Studiengang: Informatik

Prüfer: Prof. Dr. rer. nat. Kurt Rothermel

Betreuer: Dr. Cora Burger

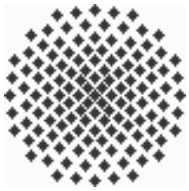
Begonnen am: 16.04.01
Beendet am: 15.10.01

CR-Nummer: C.2.4, H.4.3, I.2.6, K.3.1, K.3.2

Diplomarbeit-Nr.: 1930

Gemeinsames Lernen von Kommunikationsprotokollen durch elektronisch unterstütztes Rollenspiel

Xue Bai



Universität Stuttgart
Fakultät Informatik

Institut für Parallele und Verteilte
Höchstleistungsrechner
Breitwiesenstraße 20-22
D-70565 Stuttgart

Inhaltverzeichnis

Abbildungsverzeichnis

1 Einleitung

Motivation

Der Einsatz von Computern zum Lernen wird derzeit an vielen Universitäten erprobt. Die einen versprechen sich davon langfristig eine Kostenersparnis. Werden hingegen qualitative Verbesserungen des Lernens in den Vordergrund gestellt, dann wird am Lernprozess angesetzt. Ein sehr vielversprechender Ansatz ist das kollaborative Lernen.

In den Bereichen *Rechnernetz, Verteilte Systeme, Kooperation in verteilten Systemen und Spezifikation und Verifikation von Kommunikationsprotokollen* basieren die Lehrinhalte hauptsächlich auf dem Austausch von Nachrichten zwischen Komponenten über einen Kommunikationskanal, wobei sich die Komponenten an ein bestimmtes Kommunikationsprotokoll halten. In den letzten Jahren haben die Kommunikationsprotokolle deutlich an Komplexität zugenommen. Die wachsende Bedeutung von Kryptographie in Kommunikationsprotokollen z.B. hat einen wesentlichen Teil dazu beigetragen. Daher sind die Kommunikationsprotokolle in der Lehre schwer zu vermitteln. Um das Erlernen von Kommunikationsprotokollen zu erleichten, entstand der sogenannte HiSAP-Baukasten (**H**ighly **i**nteractive **S**imulation of **A**lgorithm and **P**rotocols) zur einfacheren Erstellung von interaktiven Animations- und Simulationsapplets. Mit dessen Hilfe können Sachverhalte in der Vorlesung visualisiert und im Selbststudium genauer untersucht werden.

Aber die Möglichkeit, Protokolle zusammen mit anderen durch Übernahme der Rollen von beteiligten Kommunikationspartnern zu erkunden (kollaboratives Lernen) und die Erfahrungen anhand von gemeinsamen und privaten Notizen festzuhalten, fehlt bislang.

Das Projekt SASCIA (**S**ystem **A**rchitecture **S**upporting **C**ooperative **I**nteractive **A**pplications) ist ein Konferenzsystem, das besonders für Lehrveranstaltungen geeignet ist. Basierend auf JSDT (**J**ava **S**hared **D**ata **T**oolkit) kann es die interaktiven, kollaborativen Applikationen unterstützen und ermöglicht damit das kollaborative Lernen von Kommunikationsprotokollen zu realisieren.

Zur Zeit bietet SASICA nur die folgende Funktionalität:

- Sitzungsverwaltung
- Teilnehmerverwaltung
- Protokollierungsdatenbank
- gemeinsame Nutzung des Smartboards.

Um die spezifische Applikation HiSAP in SASCIA zu intergieren und damit das kollaborative Lernen zu ermöglichen, ist auch SASCIA selbst zu erweitern.

Aufgabenstellung

1. Einarbeitung in die Themengebiete:
   - kollaboratives Lernen
   - JSDT
   - HiSAP-Baukasten

2. Einarbeitung in die Komponenten von SASCIA für Sitzungsverwaltung, generische Rederechtvergabe und Protokollierungsfunktion für öffentliche und private Kommentare.
3. Anforderungsanalyse bzgl. Interaktivität, gemeinsame Bedienbarkeit und Notizfähigkeit.
4. Spezifikation, Entwurf der Konzepte für Erweiterungen im Baukasten und Konferenzsystem. Außerdem gehört zu diesem Aufgabenblock auch die prototypische Implementierung der oben beschriebenen Komponenten, sowie die Erstellung eines Anwendungsbeispiels.
5. Bewertung anhand des Testfalls.

Überblick

Im folgenden Kapitel werden die Grundbegriffe, kollaboratives Lernen, das Rollenkonzept und die Floor-Control erklären.

Im Kapitel 3 & 4 wird ein kurzer Überblick über den existierenden HiSAP-Baukasten und das SASCIA-Konferenzsystem gegeben, die im Rahmen dieser Diplomarbeit entstanden und eine Ist-Analyse durchgeführt.

In Kapitel 5 wird die Funktionalität des Zielsystems genauer definiert und weiterhin die Anforderungsanalyse behandelt. Die Analyse befaßt sich mit den Möglichkeiten des Rollenspiels beim gemeinsamen Lernen von Kommunikationsprotokollen und damit, inwieweit die Interaktivität beim gemeinsamen Lernen von Kommunikationsprotokollen benötigt wird.

Der Hauptteil dieser Diplomarbeit wird im Kapital 6 behandelt, der in Entwurf der Erweiterungen von SASCIA und HiSAP-Baukasten gegliedert ist. Der Entwurf der Erweiterung von SASCIA befaßt sich mit den Defiziten der alten SASCIA Floor-Control-Komponenten und bestimmt die zu überarbeitenden Teile der Komponenten. Auch der Visualisierungs- und Simulationsbaukasten in HiSAP mußte erweitert werden, damit das Zielsystem Client-Server-fähig wird. Danach folgt die konkrete Implementierung des neuen Baukastens.

Kapitel 7 beschreibt die Implementierung einer Protokollsimulation, die auf SASCIA mit HiSAP basiert. Zuerst wird ein Überblick über die Grundprinzipien gegeben, danach die Verwendung durch ein konkretes Anwendungsbeispiel veranschaulicht.

Schließlich werden im Kapital 8 eine Zusammenfassung und ein Ausblick gegeben.

2 Grundlage

In diesem Kapitel werden die grundlegenden Begriffe erläutert, die für die nachfolgenden Kapitel wesentliche Bedeutung haben. Zuerst wird der Begriff des kollaborativen Lernens erläutert und die grundlegenden Anforderungen des kollaboratives Lernens an Kommunikation und Kooperation. Danach wird die Floor-Control beschrieben.

2.1 Kollaboratives Lernen

Beim Lernprozess wird häufig der Begriff “Kollaboratives Lernen” benutzt. Was ist eigentlich kollaboratives Lernen? Was bringt uns das kollaborative Lernen?

Unter kollaborativem Lernen verstehen wir das Lernen in Gruppen. Mit anderen Worten können wir sagen, kollaboratives Lernen bedeutet das gemeinsame Bearbeiten aller Aufgaben, die für den Lernprozeß vorgesehen sind. Durch Kooperation wird die gemeinsame Aufgabe in Teilaufgaben zerlegt, die auf Mitglieder der Gruppe verteilt werden. Dies ermöglicht die gemeinsame Lösung eines Problems. Durch Kooperation und gegenseitige Kontrolle wird eine Motivationssteigerung der Zusammenarbeitenden erzielt.

Die größten Vorteile beim kollaborativen Lernen sind die durch die verschiedenen Gruppenteilnehmer und deren unterschiedliche Sichtweisen von The-

men/Problemstellungen sich ergebenden Synergieeffekte. Durch Interaktion und Diskussion werden die besten Ideen und Ansätze ermittelt und auch eventuelle Schwachstellen Einzelner aufgefangen [SöRe99]. Auch betont Schönfeld in seinem Buch *Mathematical Problem Solving* die Wichtigkeit kollaborativen Lernens im Zusammenhang mit *kontrolliertem Problemlösen. "[Wichtig für die Fähigkeit, mathematische Probleme lösen zu können, ist]... das Betrachten von Situationen aus verschiedenen Blickwinkeln, das Planen und Evaluieren neuer Ideen, sowie das Beobachten und Beurteilen von Lösungen usw. während des Arbeitsvorgangs. Wo entstehen solche Verhaltensweisen, und wie lernt man, während des Problemlösevorgangs mit sich selbst zu diskutieren? ... Es scheint logisch, daß dies durch gemeinsames Problemlösen ... [erreicht wird]."*

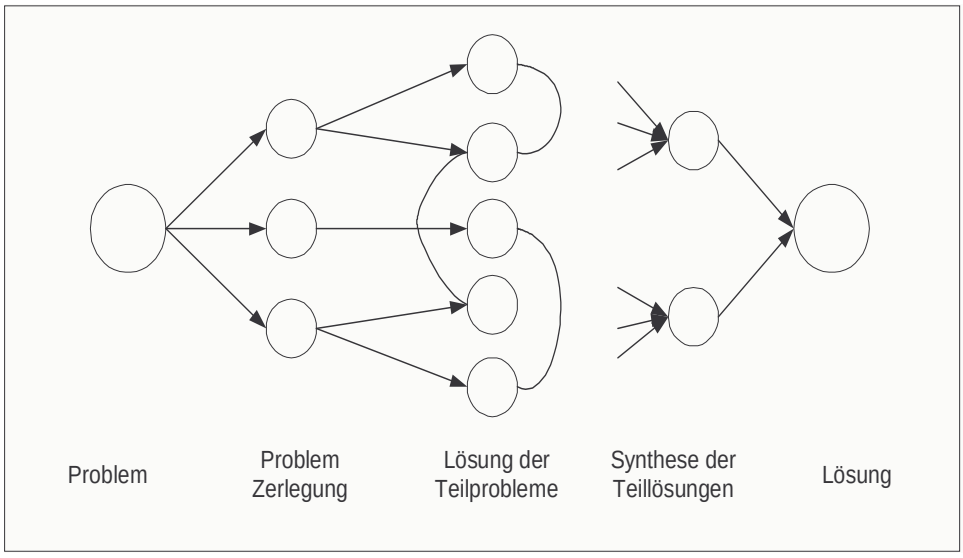


Abbildung 2.1: Phase des kooperativen Problemlösens

Kollaboratives TeleTeaching (im Englischen: CSCL - **C**omputer **S**upported **C**ollaborative **L**eraning) ist ein Teilgebiet von CSCW (Computer Supported Cooperative Work). Es wird durch Groupware (spezielle Hard- und Software

für Teams) umgesetzt. Mit den grundlegenden Betrachtungen zur Nutzung von Rechnern und Anforderungen an die Groupware läßt sich nun eine erste grobe Klassifikation von Groupware-Systemen vornehmen. Grad der Unterstützung der Nutzungsformen: informations-, ablauf- und strategieorientiert, was als Kommunikations-, Koordinations- und Kooperationsunterstützung bezeichnet wird [Burg97]. Abbildung 2.2 zeigt die Einordnung von Groupware nach diesen Kriterien [Burg97].

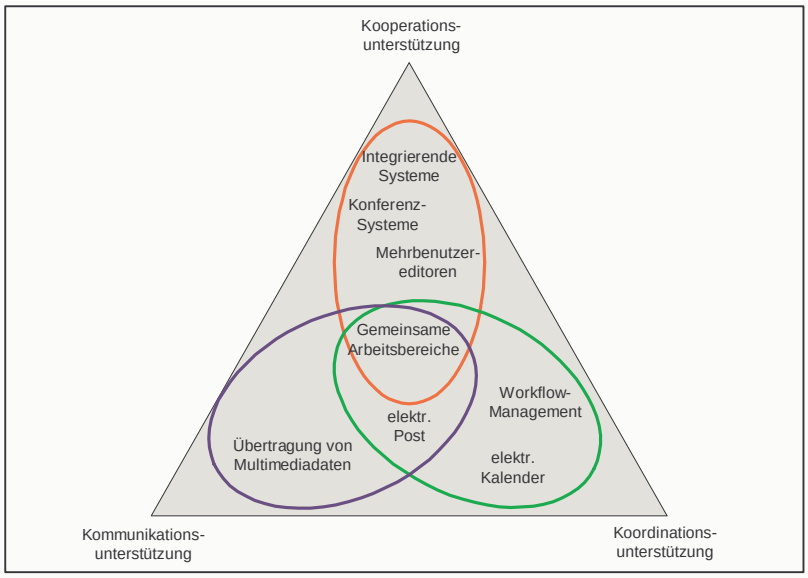


Abbildung 2.2: Klassifikation von Groupware

Die Kommunikation betrachtet Mitteilung und Austausch von Information. Die Koordination betrachtet Abstimmung von Akteuren, Aktivitäten und Ressourcen. Die Kooperation betrachtet das Zusammenwirken verschiedener Personen und System zu einem Ganzen, z.B. gemeinsame Arbeitsbereiche.

Um die Koordination im kollaborativen Lernen zu gewährleisten, wird der Begriff: Awareness betrachtet. Der Begriff Awareness (im Deutschen: Bewußtsein) beschreibt die Interaktion zwischen Umgebung und Individuum.

Awareness ermöglicht dem Individuum, die aktuelle Situation in einer Umgebung zu erfassen und sein Handeln darauf abzustimmen. Dazu müssen einerseits die Ereignisse der Umgebung wahrnehmbar sein (Wahrnehmbarkeit) und andererseits muß das Individuum die Fähigkeit haben, die Umwelt zu bewachen (Sensorik) und zu erkennen (Kognition) [Pank97]. Im Zusammenhang mit dem Begriff des “Kollaborativen Lernens” muß man bei Awareness von der Interaktion zweier oder mehrerer Lernenden untereinander, bzw. mit einem Lehrenden sprechen. Unter Awareness-Information versteht man Informationen über Zustände sowie über Ereignisse, die durch Handlungen von Personen und Veränderungen von Objekten ausgelöst werden. Auch Zusammenfassungen derartiger Informationen werden als Awareness-Informationen bezeichnet.

In folgenden Unterkapitel werden wir im Detail diskutieren, welchen Kommunikationsarten im kollaborativen Lernen nötig sind, welche Arten von Awareness für kollaboratives Lernen von Bedeutung sind. Anschließend wird das Rollenkonzept behandelt.

2.1.1 Kommunikationsarten

In diesem Abschnitt wird ein erstes Grundschema der Kommunikationsarten vorgestellt. Es wird noch verdeutlicht, welche Kommunikationsarten für das kollaborative Lernen notwendig sind.

Bei der Kommunikation müssen wir zwei Kriterien betrachten. Das erste Kriterium ist die zeitliche Dimension (synchron/asynchron) der Kommunikation.

Die zweite Komponente ist die Art der Informationsübermittlung (implizit/explizit).

Nach dem ersten Kriterium ist eine Kommunikation entweder synchron oder asynchron:

- **Synchrone Kommunikation**
  Bei der *synchronen Kommunikation* sind alle Teilnehmer zur gleichen Zeit aktiv. Bei dieser Kommunikationsart interagieren alle Beteiligten miteinander und können somit “live” Informationen austauschen.

- **Asynchronen Kommunikation**
  Bei der *asynchronen Kommunikation* sind die Beteiligten nicht zur gleichen Zeit in Aktion. Hier kann eine längere Zeit verstreichen bis andere Beteiligte vorherige Aktionen registrieren und selbst reagieren. [KoKöBü97]

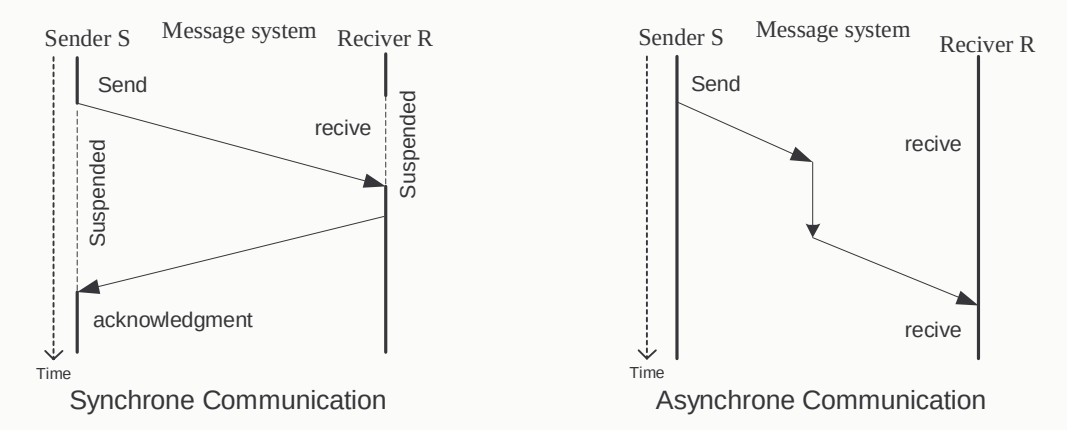


Abbildung 2.3: Kommunikation nach zeitlichen Dimension

Das zweite Kriterium ist die Unterscheidung in expliziter oder impliziter Kommunikation:

- **Explizite Kommunikation**
  Explizite Kommunikation erfordert eine direkte Aktion derjenigen Teilnehmer, die kommunizieren möchten. Bei dieser Form von Informationsübermittlung sind ausschließlich die direkt erkennbaren

und bewußt dargestellten Daten einzuordnen. Die typische explizite Kommunikation sind persönliche Gespräche, Video-Konferenzen, Audio-Konferenzen oder E-Mail.

- **Implizite Kommunikation**
  Implizite Kommunikation nennt man aufgrund geteilter Informationen entstehende Wahrnehmungen. Als Beispiel kann man die Wahrnehmung eines Ereignisses und der darin enthaltenen Information anführen. [KoKöBü97]

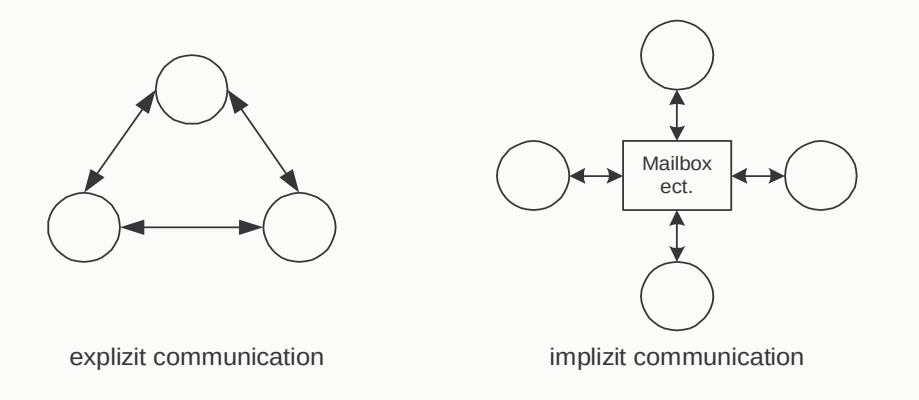


Abbildung 2.4: Kommunikation nach Informationsübermittlung

Für das kollaborative Lernen ist der Lernkontext in einem physikalischen Raum am besten untersucht und es spricht auch vieles dafür, daß viele Lernaktivitäten in dieser Umgebung am besten gestaltet werden können. Während wir die Überschneidungen der beiden Kriterien betrachten, kommen wir auf zwei Kommunikationsarten zurück, in die jede Form von Daten-/Informationsaustausch eingeteilt werden kann:

- Explizite synchrone Kommunikation (z.B. Vortrag, Lehrveranstaltung, Übungen, usw.) Diese Kommunikationsart bietet einen interaktiven Informationsaustausch in Lehrveranstaltung.
- Explizite asynchrone Kommunikation (z.B. Lehrinhalten zum Download & Upload). Diese Kommunikationsart ist für das Wiederholen des Lernprozesses geeignet.

