated by this floor control.

void [setFloorSize\(int size\)](#)
Set the floor size

void [setStrategy](#)([FloorStrategy](#) newStrategy)
Set the floor strategy

void [shutdown](#)()
Shut this floor control server down.

10.7 FloorClient-API

This FloorClient-Object provides access to a given floor control on this system.

Author:

Peter Oberparleiter

Constructor Summary

[FloorClient](#)([CommClient](#) client, java.lang.String name)

Create a new floor control client and connect it to the floor control with the specified name through the given CommClient.

Method Summary

void [claimFloor](#)()
Claim the floor.

void [disconnect](#)()
Disconnect the floor control from the floor control server.

void [exchangeFloor](#)(java.lang.String fromParticipant, java.lang.String toParticipant)
Exchange the floor between two participants.

java.lang.String [getChairName](#)()
Retrieve the name of the current chair person or NULL if there currently is no chair.

int [getFloorApplicantCount](#)()
Retrieve the number of floor applicants

java.lang.String[] [getFloorApplicantList](#)()
Retrieve a list of all applicants to the floor

int [getFloorOwnerCount](#)()
Retrieve the number of current floor owners

java.lang.String[] [getFloorOwnerList](#)()
Retrieve a list of all current floor owners

int [getFloorSize](#)()
Retrieve the current floor size

FloorStrategy	getFloorStrategy() Retrieve the current floor strategy
int	getParticipantCount() Retrieve the number of registered participants
java.lang.String[]	getParticipantList() Retrieve the list of all registered participant.
java.lang.String	getParticipantName() Return the name of the participant by which this Floor-Client is known in the floor control.
Poll	getPoll(PollID pollID) Retrieve the data of a poll
int	getPollCount() Retrieve the current number of open polls
Poll[]	getPollList() Retrieve the list of all open polls.
VoteCount	getTotalCount() Retrieve the current total number of active votes.
VoteCount	getVoteCount(java.lang.String participant) Retrieve the number of votes of a single participant.
void	grantFloor(java.lang.String participant, boolean clearFloor) Grant the floor to the specified participant.
boolean	isFloorApplicant(java.lang.String participant) Check whether the participant with the given name currently applies for the floor.
boolean	isFloorOwner(java.lang.String participant) Check whether the participant with the given name owns the floor
boolean	isParticipant(java.lang.String participant) Check whether a participant with the given name is registered with this floor control.
void	passFloor(java.lang.String toParticipant) Pass the floor to another participant.
void	relinquishFloor() Relinquish the floor.
PollID	requestExchangeFloor(java.lang.String fromParticipant, java.lang.String toParticipant) Create a new poll to exchange a floor owner.
PollID	requestGrantFloor(java.lang.String participant, boolean clearFloor) Create a new poll to grant the floor to a participant.

- [PollID](#) **[requestRevokeFloor](#)**(java.lang.String participant)
Create a new poll to revoke the floor from a participant.
- [PollID](#) **[requestSetFloorSize](#)**(int newSize)
Create a new poll to set the current floor size.
- [PollID](#) **[requestSetStrategy](#)**([FloorStrategy](#) newStrategy)
Create a new poll to set the current floor passing strategy.
- void **[revokeFloor](#)**(java.lang.String participant)
Revoke the floor from a participant.
- void **[setFloorListener](#)**([FloorListener](#) newListener)
Register the listener for events generated by this floor or NULL if no listener is required.
- void **[setFloorSize](#)**(int newSize)
Set the maximum number of concurrent floor owners.
- void **[setStrategy](#)**([FloorStrategy](#) newStrategy)
Set the floor passing strategy of this floor control.
- void **[voteAbstain](#)**()
Abstain from voting.
- void **[voteAgainstPoll](#)**([PollID](#) pollID)
Vote against a poll.
- void **[voteForPoll](#)**([PollID](#) pollID)
Vote for a poll.
- void **[voteParticipant](#)**(java.lang.String participant)
Vote for a participant.
- void **[voteRevoke](#)**()
Do not vote for anything.
- void **[withdrawClaim](#)**()
Withdraw a claim to the floor.