2.1.2 Awareness-Information

Durch die oben genannten Kommunikationsarten könnten entweder zu viele oder nicht genügend Informationen für einen Transfer von Wissen aus dem Gebiet des CSCW (Computer Supported Cooperative Work) in das kollaborativen Lernen geboten werden. Es ist noch zu unterscheiden, welchen Informationen übertragen werden sollen. In diesem Abschnitt wird verdeutlicht, welche Arten von Awareness von Bedeutung sind.

Alle für die Kooperation bzw. Interaktion mehrerer Beteiligter nötigen Awareness Informationen kann man in folgende 3 Obergruppen unterteilen [Pank98]:

- **Personal Awareness**
  Diese Menge an Awareness-Informationen beschreibt einerseits die Person/Gruppe (z.B. persönliche Daten, emotionale Situation), andererseits die Beziehung zu anderen Gruppemitgliedern (z.B. Rolle innerhalb einer Konferenzsitzung).

- **Tool Awareness**
  Diese Menge an Awareness-Informationen bezieht sich auf die eingesetzten Tools. Das wäre zum Beispiel die Angabe der Personen, die mit dem Tool arbeiten, der Arbeitsstaus der mit diesem Tool as-

soziierten Aufgaben oder das Datenobjekt, das mit diesem Tool manipuliert wird.

- **Data Awareness**
  Diese Menge an Awareness-Informationen bezieht sich auf Datenelemente und gibt unter anderem an, welche Personen gerade mit einem Datenobjekt interagieren, welche Änderungen auf dem Datenobjekt durchgeführt werden oder mit welchen Tools das Datenobjekt bearbeitet werden kann.

2.1.3 Rollenspiel im kollaborativen Lernen

Im kollaborativen Lernen wird das gemeinsame Bearbeiten aller Aufgaben, die für den Lernprozeß vorgesehen sind, durch das Rollenkonzept ermöglicht. Im Rollenkonzept übernehmen die Gruppenmitglieder nicht mehr eine feste Stelle sondern unterschiedlichen Rollen. Dadurch können die Studierenden die verschiedenen Teilaufgaben, die mit verschiedenen Rollen kombinieren, ausüben. Damit wird der Lerneffekt des kollaborativen Lernens verbessert. Aus diesem Grund lassen sich folgenden Begriffe ableiten.

Eine **Person** ist eine natürliche oder juristische Person und existiert “permanent” und unabhängig von spezifizierten Rollen. Sie verfügt über Kompetenz. Die Zuordnung von Aufgaben (Aktivitäten) zu Personen wird über Rollen vorgenommen.

Eine **Rolle** ist eine Abstraktion von Personen und beinhaltet bestimmte Fähigkeiten und Rechte, die ein Rolleninhaber für die Erfüllung der ihm zugeordneten Aufgaben besitzen muß. Sie existiert permanent und unabhängig davon, ob jemand in ihr aktiv ist. Rollen identifizieren Tätigkeiten, die gegebenenfalls

nur einen Teil der gemeinsamen Aufgabe, gegebenenfalls aber auch in einem bestimmten Zeitintervall der Lehrveranstaltung verschiedener Aufgaben beanspruchen. Deshalb kann ein Gruppenmitglied durchaus mehrere Rollen innehaben bzw. eine Rolle von mehreren Mitgliedern wahrgenommen werden.

Ein **Akteur** ist ein Mitglied der Gruppe, das im Prinzip nur eine Rolle wahrnimmt, die in einem bestimmten Zeitintervall der Lehrveranstaltung aufgabenabhängig spezifiziert wird, und die Lösungen seiner Teilaufgabe liefert.

Ein **Beobachter** ist eine Person, die ein Mitglied der Gruppe ist, das gegenwärtig in der Lehrveranstaltung nur zuschaut und keine Teilaufgabe übernimmt.

Ein **Moderator** ist eine Person, die der Gruppe vorsitzt. Er kann ein Mitglied dieser Gruppe, aber auch ein/e Dozent/in sein. Gemäß der konkreten Lehrveranstaltung (z.B. Übung oder Vorlesung) kann die Moderation von unterschiedlichen Leuten ausgeübt werden.

Akteur, Beobachter und Moderator sind grundsätzliche Rollen im kollaborativen Lernen. Im Vorgang einer Lehrveranstaltung gibt es auch Wechselmöglichkeit zwischen Akteur, Beobachter und Moderator. Gemäß den unterschiedlichen Lehrveranstaltungen könnte der Akteur auch mehrere unterschiedliche, in einer konkreten Aufgabe spezifizierten, Rollen ausüben, wenn die Anzahl der Gruppenmitgliedern weniger als die Anzahl der Rollen ist. Dies wird in Kapitel 6 im Detail diskutiert.

2.2 Floor-Control

Floor-Control stellt einen Mechanismus zur Koordinierung von Benutzeraktionen in verteilen Anwendung dar [DoGa97]. Sie dient zur Auflösung bzw. Vermeidung von Zugriffskonflikten auf gemeinsam genutzten Ressourcen und ermöglicht somit eine sinnvolle gemeinsame Nutzung dieser durch mehrere Anwender. Typischerweise ist der Bereich der Zugriffskontrolle auf Daten traditionell Gegenstand von Datenbanken. Er wird im Rahmen von Konferenzsystemen verschoben und erfordert dementsprechend teilweise neue Mechanismen bzw. Strategien. Zu den Aufgabengebieten der Floor-Control gehören [DoGa97]:

- Zugriffskontrolle für gemeinsam genutzte Ressourcen, Überwachung des Prinzips des “gegenseitigen Ausschlusses”.
- Zustandsüberwachung von verteilten Datenobjekten (bzgl. Konsistenz).
- Bereitstellung verschiedener Kontrollstrategien.
- Gewährleistung der Fairness bei der Floor-Vergabe.
- Sicherungsmechanismen bei Verlust oder Duplizierung des Floors

Der **Floor** ist als abstrakter Begriff zu verstehen, der mit dem Zugriffsrecht auf bestimmte Ressourcen verknüpft ist. Er stellt eine temporäre Erlaubnis für den Zugriff auf gemeinsam genutzte Ressourcen dar. Die Art und Weise, wie der Floor in Anwendungen realisiert wird, bezeichnet man als **Floor-Kontrollmechanismus** (floor control mechanism), während die Strategie der Floor-Vergabe als **Floor-Kontrollstrategie** (floor control policy) bezeichnet wird [Crow90], [Rein94].

2.2.1 Floor-Kontrollmechanismen

Im wesentlichen lassen sich die folgenden Mechanismen zur Floor-Control unterscheiden: Token, Zeitstempel und Semaphore.

- **Token**
  Der am häufigsten implementierten Floor-Kontrollmechanismus stellt die Token-Vergabe dar. Der Floor wird durch ein Token repräsentiert, welches entweder von den Teilnehmern explizit oder von den Anwendungen implizit weitergegeben wird.

- **Zeitstempel**
  Zeitstempel als Floor werden zu Synchronisationszwecken benutzt oder um eine bestimmte Reihenfolge von Ereignissen zu gewährleisten.

- **Semaphore**
  Semaphore können eingesetzt werden, um Zugriffe auf kritische Bereiche zu koordinieren.

2.2.2 Floor-Strategien

Es lassen sich verschiedene Basisstrategien der Floor-Control unterscheiden. Im einzelnen sind die folgenden Basisstrategien festzuhalten[Bran97]:

- **explizit**
  Bei der expliziten Floor-Control übernimmt ein Teilnehmer die Koordinierung des Floors. Bei der *benutzergesteuerten* expliziten Kontrolle existiert für diese Aufgabe ein Sitzungsleiter, welcher für eine faire Vergabe des Floors verantwortlich ist. Bei der *autonomen* expliziten Floor-Control wird der Floor vom aktuellen Besitzer aktiv weitergegeben. Dieser kann allein über Vergabezeitpunkt und künftigen Besitzer entscheiden. Bei dieser Variante ist die Fairness kaum zu gewährleisten, da ein egoistischer Teilnehmer den Floor beliebig

lange blockieren bzw. lediglich an bevorzugte Teilnehmer weitergegeben kann.

- **implizit:**
  Bei impliziten Strategien wird die Koordination und Weitergabe des Floors vom System übernommen. Dies kann zum einen zentral geschehen, d.h. es existiert eine Koordinationsstelle im System, die über die Vergabe des Floors entscheidet. Hierbei kann die Fairness gewährleistet werden, indem beispielsweise dem aktuellen Floorinhaber bei zu langer Besitzdauer der Floor entzogen wird. Diese Kontrollvariante wird auch als *automatische* Floor-Control bezeichnet. Zum anderen wird bei der dezentralen impliziten Floor-Control die Koordinierung des Floors von den einzelnen Anwendungen übernommen. Benötigt ein Teilnehmer den Floor, so entscheidet die Anwendung des aktuellen Inhabers - nicht der Inhaber selber - über eine mögliche Weitergabe.

Häufig benutzte Strategien (dynamisch) zur Floor-Control in Konferenz Anwendungen sind die folgenden [DoGa97], *Sitzungsleitergesteuert* und *Teilnehmergesteuert*:

- **Sitzungsleitergesteuert (Moderator):**
  Ein ausgewählter Teilnehmer (oft der Initiator der Sitzung) ist als Sitzungsleiter für die Vergabe des Floors verantwortlich, d.h. er bestimmt die jeweils eingesetzte Strategie. Es handelt sich hier also um explizite Floor-Control.

- **Teilnehmergesteuert**
  Der aktuelle Inhaber des Floors entscheidet, wann und an welchen Teilnehmer der Floor weitergegeben wird. Im Gegensatz zum Konferenzleiter wechselt hier die Kontrolle dynamisch im Verlauf einer Sitzung. Die Floor-Control ist wie beim Konferenzleiter explizit aber autonom.

Die oben genannten Strategien sind alle in der Lage, den dynamischen Verlauf einer Konferenz bzw. Zusammenarbeit zu unterstützen. Die nachfolgende Strategie ist diesbezüglich eher statisch ausgerichtet:

- **Abstimmung**
  Hierbei wählen die Teilnehmer den neuen Floorinhaber durch Abstimmungsverfahren aus, beispielsweise durch Mehrheitsverfahren. Diese Strategie ist nur bei größeren Wechselintervallen des Floors sinnvoll, da mit der Abstimmung ein nicht unerheblicher zeitlicher Aufwand verbunden ist. Strategien zur Abstimmung sind abgeleitet von Voting-Strategien im Bereich der Verteilten Systeme [BoSc95].

Nach der Erklärung der benötigten Begriffe wird im folgenden Kapitel eine Ist-Systemanalyse von HiSAP behandelt.

3 Ist-Systemanalyse: HiSAP - das Simulationssystem

Das Projekt HiSAP (**H**ighly **i**nteractive **S**imulation of **A**lgorithm and **P**rotocols) zielt auf eine rechnergestützte interaktive Simulation und Visualisierung von Algorithmen und Protokollen ab.

Im Lehrbereich Verteilte Systeme sind die Kommunikationsprotkolle und Algorithmen deutlich komplex. Somit fällt es Studierende zunehmend schwerer, Kommunikationsprotokolle zu verstehen. Umgekehrt ist es auch schwer für den Lehrer, die Kommunikationsprotokolle zu erklären. Um das Erlernen und Lehren von Kommunikationsprotokollen zu erleichtern, versucht die Abteilung *Verteilte Systeme* der Universität Stuttgart, Kommunikationsprotokolle geeignet darzustellen und einen eigenen Baukasten zu erstellen. Der HiSAP-Baukasten liefert nicht nur die Simulationsmöglichkeit vielfältiger Kommunikationsprotokolle und Algorithmen, sondern auch die Manipulationsmöglichkeiten des Ablaufs der Simulation.

Zum einen soll der HiSAP-Baukasten den Studierenden darin unterstützen, Algorithmen bzw. Protokolle so darzustellen, daß diese von den Studierenden besser und schneller verstanden werden. Dabei soll der Baukasten die Möglichkeit liefern, während des Vortrages durch Interaktionen des Lehrenden bestimmte Abläufe der Algorithmen bzw. Protokolle hervorzuheben. Zum anderen soll der HiSAP-Baukasten dem Studierenden ermöglichen, die vorgestellten Algorithmen bzw. Protokolle jederzeit wieder zu betrachten, und

gleichzeitig mittels interaktiver Benutzung verschiedener Abläufe durchzuspielen. So kann der Studierende sein Wissen vertiefen bzw. Unklarheiten beseitigen.

Im Rahmen des HiSAP-Projektes wurden schon viele Diplom- bzw. Studienarbeiten durchgeführt. Vor der Diplom- bzw. Studienarbeit von Torsten Brodbeck hatten viele Studenten den Visualisierungsbaukasten und die Werkzeuge zur Generierung von Simulationsmodellen entworfen. Torsten Brodbeck hat die systematische Architektur erstellt und den Visualisierungs- und Simulationsbaukasten integriert. In diesem Abschnitt gibt es einen kurzen Überblick über die bereits vorhandenen Arbeiten von Torsten Brodbeck. Im Übrigen werden die Mängel im Hinblick auf die Notwendigkeiten für gemeinsames Lernen von Kommunikationsprotokollen durch elektronisch unterstützes Rollenspiel aufgezeigt.

3.1 Systemarchitektur des HiSAP-Baukasten von Brodbeck

Die Grundarchitektur (Abbildung 3.1) des HiSAP-Baukastens ist in die vier Bereiche Simulationskomponenten, Visualisierungs- & Animationskomponenten, Generatoren und Anwendungsprogramme gegliedert.

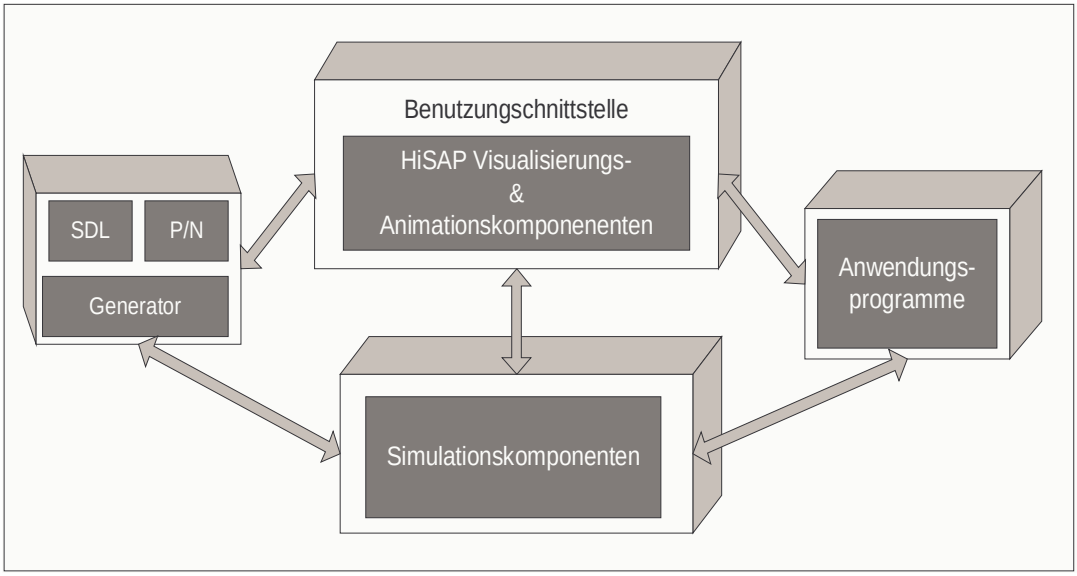


Abbildung 3.1: Architektur des HiSAP-Baukasten [Brod00]

Die vier Bereiche werden durch definierte Schnittstellen gekoppelt. Der wichtigste Bereich betrifft die Simulationskomponenten. Diese Komponenten sind für den Aufbau und Ablauf eines Simulationsmodells zuständig. Die Trennung zwischen den Simulationskomponenten und den Visualisierungs- und Animationskomponenten ermöglicht eine Entkopplung des Simulationsmodells von seiner eigenen Visualisierung. Durch diese Entkopplung kann ein Simulationsmodell unterschiedlich dargestellt werden. Die für eine Simulation erzeugten Visualisierungskomponenten dienen außer der Darstellung auch der Animation und der interaktiven Steuerung durch den Benutzer [Brod00].

Die Komponente Anwendungsprogramme bieten konkrete Simulationsbeispiele, indem sie mit Hilfe von Simulationskomponenten das benötigte Simulationsmodell und, mittels Visualisierungs- und Animationskomponenten, die erwünschte Visualisierung und Benutzerschnittstelle erzeugen. Zusätzlich zu

den Anwendungsprogrammen bestünde die Möglichkeit, mit geeigneten Spezifikationssprachen (z.B. SDL oder Petri Netzen) und einem zugehörigen Generator den Aufbau eines Simulationsmodells und einer eventuellen Visualisierung zu automatisieren [Brod00].

3.2 Klassen zur Erzeugung von Simulationsmodellen

Im HiSAP-Baukasten gibt es zuerst vier Klassen zur Erzeugung von Simulationsmodellen. Diese sind Knoten, Nachricht, Strategie und Verbindung,(engl. *Node*, *Message, Strategy* and *Connection*). Zum Überblick zeigt Abbildung 3.2 die momentan entworfenen Grundkomponenten zur Erzeugung von Simulationsmodellen.

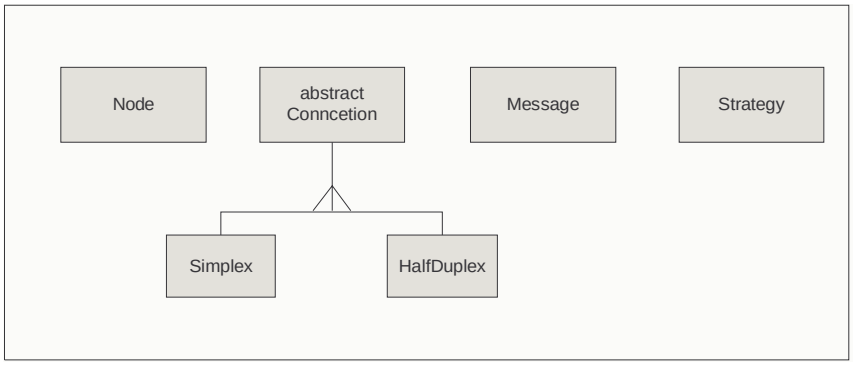


Abbildung 3.2: Komponenten zur Erzeugung von Simulationsmodellen [Brod99]

Ein *Node* beschreibt einen Prozeß, der *Messages* versenden und empfangen kann. Wann *Messages* versendet und welche Aktionen nach Empfang einer *Message* ausgelöst werden sollen, wird durch das Kommunikationsprotokoll bestimmt, welches aus mehreren Strategien besteht. Somit muß jedem *Node* eine *Strategy* zugewiesen werden. *Strategy* ist eine abstrakte Klasse, die eine

Abstraktion unterschiedlicher Kommunikationsprotokolle beschreibt. Jeder *Node* entspricht einer Rolle auf der Visualisierungsebene. Bevor ein *Node* mit einem anderen *Node* kommunizieren kann, müssen beide durch eine *Connection* verbunden werden.

Die Klasse *Connection* ist eine abstrakte Klasse. Sie ist in zwei Kommunikationsarten unterteilt, den *Simplex*- und *HalfDuplex*-Betrieb. *Connections* übertragen *Messages* und ermöglichen die Kommunikation zwischen *Nodes*. Eine *Conncetion* kann stets nur zwei *Nodes* verbinden.

Die Aktionen, die auf einem *Node* geschehen, würden durch die auf diesem *Node* eingesetzten *Strategy* neue Aktionen (Ereignisse) auslösen, die die Visualisierung beeinflussen könnten. Allgemeine sind die Simulation durch Aktionen (Ereignisse) gesteuert.

3.3 Die Kategorie der Visualisierungsmöglichkeiten

Ein Simulationsmodell ist ein Werkzeug zur Generierung von Simulation der Algorithmen und Kommunikationsprotokolle. Um die Ereignisse und Ergebnisse der Simulation auf dem Bildschirm graphisch darzustellen, damit die Studierenden besser verstehen und lernen können, benötigt der HiSAP-Baukasten eine geeignete Visualisierung. Grundsätzlich erscheinen folgenden Darstellungsformen im HiSAP-Baukasten: textbasierende Darstellung und Graphendarstellung.

3.3.1 Textbasierte Darstellung

Die textbasierte Darstellung ist die einfachste Darstellungsform, die Kommunikationsprotokolle zu beschreiben. Es bedarf jedoch einer guten Abstraktionsfähigkeit bzw. Vorstellungskraft, um rein textuell beschriebene Kommunikationsprotokolle zu verstehen. Die textbasierte Informationen für Kommunikationsprotokolle werden in drei Varianten eingesetzt: natürliche Sprache, künstliche Sprache (z.B. Programmiersprachen, **S**pecification **D**escription **L**anguage usw.) und Textdarstellung in tabellarischer Form, um die Protokolle zu beschreiben.

Die drei Varianten sind für die Darstellung aller historischen Informationen nötig, die im Laufe der Simulation erzeugt werden. Solche Informationen sind die Ergebnisse der Simulation und können den Ablauf bzw. die Ergebnisse des Kommunikationsprotokolls charakterisieren. Um den Studierenden zu helfen, den Ablauf der Kommunikationsprotokolle besser zu verstehen, sollen verschiedene textuelle Darstellungsformen mit verschiedenen Zwecken kombiniert werden. Für eine Darstellung von Ereignissen während der Nachrichtenübertragung wird eine formale künstliche Sprache gebraucht, dagegen ist die Tabellendarstellung besonders für einen Ergebnissevergleich (Abbildung 3.3) geeignet, wenn man nur eine Übersicht sehen möchte.

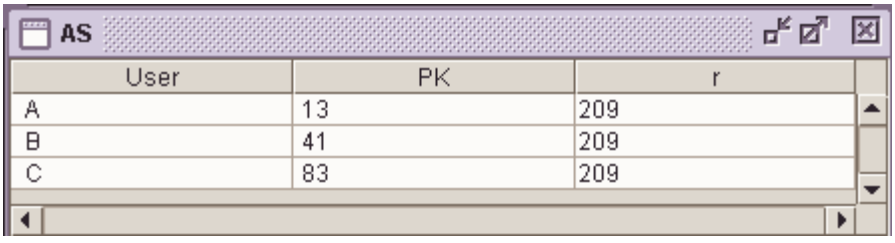


Abbildung 3.3: Tabellendarstellung

Aber für ein kompliziertes, dynamisches und nebenläufiges Kommunikationsprotokoll kann die textbasierte Darstellung nicht genau beschreiben, wie das Kommunikationsprotokoll funktioniert. Problem ist: wenn das Protokoll abhängig von der Topologie ist, ist die textbasierte Darstellung zu schwach dieses Protokoll zu spezifizieren und zu beschreiben. Für diese Sicht wird die Graphendarstellung benötigt.

3.3.2 Graphendarstellung

Graphendarstellungen sind sinnvoll für ein besseres Verständnis und die Einprägsamkeit von Informationen. Da bei graphischer Darstellung die Informationen am effektivsten von der Mehrheit der Menschen wahrgenommen werden können. Am häufigsten werden Topologie- und Sequenzdiagrammdarstellungen zur Darstellung von Kommunikationsprotokollen in graphischer Form benutzt.

Besonders gut zur Lehre von relativ vielen Kommunikationsprotokollen geeignet ist die Topologiedarstellung, da die Verbindungsstrukturen der Rechnernetze bei der Topologie größtenteils wie in der Realität abgebildet werden, und deshalb eine Abstraktion nur in geringem Maße erfordert. Abbildung 3.4 zeigt die Topologiedarstellung von “Routing with backward lerning”.

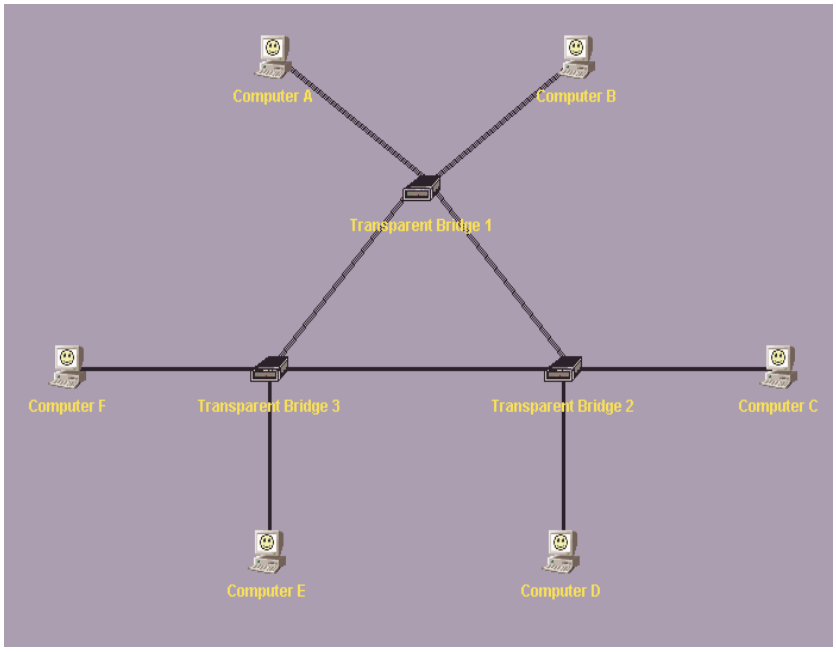


Abbildung 3.4: “Routing with backward lerning” - Topologiedarstellung

Sequenzdiagramme zeigen sehr gut die Ablaufsequenz von Kommunikationsprotokollen. Dabei ist die Sequenz entweder abhängig von der Zeit (Time-Sequence-Diagram) oder von bestimmten Interaktionen (Interaction-Diagram, siehe Abbildung 3.5). Sequenzdiagramme ermöglichen eine gute Darstellung von Historien.

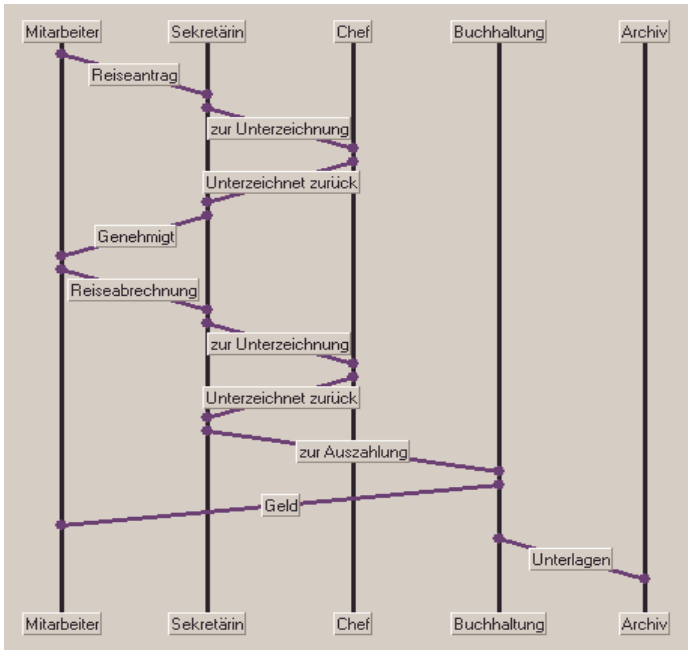


Abbildung 3.5: Interaktionsdiagramm

3.3.3 Fazit

In kurzen Worten ist festzustellen, daß alle besprochenen Visualisierungsmöglichkeiten für die Lehrveranstaltung geeignet sind. Wann und welche Darstellungen besser geeignet sind, ist nach den charakteristische Aspekten eines Protokollbeispiels zu entscheiden. Generell sind die Sequenzdiagramm- und Graphendarstellungen für Kommunikationsprotokolle am besten geeignet. Die Detailinformationen und Historie werden als die abstrakte Textdarstellung dargestellt.

3.4 Interaktionsmöglichkeit

Um den Lerneffekt zu verbessern, ist die interaktive Steuerung des Systems nötig. Vor allem ist es für die Lehre besonders wichtig, dem Benutzer nur so viele Manipulationsmöglichkeiten wie notwendig zur Verfügung zu stellen. Dem Benutzer sollte es auf keinen Fall gelingen durch bestimmte Interaktionen das System zum Absturz zu bringen. HiSAP bietet zwei Manipulationsmöglichkeiten. Eine ist die **allgemeine Steuerung** ohne Manipulation des internen Kommunikationsprotokollablaufs. Die andere ist die **Manipulation des Protokollablaufs**, bei der der Benutzer Aktionen nacheinander anstößt und das eigentliche Protokoll abändern kann.

3.4.1 Allgemeine Steuerung

Bei der allgemeinen Steuerung erhalten die Benutzer die Möglichkeit, die Geschwindigkeit des Protokollablaufs zu ändern, den Protokollablauf jederzeit anzuhalten und wieder fortzusetzen. Ebenso würde eine Undo und Redo Funktionalität möglich sein. Im vorhandenen HiSAP-Baukasten sind diese allgemeinen Steuerungen in der Benutzungsoberfläche integriert (Abbildung 3.6). Da diese Interaktionsmöglichkeiten für die Lehre bei jedem Protokollbeispiel sinnvoll sind, müssen diese unabhängig vom Protokollbeispiel existieren.

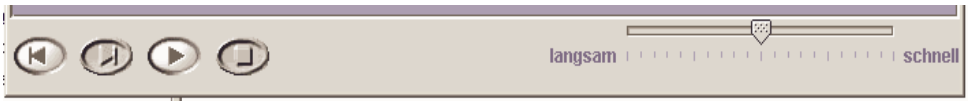


Abbildung 3.6: Benutzerschnittstelle zu allgemeinen Steuerungen

3.4.2 Manipulation des Protokollablaufs

Eine weitere Interaktionsform ist die Manipulation des Protokollablaufs. Bei einem größeren Protokollbeispiel kann die Anzahl der möglichen Protokollabläufen beliebig groß sein. In der Lehrveranstaltung ist es notwendig, dem Benutzer die Manipulation des Protokolls zur Verfügung zu stellen.

Es gibt für die Lehre zwei Möglichkeiten diese Interaktionsform zu realisieren. Die erste Möglichkeit ist, daß die Protokollabläufe im voraus festgelegt und dem Benutzer nur an gewissen Ablaufzweigen die Wahl gegeben wird, welchen Protokollablauf verfolgt werden könnte. Die zweite Möglichkeit ist, daß der Ablauf eines Kommunikationsprotokolls nachsimuliert werden kann

und der Benutzer über definierte Schnittstellen die Möglichkeit bekommt, jederzeit in die Simulation einzugreifen.

Bei der Variante festgelegter Protokollabläufe ist bevorzugt an Vorlesungen oder Seminare gedacht. Der Leiter der Lehrveranstaltungen hat die Möglichkeit, den Ablauf der Kommunikationsprotokolle zu manipulieren, im gleichen Augenblick schauen die Studierenden zu und machen Notizen.

3.4.3 Fazit

Der HiSAP-Baukasten stellt zwei Interaktionsmöglichkeiten zur Verfügung. Die allgemeine Steuerung ist für das Starten und Beenden eines HiSAP-Anwendungsbeispiels nötig. In diesem Zusammenhang ist die allgemeine Steuerung in der Lehrveranstaltung für den Dozent oder den Moderator geeignet, da der Lehrer oder Moderator sich nach der Abstimmung (siehe [Ober01]) entscheiden kann, wann und welches Anwendungsbeispiel gestartet oder beendet werden soll. Dagegen ist die Manipulationsmöglichkeit des Protokollablaufs dem Node zugeordnet. Es bietet eine wesentliche Interaktionsmöglichkeit im Rollenspiel.

3.5 Fehlendes Rollenkonzept im HiSAP-Baukasten

Der HiSAP-Baukasten bietet die Möglichkeit, unterschiedliche Kommunikationsprotokolle und Algorithmen zu simulieren. Sie ist momentan nur lokal aber

nicht verteilt realisierbar. Das Kommunikationsprotokoll ist die formale Darstellung einer Prozedur, die zur Beschreibung einer Kommunikation zwischen zwei oder mehreren Komponenten über einen Kommunikationskanal eingeführt ist. Die Komponenten, die sich an ein bestimmtes Kommunikationsprotokoll halten, können z.B. Client, Server, Router oder Kommunikationskanal sein.

Um das Rollenspiel für gemeinsames Lernen von Kommunikationsprotokollen zu ermöglichen, müssen zuerst die Rollen in HiSAP identifiziert werden. Momentan ist die Identifikation der Rollen des HiSAP-Baukastens für eine Komponente noch nicht realisiert.

Um das verteilte Bearbeiten einer Aufgabe zu ermöglichen, benötigt man für das CSCL (**C**omputer **S**upported **C**ollaborative **L**eraning - Kapitel 2, Abschnitt 2.1) nicht nur ein Rollenkonzept, sondern auch die Koordination zwischen unterschiedlichen Gruppenmitgliedern, wenn die Gruppenmitglieder unterschiedliche Rollen eines Kommunikationsprotokolls wahrnehmen. Deshalb muß der HiSAP-Baukasten erweitert werden, um das koodinierte Spiel des Kommunikationsprotokolls zwischen Gruppenmitgliedern zu ermöglichen.

Das SASCIA-Projekt bietet eine Umgebung, die CSCL unterstützt. Im folgenden Kapitel werden wir im Detail diskutieren, wie der SASCIA-Baukasten arbeitet und welche Funktionalität, die CSCL unterstützen, geboten werden.

4 Ist-Systemanalyse: SASCIA - das Framework

Im vorigen Kapitel wurde eine Einführung in das Projekt HiSAP gegeben. Um die Simulation der Algorithmen und Protokollen verteilt und kollaborativ durchzuführen, benötigt man eine geeignete Infrastruktur. Das SASCIA-System (**S**ystem **A**rchitecture **S**upporting **C**ooperative and **I**nteractive **A**pplications) bietet die Möglichkeit, gemeinsam Anwendungen zu benutzen (im Englischen: application sharing). In diesem Kapitel wird der grundlegende Aufbau dargestellt und danach werden die für diese Diplomarbeit relevanten Konzepte und Komponenten des Projektes SASCIA beschrieben. Insbesondere werden die in derzeitigen Komponenten existierenden Floor-Kontrollmechanismen bewertet und Defizite behandelt.

4.1 Systemarchitektur

Das Projekt SASCIA (**S**ystem **A**rchitecture **S**upporting **C**ooperative and **I**nteractive **A**pplications) ist ein Unterprojekt des Festival Projektes in der Abteilung *Verteilte Systeme* der Universität Stuttgart. Festival steht für **F**ertilizing distibut**E**d **S**ystems **T**o support interact**IV**e **A**pp**L**ications. Es zielt darauf ab, neuere Entwicklungen im Bereich verteilter Systeme im Hinblick auf ihre Verwendung bei rechnergestützter Kooperation zu untersuchen ([FESTIVAL]). Als Unterprojekt zielt das Projekt SASCIA darauf ab, ein Framework zu entwickeln, welches Anwendungen zu CSCL integriert. Einerseits bietet SASCIA die Basisfunktionen zur Realisierung eines Konferenzsystems für elektronisch unterstützte Lehrveranstaltung. Andererseits sollen verschiedene Dienste bzw. Anwendungen integriert werden können.

Momentan ist SASCIA nach Client-Server-Architektur entwickelt. Die Grundkomponenten (Abbildung 4.1) des SASCIA-Projektes sind in fünf Bereiche gegliedert: Kommunikationssystem, Session-Management, Floor-Control-Komponenten, Protokollierungsdatenbank und Anwendungsprogramme.

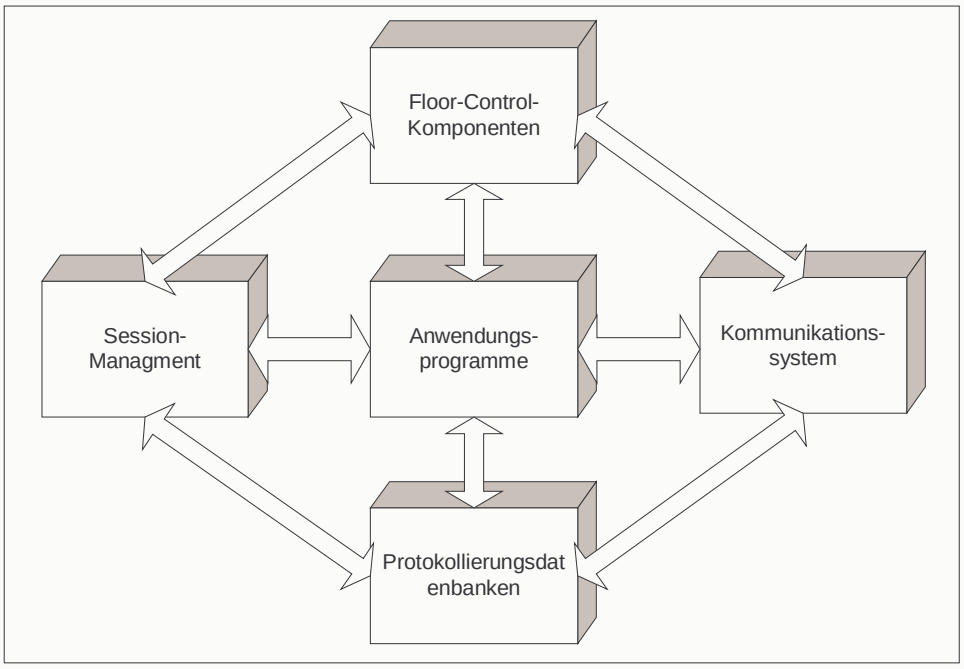


Abbildung 4.1: Grundarchitektur von SASCIA

Diese fünf Bereiche werden über definierte Schnittstellen gekoppelt. Das Kommunikationssystem ist für den Datenaustauch zwischen Komponenten zuständig. Das Session Management ist für die Verwaltung registrierter Sitzungen (z.B starten und beenden einer Sitzung) und Applikationen (z.B. starten und beenden einer Applikation, Möglichkeit zur Auswahl der gestarteten Applikationen) zuständig. Außerdem ist das Session Management auch für die Verwaltung der Teilnehmer verantwortlich. Die Floor-Control stellt einen Mechanismus zur Koordinierung von Benutzeraktionen in verteilten Anwendungen dar. Mit der Floor-Control wird sozusagen das Rede- bzw. Schreibrecht überwacht. Die Protokollierungsdatenbanken zeichnen die mit den Anwendun-

gen erstellten Daten auf. Die Anwendungsprogramme sind die konkreten Applikationen. Die Anwendungsprogramme, Floor-Control und Protokollierungsdatenbank bilden zusammen eine sogenannte Service-Komponente.

Die oben genannten Basiskomponenten werden in den folgenden Abschnitt noch genauer vorgestellt.

4.2 Kommunikationssystem

Als Kommunikation bezeichnen wir den Austausch von Informationen zwischen den Einheiten eines Systems. Das Kommunikationssystem bietet den Komponenten eines Systems eine Schnittstelle zum Kommunizieren und Datenaustausch. Auf Serverseite wird ein Kommunikationskanal, der für den Verbindungsaufbau verwendet wird, für jede Applikation geöffnet und auf Clientseite diesem Kanal beigetreten. Über diesen Kanal erfolgt der Nachrichtenaustausch. Dabei ist das Kommunikationsmuster der Nachrichten von Client aus 1-zu-1 (im Englischen: unicast), vom Server aus rundsenden (im Englischen: broadcast). Die empfangenen Nachrichten werden vom Kanal bei jedem Teilnehmer an einen Kanal-Consumer weitergegeben, der die Nachrichten weiter verarbeitet. Dieser Kanal-Consumer muß vom Empfänger implementiert werden. Das Kommunikationssystem ist zur Zeit durch JSDT (**J**ava **S**hared **D**ata **T**oolkit) realisiert. JSDT ist ein Toolkit, das u.a. zur Erstellung von interaktiven, kollaborativen Applikationen geeignet ist. Da es in Java implementiert ist, ist JSDT plattformunabhängig (siehe [JSDT]).

Außer der Kommunikation zwischen Anwendungsprogrammen wird das Kommunikationssystem für den Datenaustausch zwischen Floor-Control-Server, Floor-Control-Client und Protokollierungskomponenten eingesetzt.

4.3 Session-Management

Die Session-Management-Komponenten stellen sich der Verwaltung, der Sitzung bzw. Lehrveranstaltung zur Verfügung. Die Sitzungsverwaltung bietet die Möglichkeit, Sitzungen in einem Sitzungsverzeichnis zu veröffentlichen, so daß die Teilnehmer sich über die vorhandenen Sitzungen informieren und sich ohne weitere Probleme mit einer Sitzung verbinden können [Somm01].

Als zentrale Komponente in SASCIA besteht die Session-Management-Komponente aus *Session Directory, User Database, Session Database, Server/Client Configuration, Administrator Client, Session Management, Session Management Server/Client, und Benutzer Client.* Die Hauptaufgaben der Session-Management Komponenten sind (vgl. [Somm01]):

- Starten und Beenden des Kommunikationssystems
- Anlagen von Konfigurationsdaten
- Verbindung mit dem *Session Directory,* Eintragung/Austragung der Sitzung im *Session Directory* (wenn gewünscht)
- Server-Konfiguration bzw. Client-Konfiguration laden
- Ausgabe der Serveradresse
- Verbinden der lokalen Benutzerdatenbank (*User Database*) mit der zentralen *User Database* (auf Serverseite)

- Start eines Kanals für die Kommunikation zwischen Session Management und Benutzer Clients
- Start und Ende der benötigten Applikationen
- Benutzerauthentifizierung
- An- & Abmeldungsmöglichkeit des Benutzers
- Entfernen Teilnehmer
- Sperren, Freigeben und Beenden einer Sitzung

In Abbildung 4.2 sind die Hauptkomponenten und ihre Beziehungen untereinander im Detail dargestellt.

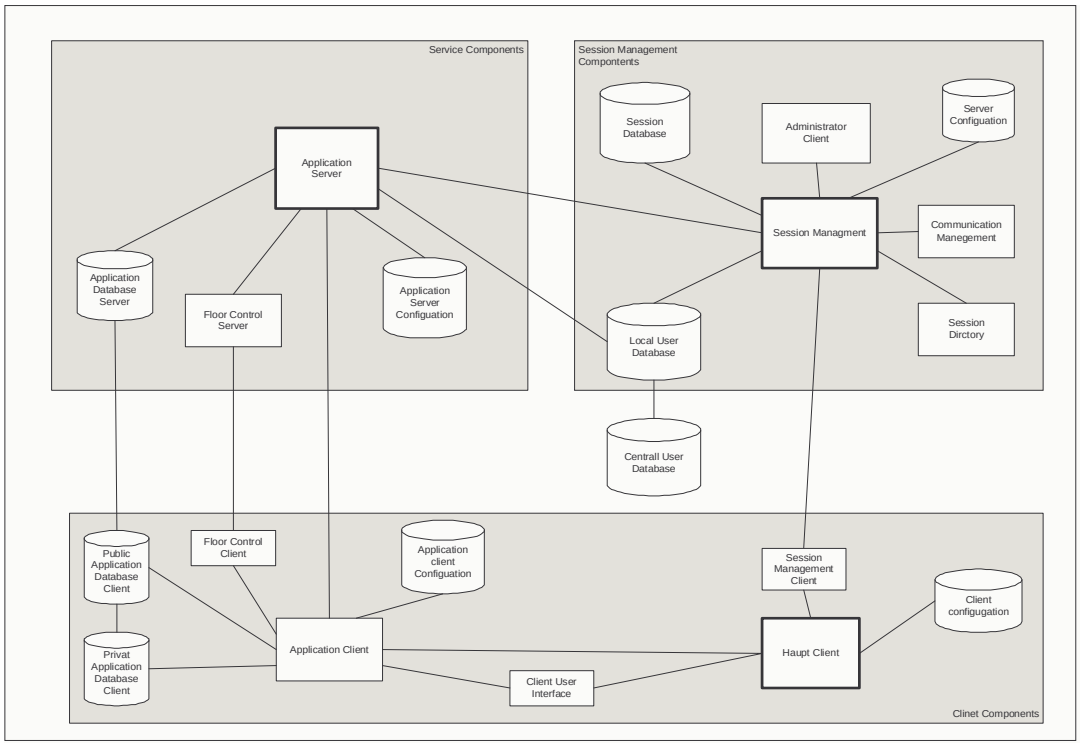


Abbildung 4.2: die Komponenten und ihre Beziehungen in SASCIA

Der *Session-Management-Server* befindet sich auf dem zentralen Präsentationsserver. Auf der Serverseite wird die *Server Config* nach dem Start des *Administrator Clients* geladen. Der *Administrator Client* startet den *Session-Management-Server*. Über die Benutzungsoberfläche, die auf dem *Administra-*

tor Client liegt, werden die grundlegenden Daten der Sitzung (z.B. Sitzungsname, Dozent, Sitzungsort, Sitzungsdatum, Sitzungsbeschreibung usw.) gesammelt. Der *Session-Management-Server* überträgt diese Konfigurationsdaten auf das *Session Directory* und startet das Kommunikationssystem. Der *Session-Management-Server* verbindet die *lokale User Database* mit der *zentralen User Database.* Nach dem Verbinden mit dem *Session Directory* wird die Sitzung dort eingetragen. Zusätzlich zu dem vom Benutzer gewünschten Anwendungsprogramm wird ebenfalls ein spezieller *Application Server* vom *Session-Management* gestartet. Jede Applikation wird durch ein spezielles Rahmensystem gekapselt (Abbildung 4.1). Nach dem oben beschriebenen Ablauf beim Start warten der *Session-Management-Server* und die *Application Server* auf Verbindungswünsche von Clients.

Auf der Clientseite werden die benötigten Daten über die graphische Benutzungsoberfläche eingetragen und eine Verbindung mit dem *Session Directory* erstellt. Die benötigten Daten beinhalten die Identifikation einer Sitzung, die aus der Liste von Sitzungen im *Session Directory* ausgewählt wird, die Serveradresse und den Pfad der Protokollierungsdatenbank. Die Authentifizierung eines Clients erfolgt automatisch über das Kommunikationssystem und die *User Database*, die sich auf Serverseite befindet. Die *User Database* beinhaltet wesentliche Daten eines Benutzers, wie z.B. Name, Authentifizierungsinformationen und statische Priorität innerhalb des *Floor-Control* Systems. Nach der erfolgreichen Authentifizierung tritt der Client dem vom *Session Management* geöffneten Kanal bei und startet das Anwendungsprogramm.

4.4 Floor-Control

Floor-Control ist die Komponente, die für die Überwachung des Rede- bzw. Schreibrechts im System zuständig ist. Der Floor (Rederecht) wird durch ein Token (Abschnitte 2.2.1) repräsentiert. Solange ein Benutzer ein Token besitzt, hat er das Rederecht in einer Sitzung. Bei der Floor-Vergabe ist die implizite Floor-Kontrollstrategie genau so wichtig wie die explizite Floor-Kontrollstrategie. Für eine dynamische Floor-Control ist der Sitzungsleiter zuständig (Abschnitt 2.2.2). Die statische Floor-Control ist durch Abstimmung-Strategie realisiert.

In SASCIA sind folgende Regeln für Floor-Control definiert (Vergleiche mit [Ober01]):

- R1: Für jede Applikation gibt es nur einen Floor.
- R2: Bei der Vorlesung kann der Floor durch den Sitzungsleiter explizit an einen Teilnehmern vergeben bzw. entzogen werden.
- R3: Die implizite Floor-Strategie ist prinzipiell nach “first come, first served” realisiert. In diesem Fall sind die Beantragung und die Rückgabe des Floors möglich.
- R4: Rückgabe oder Weitergabe des Floors an einen bestimmten Teilnehmer sind möglich.
- R5: Durch Abstimmung kann Floor-Control beeinflußt werden.
- R6: Jeder Teilnehmer hat nur eine Stimme.
- R7: Das Gewicht einer Stimme hängt von der Priorität des Teilnehmers ab.

- R8: Beim Systemstart kann einem bestimmten Teilnehmer eine “maximale Priorität” zugewiesen werden, wodurch dieser die Rolle des Vorsitzenden erhält.
- R9: Eine einzelne Stimme mit höherer Priorität zählt mehr als alle Stimmen mit niedrigerer Priorität.
- R10: Ob die Abstimmung erfolgreich ist, hängt von der Anzahl der Pro-Stimmen ab.
- R11: Die Stimme kann an jeden anderen Teilnehmer übertragen werden (Abbildung 4.3)

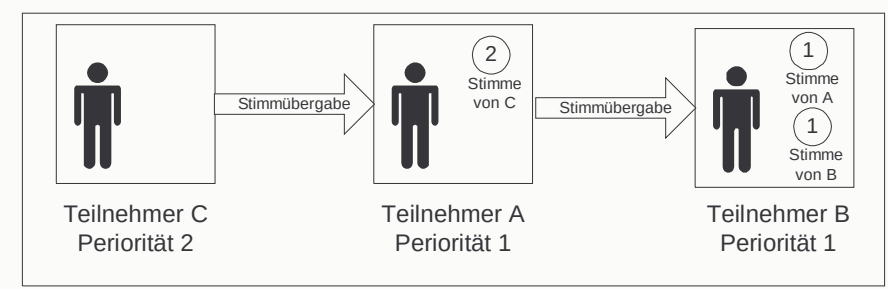


Abbildung 4.3: Prinzip der Stimmübergabe

Momentan besteht die Floor-Control nicht aus einer hierarchischen Architektur. In existierenden Floor-Control-Komponenten ist die Anzahl der Floor-Schichten auf eins festgelegt. Nur die Floorgröße kann verändert werden. Aber es reicht nicht für Integration des HiSAP-Baukastens aus. Im Abschnitt 3.5 haben wir uns vorgestellt, daß es in einem HiSAP-Anwendungsbeispiel viele unterschiedliche Rollen gibt, die einem Node zugeordnet sind. Durch Manipulation der Rollen bzw. Nodes können wir den Ablauf des Kommunikationsprotokolls kontrollieren. Dies ermöglicht das gemeinsame Lernen von Kommunikationsprotokollen durch elektrisch unterstütztes Rollenspiel. Um das Zugriffsrecht den Rollen zu zuordnen, müssen wir Sub-Floors für jedes Hi-

SAP-Anwendungsbeispiel definieren. Im Kapitel 6 Entwurf werden wir darauf im Detail eingehen.

4.5 Protokollierungsdatenbank

Die Protokollierungsdatenbank stellt die Möglichkeit zur Aufzeichnung und Wiedergabe von Lerninhalten zur Verfügung. Es gibt für jede Applikation eine eigene Protokollierungsdatenbank. Die Protokollierungsdatenbank ist dabei in einen Serverteil (*DBAppServer* bzw. *DBAppServerImpl*) und ein Clientteil (*DBAppClient* bzw. *DBAppClientImpl*) aufgeteilt. Der Serverteil läuft auf dem Präsentationssystem und wird vom *Application Server* angesprochen. Der Clientteil läuft bei jedem einzelnen Teilnehmer und wird vom *Application Client* angesprochen. Jede Applikation erhält ihren eigenen Satz an benötigten Protokollierungsdatenbanken. Auf der Clientseite ist die Datenbank in die *Public Application Database Client* und *Private Application Database Client* aufgeteilt. Dementsprechend sind die zu protokollierenden Daten in öffentliche und private Daten unterteilt.

Die öffentlichen Daten sind die Daten, die durch das Kommunikationssystem vom Client aus an den Server gesandt werden und für die Allgemeinheit bestimmt sind. Die privaten Daten sind die Daten, die vom Teilnehmer selbst lokal erzeugt wurden und nicht in Beziehung zu anderen Teilnehmer stehen. Auf der Serverseite werden nur die öffentlichen Daten gespeichert. Auf der Clientseite werden nicht nur die öffentlichen Daten sondern auch die privaten Daten gespeichert. Die privaten und öffentlichen Daten sind mit einem Zeitstempel und einem Reihenfolgeindex in der Datenbank gespeichert. Mit Hilfe des Zeit-

stempels und Reihenfolgeindexes wird die Wiedergabe nach der Lehrveranstaltung möglich.

Um die Synchronisation zwischen *Public Application Database Server* und *Public Application Database Client* zu erhalten, geht dieser öffentliche Datenfluß nur in eine Richtung (Abbildung 4.4). Bevor die Daten in *Public Application Database Server* geschrieben werden, werden sie zuerst durch der Floor-Control kontrolliert.

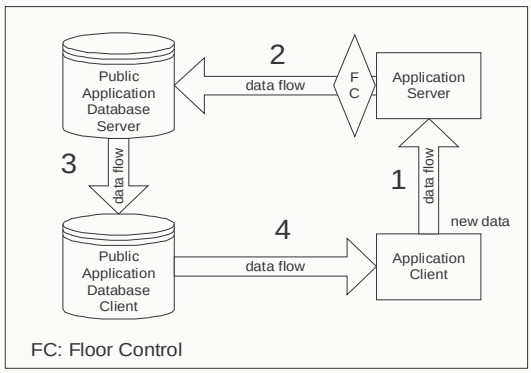


Abbildung 4.4: Datenfluss in der Protokollierungsdatenbank von SASCIA

4.6 Applikation

Die Applikationen, die in das Framework integriert werden, nutzen die bereits beschriebenen Komponenten. Eine Applikation teilt sich in einen Server (*Application Server*) und einen Client (*Application Client*) auf. Der *Application Server* ist für die einzelne Applikation zuständig. Auf ihm wird die eigentliche Dienst-Funktionalität realisiert. Der *Application Client* ist die Verbindung zwischen Teilnehmer und Dienst. Er übernimmt die Kommunikation mit dem jeweiligen Dienst auf dem *Application Server.*

Das Framework bietet einen sogenannten Anwendungsrahmen (*Application Frame*) an, um die Applikation in das Framework zu integrieren. Dieser Anwendungsrahmen stellt die Schnittstelle zum Framework und seinen Komponenten dar. Die Applikation bekommt damit den Zugriff auf das Kommunikationssystem, *Floor-Control* und Protokollierungsdatenbank. Um eine Dienst zu starten und zu beenden, ist das *Session Management* zuständig. Über das Kommunikationssystem kann der *Application Client* eine Verbindung zu seinem *Application Server* herstellen. Mit Floor-Control wird festgelegt, welcher Teilnehmer das Recht hat, einen bestimmten Dienst zu nutzen. Die generierten Anwendungsdaten können auf Server- und Clientseite aufgezeichnet werden und an die Protokollierungsdatenbanken übergeben werden.

Bis jetzt stehen folgende Applikationen in SASCIA zur Verfügung:

- Whiteboard Applikation
- Chat Applikation

Um die spezifische Applikation HiSAP in SASCIA zu integrieren und damit das kollaborative Lernen zu ermöglichen, werden im folgenden Kapitel die Anforderungen an das Soll-System behandelt.

5 Anforderungsanalyse

In den vorigen zwei Kapiteln wurden der HiSAP-Baukasten und SASCIA beschrieben und die Ist-Analyse durchgeführt. In diesem Kapitel wird die Funktionalität des Zielsystems genauer definiert und weiterhin die Anforderungsanalyse behandelt. Anhand verschiedener Testszenarien befaßt sich die Analyse mit der Möglichkeit des Rollenspiels beim gemeinsamen Lernen von Kommunikationsprotokollen. Außerdem wird noch behandelt, welche Bedeutung der Interaktivität beim gemeinsamen Lernen von Kommunikationsprotokollen zukommt.

5.1 Funktionalität des Zielsystems

5.1.1 Grundfunktionen

Im Projektplan wird der geforderte Funktionsumfang wie folgt beschrieben:

- Client-Server-fähige Erweiterung des HiSAP-Baukastens mittels SASCIA
- zentrale Durchführung der Simulation auf dem Server
- dezentrale Visualisierung auf beliebig vielen Clients
- Automatische Übertragung der Ereignisse und Ergebnisse, die die HiSAP-Visualisierungsebene und Simulationsebene brauchen
- Koordination des Zugriffes mehrerer Rollen eines simulierten Kommunikationsprotokolls per Floor-Control, bei Moderation

gesteuert durch Dozent/in, Moderator/in oder anderen berechtigten Client

- Verfassen von Anmerkungen und Veröffentlichung von Anmerkungen an die Clients
- Sichtbarkeit aller registrierten Teilnehmer und ihrer Rollen (*Personal Awareness* - Kap. 2, Abschnitt 2.1.2)
- Sichtbarkeit und Wechselmöglichkeit aller registrierten HiSAP-Anwendungsbeispiele
- Möglichkeit des Rollenwechsels
- Fairness beim Rollenwechsel

5.1.2 Allgemeiner Ablauf des Zielsystems

Der Client, der Moderatorrechte besitzt, initiiert auf dem HiSAP-Simulations-Server die Simulation eines Kommunikationsprotokolls. Die Clients initiieren die HiSAP-Visualisierung auf der Clientseite. Über die Floor-Control werden die Rollen innerhalb der Simulation an die Clients verteilt, so daß diese berechtigt sind, die Simulation entsprechend ihrer jeweiligen Rolle zu steuern. Im Laufe der Simulation werden die Ergebnisse und Ereignisse zur Visualisierung an alle Clients übertragen.

In diesem Zusamenhäng bezieht sich *Personal Awareness* auf die Informationen, wieviel und welche Teilnehmer sich an einer Lehrveranstaltung bzw. Simulation eines Kommunikationsprotokolls beteiligen. Weitere Bestandteile des *Personal Awareness* sind die ID jedes Teilnehmers innerhalb dieser Sit-

zung, die in der *User Database* gespeichert werden, und die Identifikation der Rollen, die die Teilnehmer in der Simulation des Kommunikationsprotokolls wahrnehmen.

5.2 Testszenarien

In diesem Abschnitt wird die Gestaltung der Testszenarien diskutiert. Die Testszenarien, von denen wir im SASCIA-Projekt ausgehen, beschreiben die konkreten Lehrveranstaltugen. Es wird zwischen folgenden Kategorien unterschieden.

5.2.1 Szenario I

Szenario I betrifft den Fall, daß ein/e Dozent/in eine Vorlesung oder ein Seminar abhält. Die Studierenden sitzen im gleichen Raum. Sie sind miteinander über ein drahtloses lokales Netz (**W**ireless **L**ocal **A**rea **N**etwork) verbunden. Der/Die Dozent/in präsentiert eine Vorlesung oder ein Kommunikationsprotokoll auf dem Präsentationsrechner. Die Studierenden hören und sehen zu (Abbildung 3.1).

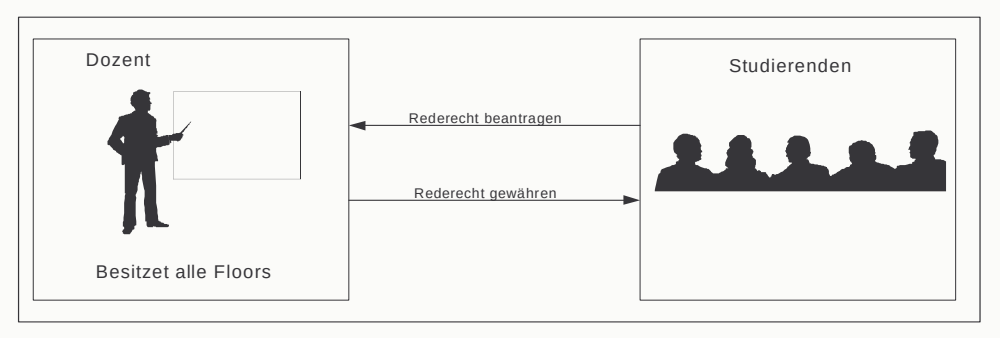


Abbildung 5.1: Szenario I, Vorlesung oder Seminar

In diesem Fall ist der/die Dozent/in ein Moderator. Er besitzt alle Rechte über die ganze Lehrveranstaltung. Die Geschwindigkeit, d.h. wie schnell die Folien gezeigt werden müssen, kann der/die Dozent/in selbst bestimmen. Der Lehrer ist Vortragender. Wenn die Studierenden in der Vorlesung etwas nicht verstehen und Fragen haben, können sie auf herkömmliche Weise (Handheben und Sprache oder Schrift) mit dem Lehrer oder anderen Studierenden kommunizieren.

Gleichzeitig ist die Wiederholbarkeit für einen besseren Lerneffekt von größter Bedeutung. Dazu brauchen wir eine “private” Ansicht, um alle Daten, die in der Lehrveranstaltung erscheinen, zu wiederholen. Der Unterschied zwischen der privaten Ansicht und der üblichen (öffentlichen) Ansicht besteht darin, daß man mittels des *private Application Database Clients* und des lokal geladenen HiSAP-Baukastens die Simulation durchführt und alle Rollenrechte besitzt. In der privaten Ansicht können alle Lernstoffe, z.B. PowerPoint-Daten, Simulation des Protokolls usw. lokal wiederholt werden. Zwischen öffentlicher und privater Ansicht gibt es eine Umstellungsmöglichkeit. Außerdem sollten die HiSAP-Anwendungsbeispiele, die auf einzelnen Clients lokal laufen, die

anderen Clients nicht stören. Das heißt, nur der öffentlich durchgeführte Visualisierungsteil darf mit dem Server kommunizieren.

Aber wenn die Studierenden die private Ansicht während der Vorlesung nutzen, könnte es sein, daß sie der eigentlichen Vorlesung nicht mehr folgen können. Deshalb sollte der Dozent die Möglichkeit haben, die private Ansicht zu sperren, sobald die Frage oder die Unverständlichkeit klar ist.

5.2.2 Szenario II

Szenario II betrifft den Fall, daß mehrere Studierende (Teilnehmer der Gruppe) eine Übungsvorbereitung abhalten. In diesem Fall werden eine oder mehrere Aufgaben gestellt, die u.U. von keinem Studierenden allein gelöst werden könnten. Nur das gesammelte Wissen der Gruppenteilnehmer ermöglicht die Lösung.

Im Szenario II gibt es keinen Vortragenden, der alle Rechte besitzt, also keinen Vorsitzenden. Um einen effektiven Ablauf des Lernprozesses zu ermöglichen, ist ein Moderator notwendig, dessen Rolle von verschiedenen Personen wahrgenommen werden kann. Dadurch wird die Fairness gewährleistet. Der Moderator ist für die Koordination zwischen Teilnehmer und das Ermöglichen von Kooperation zuständig.

5.2.3 Szenario III

Szenario III beschreibt den Fall, daß mehrere Studierenden eine Übung abhalten und Lösungen vortragen sollen. Die Gruppenübung wird vom Vorsitzenden (Übungsleiter) geleitet, dieser übernimmt automatisch die Moderator-Rolle. Dabei ist der Vorsitzende im Besitz der Lösung und trägt diese vor, wenn keiner der Übungsteilnehmer in der Lage ist, sie selbst zu präsentieren. Er ist die Koordinationsstelle. In diesem Fall soll der Wechsel des Moderators möglich sein, damit nicht alle Lösungsansätze auf einmal versucht werden, sondern nur der Lösungsansatz des derzeitigen Moderators. Sind die anderen Teilnehmer mit diesem Ansatz nicht zufrieden, oder haben eine vermeintlich sinnvolle weiterführung, sollte die Rolle des Moderators einer andern Person zugesprochen werden können. Eine Abstimmung führt den Wechsel des Moderators herbei und macht den Sieger der Abstimmung zum neuen Moderator.

5.3 Resultierende Anforderungen

Gemäß der Klassifikation von Groupware (Kapitel 2, Abschnitt 2.1) lassen sich in dieser Arbeit noch die Interaktivität beobachten. Wichtiges Merkmal im SASCIA-System, das eine HiSAP-Anwendung integriert, ist die Interaktivität zwischen den Studierenden und dem simulierten Kommunikationsprotokoll. Hoch interaktive Lernumgebungen ermöglichen es den Studierenden, ihren Lernprozeß ganz gezielt auf ihre individuellen Bedürfnisse hin auszurichten.

Das System sollte folgende Interaktivität anbieten:

- Einfluß des Studierenden auf die Lerninhalte, z.B. Auswahl des Kommunikationsprotokolls
- interaktive Gestaltung der Visualisierung, die von den Studierenden durch HiSAP-Simulationskomponenten verändert werden kann
- Simulationen, bei denen die Studierenden die Ausgangswerte, die von Rollen abhängig sind, eingeben können
- Möglichkeit, eigene Notizen zu schreiben usw.

Im folgenden Abschnitt werden wir die spezielle und allgemeine Anforderungen an unserem System darstellen.

5.3.1 Spezielle Anforderung

Diese Diplomarbeit basiert auf zwei Systemen: SASCIA (ein Framework für gemeinsame Nutzung von Anwendungen) und HiSAP (die Applikation, ein Tutorial-Tool zur Simulation von Algorithmen und Protokolle). Die folgenden Ziele sollen erreicht werden.

Teilnehmerverwaltung

Es muß die Möglichkeit bestehen, jede teilnehmende Personen innerhalb des Systemablaufs eindeutig zu identifizieren und ihr eine Rolle (Moderator/Akteur/Beobachter - siehe auch Kapitel Grundlage, Abschnitt 2.1.3) zuzuweisen. Ohne eine derartige Möglichkeit ist die Vergabe der Zugriffsrechte und damit der geordnete Ablauf einer Vorlesung oder Übung nicht gewährleistet. Gleich-

zeitig sollte das System die Möglichkeit bieten, die in den konkreten HiSAP-Anwendungsbeispielen erscheinenden Rollen eindeutig zu identifizieren. Weiterhin sollte für Teilnehmer sichtbar sein, welche Personen an einer Sitzung oder Protokollsimulation teilnehmen (**Personal Awareness**).

Zuordnung der anwendungsspezifischen Rollen

Die Rollen, die in den HiSAP-Anwendungsbeispielen spezifiziert werden, müssen den Teilnehmern zugeordnet werden. Ohne diese Zuordnung ist das gemeinsame Lernen von Kommunikationsprotokollen nicht gewährleistet.

Datentransfer

In der Lehrveranstaltung ist eine integrierte Datenübertragungsmöglichkeit notwendig. Diese wird zum Beispiel benutzt, um die Ereignisse der Kommunikationsprotokolle, die die HiSAP-Visualisierungsebene auslösen und von der HiSAP-Simulation gebraucht werden, an den Server und von dort eventuell die Simulationsergebnisse und neu erzeugte Ereignisse, die von den HiSAP-Visualisierungen gebraucht werden, an die Endgeräte zu übertragen. Es sollte eine Möglichkeit angeboten werden, den HiSAP-Baukasten zu übertragen, falls dieser nicht auf der Clientseite besteht.

Applikationssteuerung

Eine integrierte Schnittstelle zur Fernsteuerung der auf dem Server ablaufenden Simulation ist erforderlich. Diese Schnittstelle zur Applikationssteuerung soll nicht nur die Fernsteuerung von einzelnen Simulationen integrieren, sondern auch eine Auswahl aus verschiedenen Simulationen von Algorithmen und Kommunikationsprotokollen. Durch die Schnittstelle zur Auswahl aus verschieden Simulation könnte der berechtigte Teilnehmer die Simulationsteile

auf dem Server starten. Durch die Fernsteuerung von einzelnen Simulationen können die berechtigten Teilnehmer den Ablauf eines konkreten Kommunikationsprotokolls manipulieren.

Floor Control und Sub-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. Ein Sub-Floor dient dazu, die Rollen, die mit den Kommunikationsprotokollen verbunden sind, zu manipulieren und zu steuern.

Jeder Sub-Floor entspricht einer *Node* (Kapitel HiSAP-das Simulationssystem, Abschnitt 3.5). Bevor eine neue Applikation gestartet wird, müssen die Sub-Floors definiert sein. Um das zu gewährleisten, müssen bei dem HiSAP-Baukasten jede Rolle Identifiziertet werden. Jeder Sub-Floor muß von nur einem Teilnehmer besetzt werden. Da wenn mehrere Teilnehmer den selben Sub-Floor besetzen, könnte ein gegenseitiger Ausschluss auftreten.

Möglichkeit für Anmerkungen

Es ist von besonders großer Bedeutung, daß das Konferenzsystem eine “Anmerkungsfunktionalität” unterstützt, beziehungsweise daß eine vorhandene Whiteboard-Funktion entsprechend erweitert werden kann, um die folgenden Aktionen abzudecken [Ober02]:

- Anmerkungen in Form von Freihandzeichnung und einfacher Textdarstellung 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

Da die HiSAP-Visualisierungen in das Whiteboard eingebunden sind und das Whiteboard diese Funktionen bietet, sind damit Anmerkungen möglich.

Fairness

Das System soll die Möglichkeit anbieten, daß jeder Teilnehmer das Recht hat, seine eigene Meinung, z.B. Wechsel des Moderators, Startwünsche eines konkreten HiSAP-Anwendungsbeispiels, auszudrücken.

Unterstützung der verteilten Durchführung der HiSAP-Anwendung

Eine client-server-fähige HiSAP-Anwendung ist erforderlich. Der HiSAP-Bauksten muß erweitert werden, um ihn an die Client-Server-Fähigkeit anzupassen.

5.3.2 Allgemeine Anforderung

Außer den oben genannten speziellen Anforderungen sind folgende allgemeine Anforderungen zu berücksichtigen.

Offenheit

Die Integration neuer HiSAP-Simulationskomponenten, sowie die Erweiterung bestehender Funktionalität des HiSAP-Baukasten muß durch das System unterstützt werden.

Konsistenz

Die Benutzerschnittstelle der HiSAP-Anwendung sollte im Erscheinungsbild und in der Bedienung konsistent sein, um die Akzeptanz des Endsystems zu verbessern. Dazu muß auch die Sprache in der Benutzungsschnittstelle konsistent sein.

Flexibilität

Die Benutzungsschnittstelle sollte auf unterschiedliche Plattformen angepaßt werden. Da Java plattformunabhängig ist, eignet es sich für eine Realisierung. Außerdem soll die Internationalisierung der Sprache, die auch ein Anforderung an HiSAP ist, anpaßbar sein.

6 Entwurf

Nach der Anforderungsanalyse ist nun der Entwurf des zu erweiternden HiSAP-Baukastens und des SASCIA-Framework möglich. Nach der Erstellung der Grobarchitektur werden schrittweise die Erweiterungen von HiSAP-Baukasten im Detail modelliert. Weiterhin wird die Sub-Floor-Control behandelt. Anschließend werden die Applikationsrahmen auf die HiSAP-Anwendungsbespiele angepaßt.

6.1 Grobarchitektur für HiSAP-Anpassung

Die vier Komponenten in HiSAP, Simulationskomponente, Visualisierungs- & Animationskomponente, Anwendungsprogramm und Generator, werden durch definierte Schnittstelle gekoppelt. Um den HiSAP-Baukasten als Applikation in SASCIA zu integrieren und das kollaborative Abspielen von Kommunikationsprotokollen zu ermöglichen, muß die Simulation zentral beim Server durchgeführt werden. Dadurch können die Visualisierungsteile der simulierten Kommunikationsprotokolle auf der Clientseite beliebig oft dargestellt werden. Ein anderer Vorteil ist, wenn die Simulation zentral auf dem Server durchgeführt wird, ermöglicht die Synchronisation zwischen Simulationskomponenten einfach zu gewährleiten. Da die Simulationskomponenten sind eindeutig aber nicht dupliziert. Abbildung 6.1 stellt die Grobarchitektur von HiSAP im Client-Server-System dar.

Die gegenwärtig existierenden Simulationssysteme verwenden größtenteils die ereignisgesteuerte Simulation. Durch solche Ereignisse (*ActionEvents*) sind die Visualisierung und Simulation in HiSAP getrennt. Die *ActionEvents*, die auf Visualisierungsebene ausgelöst werden, rufen die Strategie auf, die auf den einzelnen Knoten (*Node*) eingesetzt wurde, um die Simulation eines Kommunikationsprotokolls durchzuführen. Solche *ActionEvents* sind Auslöser für die Simulation des Protokolls. Sie ermöglichen die Entkopplung zwischen Visualisierung und Simulation. Nach dem Simulationsschritt werden neue *ActionEvents* von den Strategien erzeugt und wieder an die Visualisierungskomponenten weitergeleitet. Alle *ActionEvent*s, die Simulation und Visualisierung austauschen, werden durch einen Kommunikationskanal, der von SASCIA angeboten wird, zwischen Simulationsserver und Visualisierungsclient übertragen.

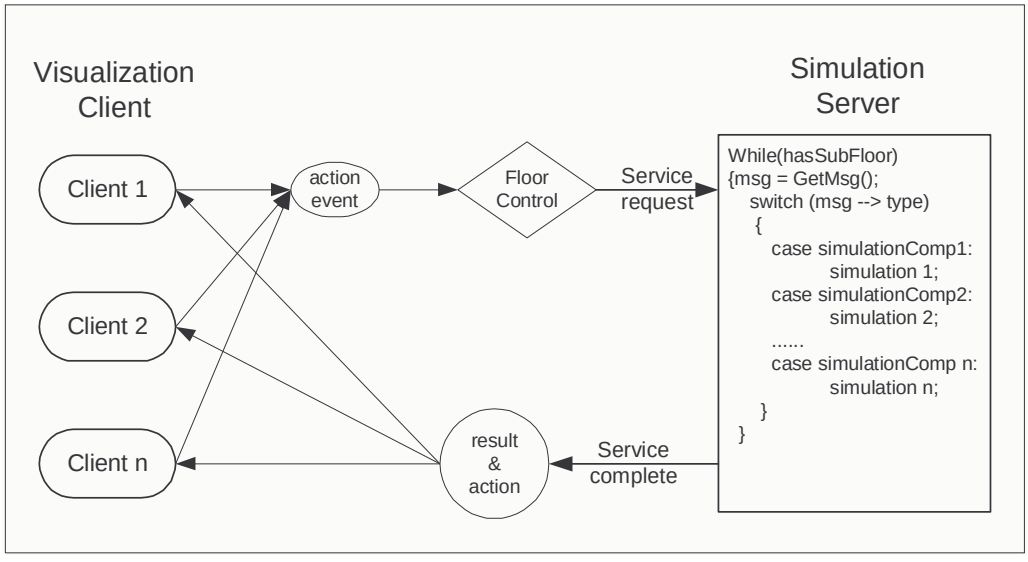


Abbildung 6.1: Grobarchitektur

Im folgenden Abschnitt werden wir schrittweise die Erweiterungen des Rollenkonzepts in HiSAP entwerfen. Es müssen dabei drei Varianten berücksich-

tigt werden: Rollenidentifikation, die Anzahl der Rollen und *ActionEvent* einer Rolle.

6.2 Rollenkonzept von HiSAP

Im Kapitel 3, Abschnitt 3.4 haben wir die Klasse zur Erzeugung von Simulationsmodellen vorgestellt. Dies waren Knoten, Verbindung, Nachricht und Strategie (im Englischen: *Node*, *Connection*, *Message*, und *Strategy*). Prinzipielle können die Instanzen von *Node, Connection*, *Message* und *Strategy* Rollen zugeordnet werden, da aber *Connection* nur die physische Verbindung simuliert, die vom System automatisch durchgeführt wird, brauchen wir die Instanzen von *Connection* den Rollen nicht zuordnen. Beim *Node*, dem die *Strategy* zugewiesen wird und der *Messages* versenden und empfangen kann, macht es Sinn, ihm einer Rolle zuordnet werden. Da die *Messages* nach der Simulation von den empfangenen und versandten *Nodes* aufgerufen und dargestellt werden, brauchen wir die *Messages* den Rollen auch nicht zuzuordnen. Die *Strategy* beschreibt den Ablauf einer simulierten Kommunikationsprotokoll. Dieser Ablauf wird in der Historie dargestellt (vgl. [Bord99]). Damit ist es auch nicht notwendig, der *Strategy* eine Rolle zuzuordnen.

Nach der Erklärung, welche Klassen Rollen in HiSAP zugeordnet werden, diskutieren wir nun, wie das Rollenkonzept in HiSAP realisiert werden kann.

6.2.1 Rollenidentifikation

Der HiSAP-Baukasten bietet in seiner bisherigen Form keine Schnittstelle zum Realisieren des Rollenkonzepts zwischen mehreren Benutzern, d.h. User A und User B sind nur zwei unterschiedliche Instanzen auf der Simulationsebene. In diesem Protokoll sind sie nicht auf der Simulationsebene identifizierbar. Um das Rollenkonzept zu unterstützen, sollen zwei neue Methoden: *setRole* und *getRole* eingefügt werden. Dadurch können künftig die Rollen auf der Simulationsebene identifiziert werden.

Es muß beachtet werden, daß die Rollen im HiSAP-Baukasten nicht gleich den allgemeinen Rollen in der Floor-Control sind, da die Rollen im HiSAP-Baukasten nach *Nodes* spezifiziert werden, aber die Rollen in Floor-Control die dynamische Eigenschaft eines Teilnehmers darstellen.

6.2.2 Anzahl der Rollen

In einem simulierten Kommunikationsprotokoll gibt es nur eine bestimmte Anzahl von Rollen. Um die Rollen in der computerunterstützten Lehrveranstaltung richtig zu verteilen, muß der HiSAP-Baukasten nicht nur die Identifikation von Rollen sondern auch die Anzahl der Rollen anbieten. Die Anzahl der Rollen ist ein Kriterium zum Auswählen der Rollenzuteilungsstrategie (Abschnitt 6.4) und zum Erstellen des Floor-Control-Servers/-Clients. Bei der Rollenzuteilung gibt es zwei Möglichkeiten: die Anzahl der Rollen ist größer als die Anzahl der Teilnehmer, oder es gibt weniger Rollen als Teilneh-

mer. Wenn die Anzahl der Rollen größer ist, kann ein Teilnehmer mehrere Rollen im gleichen Simulationsablauf des Kommunikationsprotokolls übernehmen. Dagegen könnte im Fall, daß weniger Rollen als Teilnehmer vorhanden sind, die Simulation so oft wiederholt werden, daß jeder Teilnehmer das Protokoll wenigstens einmal mitspielen konnte.

6.2.3 ActionEvent einer Rolle

Die Visualisierung und Simulation im HiSAP-Baukasten sind durch *ActionEvents* gekoppelt. Die *ActionEvents* sollen durch einen Kommunikationskanal zwischen dem Simulationsserver und den Visualisierungsclients übertragen werden. Berücksichtigen wir die Eigenschaft des Kommunikationskanals, daß nur die serialisierbaren Objekte übertragen werden können, müssen die *ActionEvent* transformiert werden.

Die *ActionEvents* stammen von bestimmten Komponenten, die entweder Visualisierungskomponenten oder Simulationskomponenten sind. Solche Komponenten nennen wir *Source*. Mittels der Methode *getSource* von *ActionEvent* können wir die *Source* feststellen. Aber die *getSource*-Methode von *ActionEvent* in JAVA gibt in der Regel ein Objekt zurück, das eine Instanz einer Subklasse von *Component* ist. Diese Objekte können sehr groß werden, da sie die gesamte dargestellte Grafik einer Komponente enthalten können und daher im AWT (siehe [JAVA]) transient sind. Das bedeutet, daß sie bei der Serialisierung eines *ActionEvents Source* nicht mitserialisiert werden und daher nicht durch den Kommunikationskanal übertragen werden können.

Kommen die *ActionEvents* von der Visualisierungsebene, muß die Simulationseben wissen, welche Quelle das *Event* ausgelöst hat. Daher werden wir die Quelle durch ein *String* identifizieren. Ein weiterer Vorteil ist, daß dieser *String* nur die wesentlichsten Daten beinhaltet und dadurch bei einer Übertragung das Netz nicht stark belastet.

In der Gegenrichtung sollen auch die Simulationsergebnisse durch den Kommunikationskanal übertragen werden. Sie sollen als Parameter mit den *ActionEvents* an den Client weitergeleitet werden.

Ein ähnliches Problem tritt bei der Klasse *Message* auf. Die Klasse *Message* beinhaltet einige Informationen, die von der Visualisierung gebraucht werden. Deshalb müssen die *Messages* auch an Client bzw. Server übertragen werden. Weil im HiSAP-Baukasten die Klasse *Message* nicht serialisierbar ist, sollten die Informationen von *Message* vor dem Versand isoliert und einzeln weitergeschickt werden.

Nachdem wir festgestellt haben, welche Erweiterungen in HiSAP nötig sind, um das Rollenkonzept umzusetzen, ist der kooperative Zugriff mehrerer Teilnehmer auf unterschiedliche Rollen eines Kommunikationsprotokolls zu diskutieren. Im folgenden Unterkapitel wird daher zunächst die Vergabe des Zugriffsrechts von Rollen durch SASCIA behandelt.

6.3 Erweiterung der Floor Control von SASCIA

Floor Control regelt in synchroner Groupware, wie und wann die Teilnehmer in einer verteilten Umgebung bei allgemeinen Aufgaben auf Anwendungen gleichzeitigem Zugriff könnten.

Aus der vorhergehenden Definition (Kapitel 2 Grundlage, Abschnitt 2.3) lassen sich folgende Begriffe ableiten, daß deren Modellierung eine Voraussetzung für den Entwurf der Floor Control bzw. Sub-Floor Control in SASCIA darstellt.

6.3.1 HiSAP-Anwendung und HiSAP-Anwendungsbeispiele

HiSAP-Anwendung ist ein Oberbegriff. Es handelt sich um ein Programmobjekt, das in SASCIA integriert werden soll. Mit Hilf der HiSAP-Baukasten, der in HiSAP-Anwendung eingesetzt ist, können wir verschiedene Kommunikationsprotokolle simulieren und darstellen. HiSAP-Anwendungsbeispiele sind die konkreten simulierten Kommunikationsprotkolle.

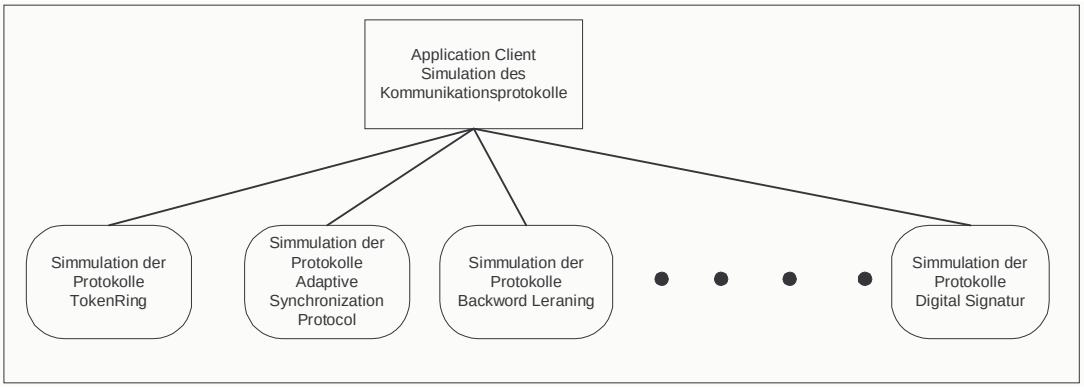


Abbildung 6.2: HiSAP-Anwendungsbeispiele in SASCIA

Wie in Abbildung 6.2 gezeigt, kann die HiSAP-Anwendung mehrere einzelne HiSAP-Anwendungsbeispiele beinhalten. Die Ein- und Ausgaben der HiSAP-Anwendung bzw. eines konkreten HiSAP-Anwendungsbeispiels sollen durch Floor-Control bzw. Sub-Floor-Control koordiniert werden. Die HiSAP-Anwendung bzw. HiSAP-Anwendungsbeispiele bedienen sich dabei der Funktionen der Floor-Control, weshalb innerhalb des Floor-Mechanismus keine Modellierung der HiSAP-Anwendung an sich stattfinden muß. Jedes HiSAP-Anwendungsbeispiel besitzt einen Floor. Die Floor-Größe ist auf eins festgelegt, das bedeutet, daß jeweils nur ein Moderator die Rechte zur allgemeinen Steuerung des HiSAP-Anwendungsbeispiels übernehmen darf.

6.3.2 Sub-Floor

Abhängig von konkreten HiSAP-Anwendungsbeispielen gibt es mehrere anwendungsspezifische Rollen. Zu jeder Rolle wird ein Sub-Floor eingerichtet. Um das Rollenspiel zu ermöglichen, müssen die Teilnehmer die Sub-Floors beantragen. Abbildung 6.3 zeigt die Sub-Floor-Struktur des Kommunikationsprotokolls “Zweiwege-Authentifikation mittels asymmetrischer Verschlüsselung”. In diesem Simulationsbeispiel sind vier mögliche Protokolle integriert: Cached Key, Cached Key (non secure), Authentification Server Cached Key (without caching), Authentification Server Cached Key (non secure). Die erste Variante hat drei Rollen: User A, User B und Intruder. Bei der dritten Variante gibt es vier Rollen: User A, User B, Intruder und Authentifiction Server.

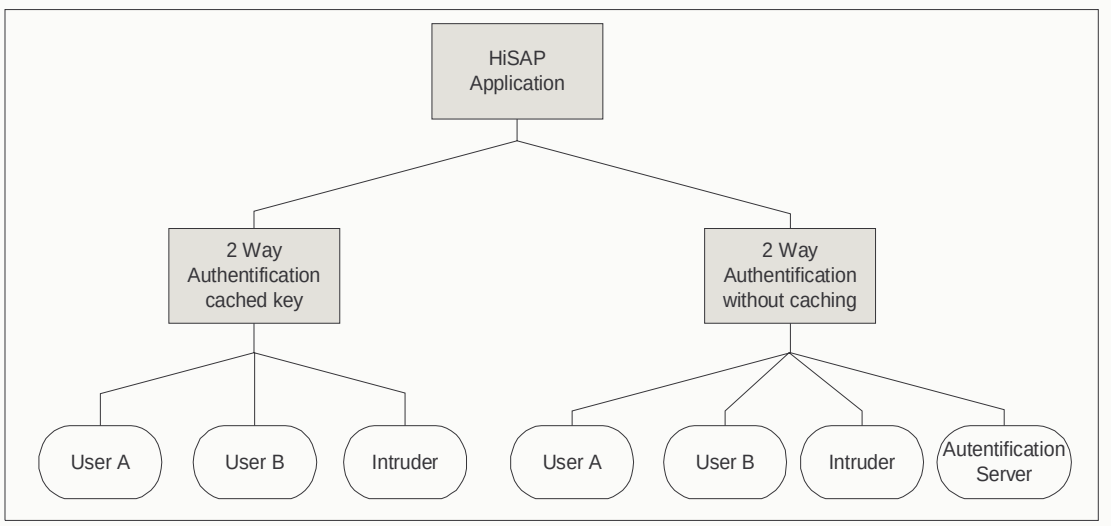


Abbildung 6.3: Floor -- Baumstruktur

Für die im HiSAP-Baukasten identifizierten Rollen werden jeweils ein Sub-Floor-Server und ein Sub-Floor-Client geöffnet. Sie verwalten die Zugriffsrechte dieser Rolle. Alle Sub-Floor-Server und Sub-Floor-Clients werden jeweils in den zugehörigen Floor-Servern und Floor-Clients als Sub-Floors in einer Art Baumstruktur gespeichert.

6.3.3 Teilnehmer

Die Definition des Teilnehmer-Begriffs ist nicht Gegenstand der Floor-Control. Sie wird vielmehr aus einer anderen Komponente (Teilnehmer-Datenbank) importiert (Kapitel 4).

Der Teilnehmerbegriff muß dabei die folgenden Anforderungen erfüllen: 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 beim Systemstart ein Teilnehmer bestimmt werden, welcher den Vorsitz übernimmt. Dadurch erhält dieser eine dynamische Priorität, welche größer ist als die aller anderen Teilnehmer [Ober01].

Speziell für diese Arbeit kann jeder Teilnehmer eine oder mehrere Rollen übernehmen oder die Rolle wechseln.

6.3.4 Rollen

Die Definition des Rollen-Begriffs in Bezug auf Floor-Control von SASCIA ist nicht gleich den Rollen in HiSAP. Eine Rolle in der Floor-Control ist die dynamische Eingenschaft eines Teilnehmers, die bestimmt, auf welche Operationen der Floor Control dieser Teilnehmer Zugriff hat.

Allgemeine Rechte aller Teilnehmer

Jeder Teilnehmer hat eine Stimme (Kapitel 4, Abschnitt 4.4) und könnte seine Stimme abgeben. Aus den beschriebenen Test-Szenarien empfiehlt es sich, die folgenden Gegenstände von Abstimmung zu unterscheiden:

- Moderatorfrage: Wer soll die Moderator-Rolle einnehmen?
- Dienstspezifischer Antrag: Soll ein HiSAP-Anwendungsbeispiel ausgeführt oder beendet werden?

Im Kapitel 2 haben wir drei Rollen definiert: Moderator, Akteur und Beobachter. Im bezug auf das Projekt SASCIA, das den HiSAP-Baukasten inte-

griert, werden wir im folgenden Abschnitt genau beschreiben, welche Recht jede Rolle hat und für was jede Rolle zuständig ist.

Moderator

Ein Teilnehmer, der die Moderatorenrolle besitzt, kann das Rederecht vergeben und entziehen, sowie die Vergabestrategie (Abschnitt 2.2) bestimmen. Außerdem besitzt ein Moderator alle Rechte zur allgemeinen Steuerung der HiSAP-Anwendung. Der Moderator ist eine Koordinationsstelle der HiSAP-Anwendung bzw. des HiSAP-Anwendungsbeispiels. Bei der Vorlesung übernimmt der Dozent automatisch die Moderator-Rolle. Bei Übungen übernimmt zunächst der erste angemeldete Teilnehmer die Moderator-Rolle. Im Laufe der Übungen kann der Moderator nach der Abstimmung aller Akteure neu gewählt werden. Außerdem hat der Moderator folgende zusätzliche Rechte:

- Start eines HiSAP-Anwendungsbeispiels nach der Abstimmung aller Akteure (Kapitel 4, Abschnitt 4.5) und allgemeine Steuerung der HiSAP-Anwendung

  Wenn die Mehrheit aller Akteure ein HiSAP-Anwendungsbeispiel mitspielen möchte, muß der Moderator das HiSAP-Anwendungsbeispiel starten. Der Moderator behält den Floor.

- Rollenverteilung

  Vor dem Start eines konkreten HiSAP-Anwendungsbeispiels müssen alle Rollen besetzt werden. Es dürfen keine freien Rolle übrig bleiben. Wenn eine Rolle noch frei ist, würde ein Teil der Simulation des Kommunikationsprotokolls fehlen. Die Simulation kann nicht voll durchgeführt werden. Es dürfen auch nicht, mehrere Teilnehmer die gleiche Rolle besitzen. Wenn mehrere Teilnehmer die gleiche Rolle besitzen, taucht der gegenseitige Ausschluss auf. Falls ein Konflikt besteht oder es noch freie Rollen gibt, muß der Moderator die Verteilung der Rollen koordinieren (Details siehe Abschnitt 6.4.2 Szenario II).

- Rücknahme der Sub-Floors

  Um Fairness zu gewährleisten, kann der Moderator vor Beendigung eines HiSAP-Anwendungsbeispiels die Sub-Floors zurücknehmen. Damit wird vermeiden, daß ein egoistischer Teilnehmer den Floor beliebig lange blockiert. Voraussetzung ist, daß der Moderator fair ist.

Akteur

Außer dem Moderator teilen wir die Teilnehmer noch in zwei weitere Gruppen: Akteur und Beobachter. Der Teilnehmer, der ein Zugriffsrecht haben möchte und einen Sub-Floor beantragt hat, ist ein Akteur.

Ein Akteur kann außer den allgemeinen Rechten (d.h. Abstimmung) auch die folgenden Aktionen ausführen:

- Rückgabe des Sub-Floors beim Beenden eines HiSAP-Anwendungsbeispiels
- Manipulationsmöglichkeit des Protokollsablaufs eines konkreten HiSAP-Anwendungsbeispiels im Rahmen seiner Rolle
- Möglichkeit zum Rollenwechsel

Nach der Rückgabe des Rederechts ist ein Akteur wieder Beobachter.

Beobachter

Der Beobachter ist ein passiver Teilnehmer. Ein Beobachter stellt keinen Antrag auf einen Rollen-Floor (Sub-Floor). Damit hat er kein Zugriffsrecht auf die Rollen eines HiSAP-Anwendungsbeispiels. Dank der allgemeinen Rechte kann ein Beobachter einen Sub-Floor beantragen um zum Akteur zu werden.

6.4 Umsetzung am Beispiel der Test-Szenarien

In der Anforderungsanalyse (Abschnitt 5.2.1) haben wir drei Test-Szenarien vorgestellt und die zwei Probleme Rollenzuteilung und Rollenwechsel dargestellt. In diesem Unterkapitel werden die möglichen Lösungen entsprechend der drei Test-Szenarien beschrieben.

6.4.1 Szenario I

Im Szenario I hat der Dozent alle Zugriffsrechte aller Rollen der HiSAP-Anwendungsbeispiele. In diesem Fall übernimmt der Dozent die Moderator-Rolle und koordiniert die Zugriffsrechte. Normalerweise dürfen die Studierenden in diesem Szenario den Protokollsablauf nicht manipulieren. Aber es besteht die Möglichkeit, daß die Studierenden auf die Rollen des HiSAP-Anwendungsbeispiels zugreifen und manipulieren, indem sie Fragen stellen können und den Protokollsablauf beeinflussen. Die Studierenden können nur die Sub-Floors des HiSAP-Anwendungsbeispiels beim Dozenten beantragen. Der Dozent entscheidet, wer den Sub-Floor übernehmen darf. Nach dem Ausüben muß der Studierende den Sub-Floor an den Dozenten zurückgeben oder der Dozent entzieht dem Studierenden den Sub-Floor. Der Dozent kann gleichzeitig mehrere Sub-Floors an mehrere Studierenden vergeben, wenn es nötig ist.

Für die später kommenden Teilnehmer können mit Hilfe der Protokollierungsdatenbank die HiSAP-Anwendungsbeispiele wiedergeholt werden. Sobald sie sich an dieser Vorlesung angemeldet haben und beigetreten sind, haben sie die

allgemeinen Rechte und das Recht zum Beantragen einer Rolle in einem HiSAP-Anwendungsbeispiel, genauso wie die anderen Teilnehmer.

6.4.2 Szenario II

Im Szenario II gibt es keinen Dozent, der alle Recht besitzt. Das heißt, es gibt auch keinen Vorsitzenden. Um den richtigen Ablauf der Übung zu gewährleisten, muß zuerst ein Moderator ausgewählt werden. Standardmäßig ist am Anfang der Moderator derjenige, der sich zuerst angemeldet hat, da am Anfang mit der Abstimmung über die Auswahl eines Moderators ein nicht unerheblicher zeitlicher Aufwand verbunden ist. Der erste angemeldete Teilnehmer soll von Session Management benachrichtigt werden. Im Laufe der Übung kann ein anderer Teilnehmer als Moderator mittels Abstimmung gewählt werden.

In einem konkreten HiSAP-Anwendungsbeispiel ist die Anzahl der Rollen vorher bekannt. Dabei kommen zwei Probleme vor: Rollenzuteilung und Rollenwechsel. Das Rollenzuteilungsproblem besteht zum Beispiel, wenn nicht alle Rollen dieses HiSAP-Anwendungsbeispiels wahrgenommen sind und deshalb die Simulation dieses Protokolls im kollaborativen Lernen nicht durchgeführt werden kann. Einerseits wenn es mehr Studierende als Rollen gibt, und jeder Studierende am HiSAP-Anwendungsbeispiel als Akteur teilnehmen möchten, entsteht ein Rollenwechselproblem, andererseits mittels Rollenwechsel wird Fairness gewährleistet. In diesem Fall können wir die Simulation des

Protokolls so oft mit anschließendem Rollenwechsel wiederholen, bis jeder Studierende am HiSAP-Anwendungsbeispiel teilnehmen konnte.

Rollenzuteilungsproblem

Sei im laufenden Kommunikationsprotokoll die Anzahl der Knoten (entsprechend den Rollen) *n*, die Anzahl der Akteure *m*. Es treten zwei Möglichkeiten beim Rollenzuteilungsproblem auf:

1. Wenn es einen Konflikt beim Beantragen einer Rolle (Sub-Floor) gibt, wird eine Konfliktliste der Rollen automatisch vom System erstellt. Jeder Teilnehmer wird benachrichtigt. Falls $m >= n$ erfolgt die Vergabe des Rederechts nach dem *"first come, first served"* Prinzip. Falls $m < n$, können die Teilnehmer sich dann unterhalten und entscheiden, wer welche Rolle nimmt. Die Rollen werden neu verteilt.
2. Wenn es noch freie Rollen gibt, die beantragt werden können, meldet das System dies jedem Teilnehmer. Falls $m >= n$ könnten andere Teilnehmer diese Rollen beantragen. Falls $m < n$ muß ein Akteur mehrere Rollen übernehmen.

Rollenwechsel im Spiel

Um Fairness zu gewährleisten, sollten die Teilnehmer, die ein Zugriffsrecht besitzen, nicht immer die gleiche Rolle spielen. Um andere Teilnehmer mitspielen lassen zu können, müssen die Sub-Floors an andere Teilnehmer explizit übergeben oder an das System implizit zurückgeben werden.

1. Wenn der Ablauf normal beendet (Abschnitt 3.3.1, allgemeine Steuerung) wurde, wird das Rederecht automatisch zurückgegeben und dem nächsten Beantragenden übergeben.

2. Beim Rücksetzen (Reset) des Ablaufs des Kommunikationsprotokolls wird das Zugriffsrecht zurückgegeben und an den nächsten Beantragenden übergeben.

6.4.3 Szenario III

Im Szenario III gibt es einen Vorsitzenden, um die Übungslösung vorzutragen und zu leiten. Der Übungsleiter übernimmt automatisch die Moderator-Rolle. Der Rollenzuteilungs- und Rollenwechselvorgang ist wie in Szenario II geschrieben.

6.5 Anwendungsrahmen (Integration aller Möglichkeiten)

Als wohldefinierte Schnittstelle zu den Diensten, die das SASCIA-System einer Anwendung anbietet, wurde ein “Rahmensystem” entwickelt. Es kapselt den Zugriff einer Anwendung auf die folgenden Dienste von SASCIA:

- Benutzerdatenbank
- Kommunikationskanäle
- Floor-Control bzw. Sub-Floor-Control
- Protokollierungsdatenbank
- Konfiguration der Anwendung
- Abstimmung
- Anwendungsschnittstelle

Der Anwendungsrahmen ist als Client-Server-Architektur konzipiert. Auf der Serverseite gibt es *AppServerFrame*, auf der Clientseite *AppClientFrame*. Abbildung 6.4 zeigt den Ablauf bei Anwendungsstart auf Serverseite. Die Anwendungsschnittstelle ist als Client-Server-Architektur entwickelt worden: auf der Serverseit ist *AppServerInterface*, auf der Clientseite *AppClientInterface*. Es müssen eigene *AppServer* und *AppClients* für HiSAP implementiert werden.

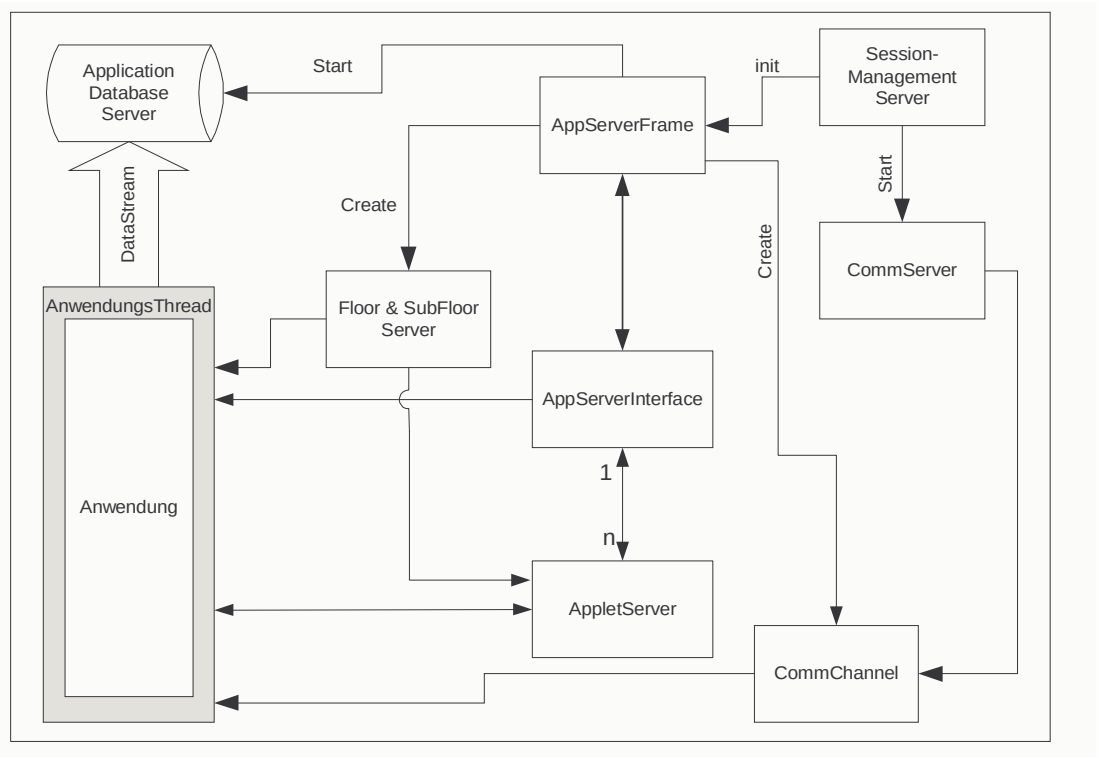


Abbildung 6.4: Ablauf innerhalb der Serverseite

Am Anfang öffnet der *SessionManagementServer* den *AppServerFrame (init)* und *CommServer (start)*. Nach der Initialisierung des *AppServerFrame*, werden der *FloorServer* und ein *CommChannel (create)* erzeugt, der *Application Server*, der die *AppServerInterface* implementiert, und der *Application Database Server (Start)* geöffnet, um die Protokollierungsdaten zu speichern. Der *FloorServer* verwaltet den Floor. Der *CommChannel* ist für die Datenübertragung zuständig. Jede Anwendung hat ihren eigenen *FloorServer*, *CommChan-*

nel, *Application Server* und *Application Database Server* und läuft auf ihrem eigenen Thread.

Wie das im Abbildung 6.2 erstellte Konzept dargestellt kann ein HiSAP-Application-Server beliebig viele *Applet Server* öffnen. Auf jedem *Applet Server* läuft der Simulationsteil der HiSAP-Anwendung in seinem eigenen Thread. Zwischen dem HiSAP-Application-Server und *Applet Server* gibt es eine 1:n-Beziehung. Mit Hilfe der im Anwendungsrahmen integrierten Floor-Control-Komponente öffnet der *Applet Server* Sub-Floor-Server, um die Sub-Floors eines konkreten HiSAP-Anwendungsbeispiels zu verwalten.

Auf Clientseite startet der *SessionManagementClient* den *AppClientFrame* und den *CommClient*. Nach der Initialisierung des *AppClientFrame*, wird ein *FloorClient* und der *Application Client*, der das *AppClientInterface* implementiert, geöffnet. Um die öffentlichen Protokollierungsdaten aus dem Server zu speichern und diesen Daten von der Clientseite lesen zu können, wird die *Public Application Database Client* geöffnet. Der *Application Client* tritt dem *CommChannel* bei, welcher von *AppServerFrame* erzeugt wurde. Mittels des *FloorServer*s verwaltet der *FloorClient* den Floor auf der Clientseite. Der *FloorClient* und der *Public Application Database Client* sind in der Anwendung identifiziert. Je nach der Anzahl der von *Application Server* gestarteten HiSAP-Anwendungsbeispiele kann der *Application Client* entsprechende Applet Client öffnen. Mit Hilfe der im Anwendungsrahmen integrierten Floor-Control-Komponente öffnet der *Applet Client* Sub-Floor-Clients, um die Sub-

Floors eines konkreten HiSAP-Anwendungsbeispiels zu verwalten. Abbildung 6.5 zeigt den Ablauf auf der Clientseite.

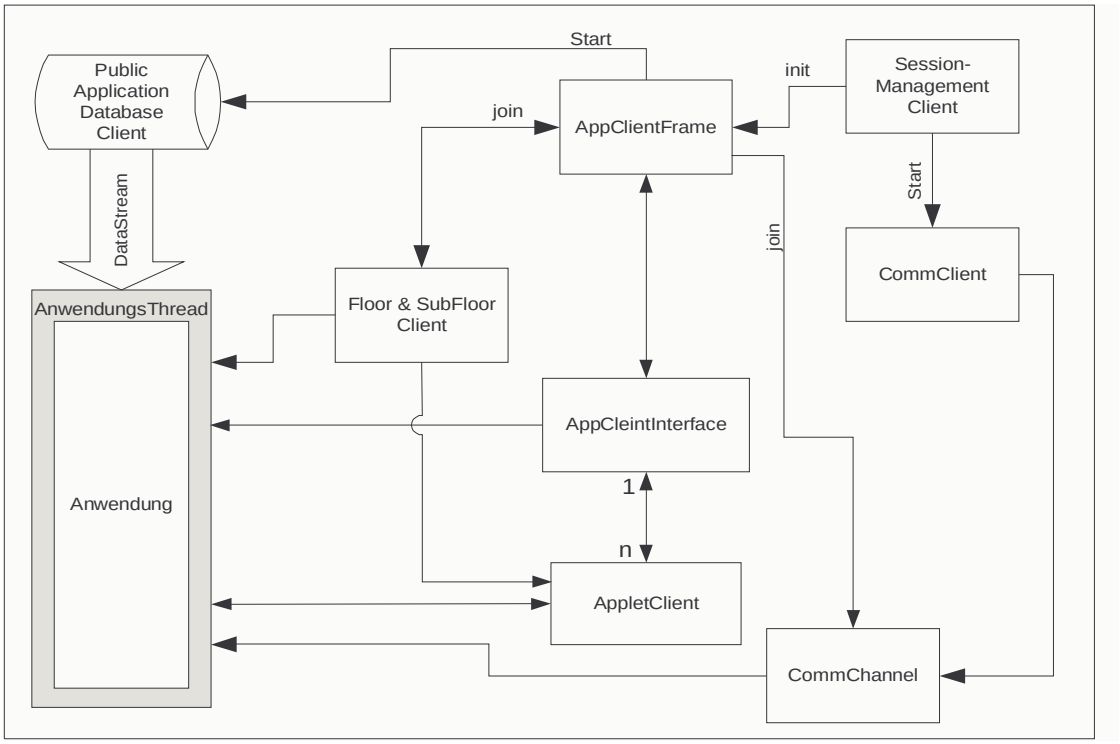


Abbildung 6.5: Ablauf innerhalb der Clientseite

Zusätzlich muß der *ClientFrame* alle verfügbaren HiSAP-Anwendungsbeispiele anbieten, damit alle Teilnehmer sie sehen und auswählen können.

Nachdem nun die Umsetzung des Rollenkonzeptes und Sub-Floors in HiSAP und SASCIA entworfen wurde, soll im folgenden Kapitel die Implementation beschrieben werden.

7 Implementierung & Bewertung

Im vorherigen Kapitel haben wir das Grundkonzept des Systems erstellt. In diesem Kapitel wird die Implementation beschrieben, um die zu erweiterende Funktionalität zu verwirklichen.

7.1 Benutzungsschnittstelle für HiSAP-Anwendung

Als eine wohldefinierte Schnittstelle zu den Diensten, die das SASCIA-System einer Anwendung anbietet, wurde ein Anwendungsrahmen entwickelt. Es kapselt die Zugriffe der Anwendung auf die folgenden Dienste: Benutzerdatenbank, Kommunikationskanäle, Floor Control, Protokollierungsdatenbank und Konfiguration der Anwendung.

Nachdem in Kapitel 6, Abschnitt 6.5 geschriebenen Ablauf werden *AppServerInterface* und *AppClientInterface* von *AppServerFrame* bzw. *AppClientFrame* geöffnet. In diesen zwei Interfaces stehen die Methoden *start* und *notifyStop* zur Verfügung, die von einer konkreten Anwendung implementiert werden müssen. Eine GUI (Benutzungsschnittstelle) wird mittels *start* dargestellt. Typischerweise ist die GUI auf der Serverseite nur eine einfache Schnittstelle, um die nötigen Konfigurationsdaten (z.B. Floorgröße) einzugeben und den *AppServer* zu beenden. Auf der Clientseite soll die GUI wohldefiniert und

benutzerfreundlich sein. Für den Spezialfall der Einbindung von HiSAP-Anwendungen sieht die GUI auf Clientseite aus wie in Abbildung 7.4 dargestellt.

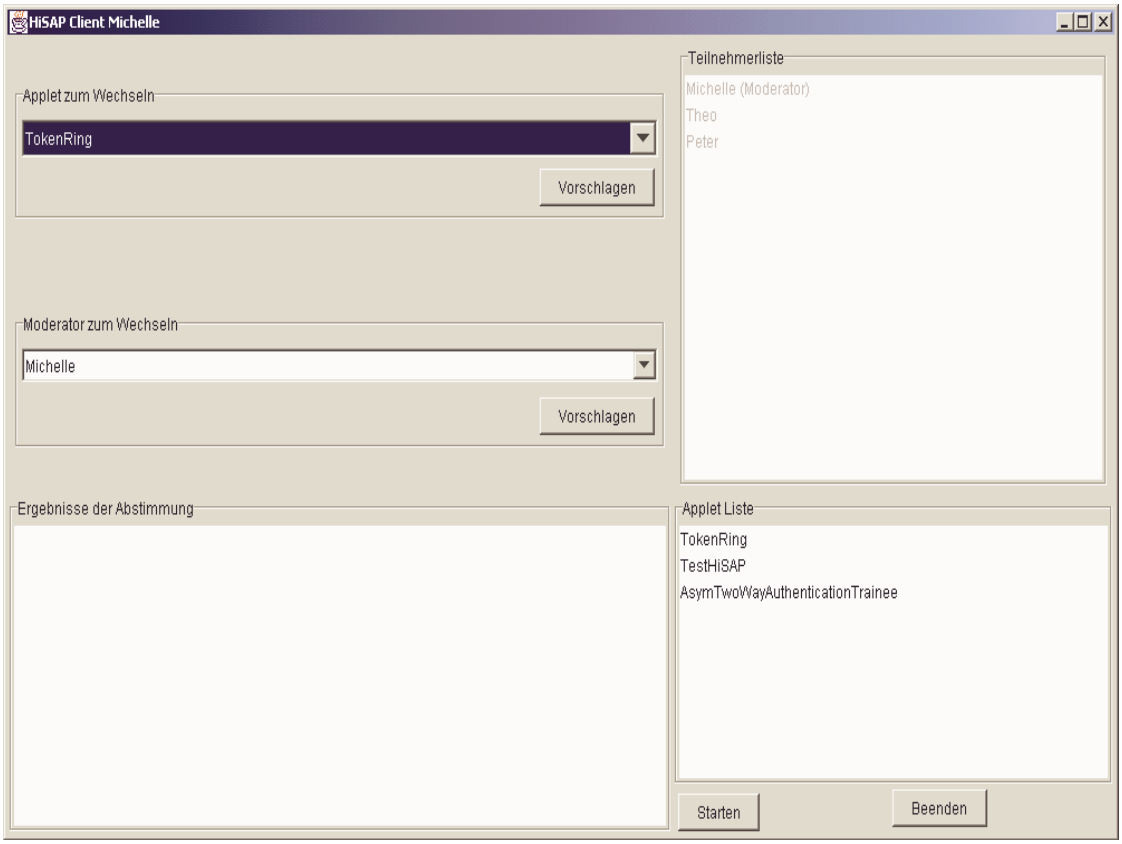


Abbildung 7.1: Benutzungsschnittstelle auf Clientseite

Diese Benutzungsschnittstelle ist in vier Teilbereiche gegliedert: Voting, Voting-Ergebnisse, Teilnehmerliste und HiSAP-Anwendungsbeispiellist (Applet Liste). Der Voting-Bereich kapselt die Zugriffe der HiSAP-Anwendung auf den Voting-Dienst, der die Fairness innerhalb einer Lehrveranstaltung ermöglicht. Die zwei Abstimmungen werden in der Ecke oben links dargestellt. Zwei Abstimmung sind vorgeschlagen: Moderator-Wechsel und Applet-Wechsel (siehe Kapitel 6, Abschnitt 6.3.4). Die möglichen Moderatorkandidaten sind in einer *ComboBox* dargestellt. Um die Fairness zu gewährleisten sind alle Teilnehmer Moderatorkandidaten. Die Kandidatenliste des Moderators ist die Teil-

nehmerliste, die vom *FloorServer* geliefert wird. Nach dem Antrag eines neuen Moderators werden die anderen Teilnehmer benachrichtigt und ein Voting-Fenster mit zwei Knöpfen “Pro” und “Contra” wird dargestellt (Abbildung 7.5). Damit können sie diesem Antrag ihre Stimme geben. Die zweite Variante ist die HiSAP-Anwendungsbeispielsliste. Die alle zu verfügten HiSAP-Anwendungsbeispiele werden in der *Application Configuration* gespeichert und an *AppServerInterface* bzw. *AppClientInterface* geliefert.

Abbildung 7.2: Voting-Fenster

Das Ergebnis des Votings erscheint dann im Voting-Ergebnisse-Bereich, das in der Ecke unten links liegt. Das Ergebnis beinhaltet den Inhalt des *Polls*, ob dieser *Poll* gewonnen hat, wieviel Stimmen gegen den *Poll* sind und wieviel dafür, usw. In diesem Zusammenhang ist ein Poll der Gegenstand der Abstimmung (Kapitel 6, Abschnitt 6.3.4). Dadurch können die Studierenden siene eigene Wünsche ausdrücken.

Die Teilnehmerliste zeigt, wer sich bei dieser HiSAP-Anwendung angemeldet hat. Sie liegt in der Ecke oben rechts. Zusätzlich wird in der Teilnehmerliste gezeigt, wer die Moderator-Rolle inne hat.

Im HiSAP-Anwendungsbeispielliste-Bereich werden alle verfügbaren HiSAP-Anwendungsbeispiele in einer *Applet-Liste* gezeigt. Durch die Knöpfe *Starten* & *Beenden* kann der Moderator das jeweilige HiSAP-Anwendungsbeispiel aktivieren und anhalten. Die anderen Teilnehmer haben keine Möglich-

keit ein HiSAP-Anwendungsbeispiel zu starten oder zu beenden. Nachdem ein konkretes HiSAP-Anwendungsbeispiel gestartet wurde, wird auf der Clientseite die entsprechende Benutzungsschnittstelle dargestellt, wie sie im nächsten Abschnitt genauer behandelt wird.

7.2 Benutzungsschnittstelle eines HiSAP-Anwendungsbeispiels

Da auf der Serverseite nur die Simulation läuft, gibt es keine Visualisierungsdarstellung auf der Serverseite. Die Benutzungsschnittstelle eines konkreten HiSAP-Anwendungsbeispiels auf Clientseite bietet eine private Ansicht und eine öffentliche Ansicht. Mittels der Klasse *JTabbedPane* [JAVA] kann man zwischen privater und öffentlicher Ansicht umschalten.

Im Gegensatz zur öffentlichen Ansicht laufen in der privaten Ansicht die Visualisierung dieses HiSAP-Anwendungsbeispiels und die Simulation lokal auf Clientseite. Abbildung 7.3 stellt eine private Ansicht der Benutzungsschnittstelle eines konkreten HiSAP-Anwendungsbeispiels, “Zweiwege Authentifikation mittels asymmetrischer Verschlüsselung”, dar.

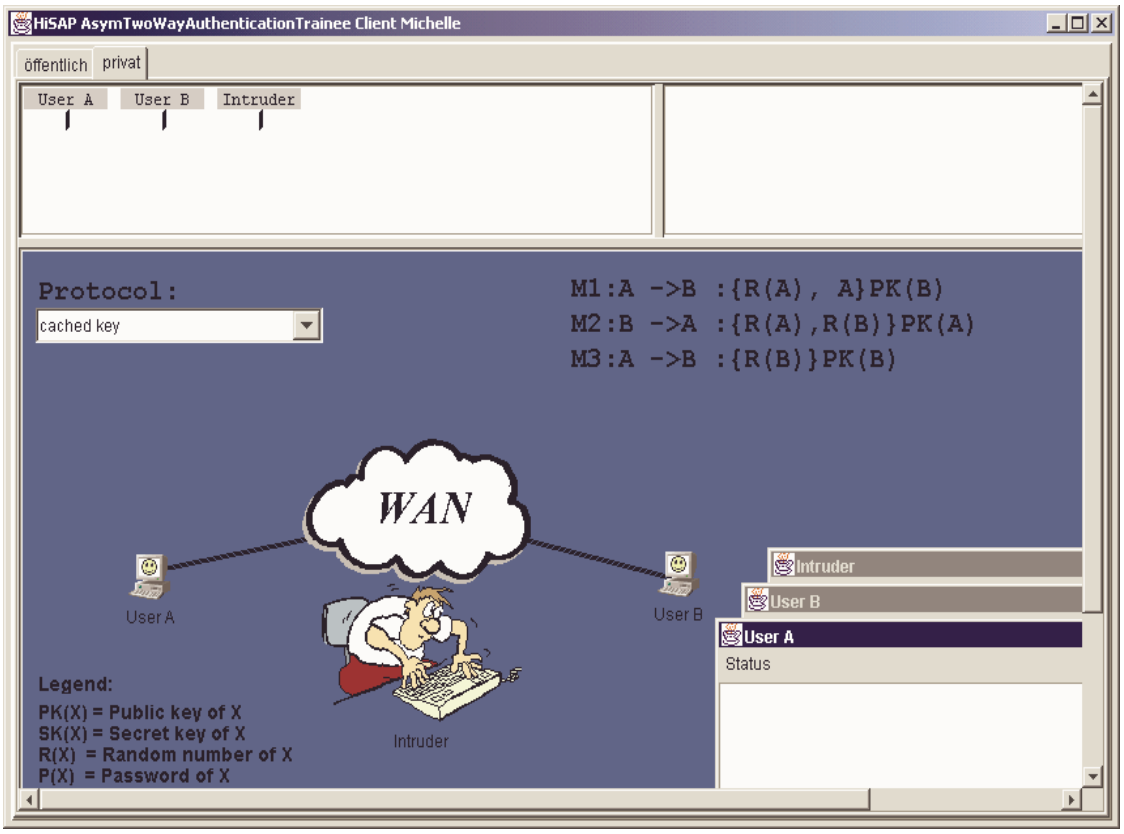


Abbildung 7.3: Private Ansicht der Benutzungsschnittstelle

Abbildung 7.4 stellt die öffentliche Ansicht dar. Die öffentliche Ansicht ist in drei Teilbereiche aufgeteilt: Teilnehmerliste, Rollenliste und Visualisierungsbereich. Die Teilnehmerliste liegt in der Ecke oben links. Sie zeigt alle Teilnehmer und die Informationen, wer der gegenwärtige Sub-Floorinhaber ist. Die Rollenliste stellt alle aktuellen Rollen dar, die in der Topologiedarstellung des HiSAP-Kommunikationsprotokolls erscheinen. Die aktuelle Rollenliste ist durch *getRolesList* der *AppletInterface* erreichbar. Mit zwei Knöpfen “fordern” und “aufgeben” kann ein Teilnehmer (vgl. Kapitel 6 Entwurf, Abschnitt 6.4) einen oder mehrere Sub-Floors beantragen oder zurückgeben. Im Visualisie-

rungsbereich wird das HiSAP-Anwendungsbeispiel dargestellt, dessen Simulation zentral auf den Server läuft.

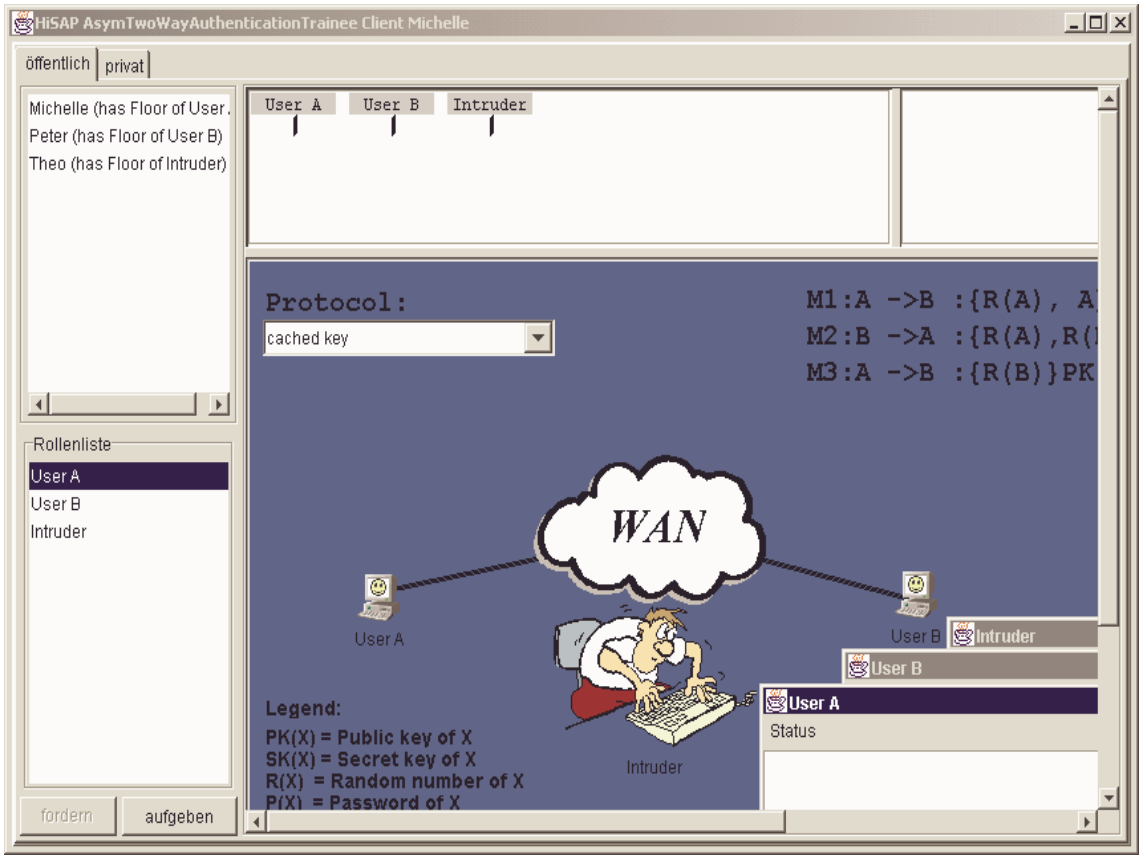


Abbildung 7.4: Öffentliche Ansicht der Benutzungsschnittstelle

Die Teilnehmer, die auf der Teilnehmerliste erscheinen, können die Rollen der Rollenliste auswählen und beantragen. Nach der Rollenzuteilung (vgl. Kapitel 6, Abschnitt 6.4) startet der Moderator mittels der Benutzerschnittstelle zur allgemeinen Steuerung (Abbildung 3.2) das entsprechende HiSAP-Anwendungsbeispiel. Sobald ein Teilnehmer den Sub-Floor hätte, wäre er ein Akteur. Die Komponenten, die dieser Rolle zugeordnet (vgl. Abschnitt 7.4) sind, sind nur bei dem berechtigten Teilnehmer aktiv. Damit kann er alle Manipulationsmöglichkeiten, die diese Rolle bietet, nutzen, um den Ablauf des Kommunikationsprotokolls zu kontrollieren.

Wenn ein Teilnehmer ein anderes Applet spielen will, kann er jederzeit zur Benutzungsschnittstelle aus Abbildung 7.1 wechseln, um einen Poll zu erstellen und abstimmen zu lassen. Sobald dieser Poll gewonnen hat, beendet der Moderator das momentan laufende Applet, zieht alle Sub-Floor zurück und startet das neue Applet.

7.3 Erweiterung für SASCIA: hisapapplet

Um die oben dargestellte Benutzungsschnittstelle zu realisieren und HiSAP in SASCIA zu integrieren, definieren wir ein Package *hisapapplet*. Abbildung 7.5 stellt alle Klassen in diesem Package dar.

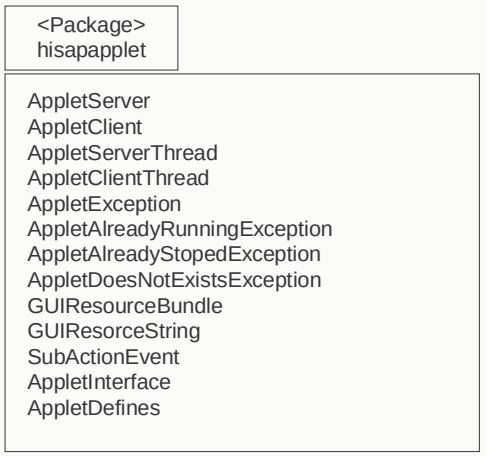


Abbildung 7.5: Package hisapapplet

Start des HiSAP-Anwendungsbeispiels

Wie im Entwurf festgestellt müssen, um den HiSAP-Baukasten in SASCIA einsetzen zu können, ein *Application Server* und ein *Application Client* definiert werden. Im Unterschied zu anderen Applikationen kann es zu der HiSAP-Applikation mehrere HiSAP-Anwendungsbeispiele geben. Um die verschiedenen HiSAP-Anwendungsbeispiele in ihrem eigenen Rahmen zu starten und

laufen zu lassen, benötigt man zusätzlich für jedes HiSAP-Anwendungsbeispiel noch einen Serverteil (*AppletServer*) und einen Clientteil (*AppletClient*). Der *AppletServer* wird vom *Application Server* gestartet. Auf der Clientseite wird der *AppletClient* von dem *Application Client* gestartet. Der *AppletServer* und *AppletClient* laufen in ihrem eigenen Thread (*AppletServerThread* und *AppletClientThread*) und bieten gleichzeitig jedem Benutzer eine Schnittstelle (siehe Abschnitt 7.2). Dadurch wurde alle benötigte Funktionalität in dieser Benutzungsschnittstelle integriert (z.B. Floor Control, Kommunikation, Protokollierungsdatenbank), um gemeinsames Lernen von Kommunikationsprotokollen durch elektronisch unterstütztes Rollenspiel zu ermöglichen.

Erstellung der Sub-Floor-Server/-Client

Nach der Initialisierung des HiSAP-Anwendungsbeispiels werden auf der Serverseite die Sub-Floor-Server und auf der Clientseite die Sub-Floor-Clients geöffnet. Die Anzahl der Sub-Floors ist gleich der Anzahl der Rollen im HiSAP-Anwendungsbeispiel. Der Name des Sub-Floors ist gemäß einer Listenstruktur erstellt, z.B “AppName.AppletName.RoleName”. Dabei wird der *AppName* (*Application Name*) vom *SessionManagement* geliefert und der *AppletName* (der Name des HiSAP-Anwendungsbeispiels) wird aus der *Application Configuration* ausgelesen. Der Name der Rolle wird dem HiSAP-Baukasten entnommen (Abschnitt 7.4, Anpassung der Simulation und Rollenkonzept). So ist im Beispiel “Zweiwege Authentifikation mittels asymmetrischer Verschlüsselung” der Name der Rolle “Intruder” entsprechend hisap.2wayAuthentifikation.Intruder. Durch diesen Floornamen kann der zur Rolle zugehörige Sub-Floor identifiziert werden.

Kommunikation zwischen AppletServer und AppletClient

Das Kommunikationsmedium zwischen dem *AppletServer,* wo die HiSAP-Simulation durchgeführt wird, und dem *AppletClient,* wo die HiSAP-Visualisierung ausgeführt wird, ist der Kommunikationskanal. Er überträgt die *ActionEvents* und die Ergebnisse der Simulation (Abschnitt 7.4). Um die Übertragung der ActionEvents zu ermöglichen, wird eine Klasse *SubActionEvent* im Package hisapapplet erstellt. *SubActionEvent* ist eine Subklasse von *ActionEvent* und für die Integration von HiSAP definiert worden. Details werden im nächsten Unterkapitel behandelt.

Beseitigen der Ausnahmen

Eine Reihe von Exceptions wurde definiert um etwaige Ausnahmefälle abfangen zu können. Als Basisklasse dient hierbei *AppletException*. Falls ein Moderator ein gestartetes HiSAP-Anwendungsbeispiel noch einmal starten möchte wird eine *AppletAlreadyRunningException* geworfen. Wenn ein beendetes Anwendungsbeispiel noch einmal beendet werden soll, tritt eine *AppletAlreadyStopedException* auf. Falls das Applet nicht existiert, erscheint eine *AppletDoesNotExistException.*

Internationalisierung der Sprachelemente

Um die Benutzungsschnittstelle an unterschiedliche Sprachen anzupassen, wurden im Package hisapapplet die Klassen *GUIResourceBundle* und *GUIResourceString* definiert. Zuerst wird durch die Methode *getDefault()* der Klasse *java.util.Locale* [JAVA] herausgefunden, welche Sprache das Betriebsystem benutzt. Mittels der Methode *init(Locale)* von *GUIResourceString* sucht *GUIResourceBundle* eine vorher erstellte Propertydatei, die den passenden Text speichert. Dadurch können mit den von Java bereitgestellten Mitteln der Inter-

nationalisierung die sprachspezifischen Elemente automatisch korrekt auf der Benutzungsschnittstelle angezeigt werden.

Schnittstelle zu HiSAP

AppletInterface ist für die Integration von HiSAP definiert worden und ist eine Schnittstelle, die den HiSAP-Anwendungsbeispielen SASCIA-Funktionalität anbietet.

7.4 Erweiterung von HiSAP

Im Entwurf haben wir festgestellt, daß, um das Rollenkonzept in HiSAP zu realisieren, die Funktionen Transformierung des *ActionEvents* und Identifikation der Rolle angeboten werden müssen. Deshalb wurde im Package *sascia.hispapplet* "*AppletInterface*" erstellt, das eine Schnittstelle zwischen SASCIA und HiSAP darstellt.

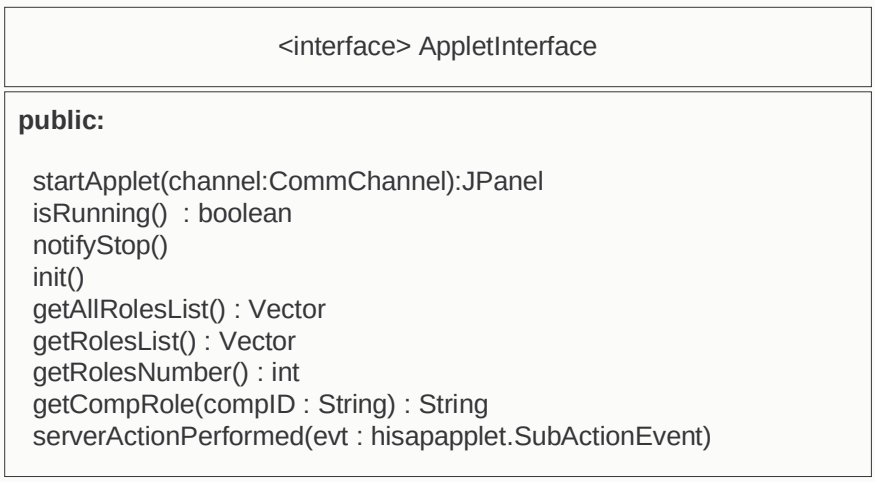


Abbildung 7.6: Interface AppletInterface

Starten eines HiSAP-Anwendungsbeispiels

Mittels *startApplet* wird die Ausführung eines Applets beim Server und Client gestartet. Als Parameter dient ein Kommunikationskanal, der die Daten, die von der Simulation gebraucht und erzeugt werden, überträgt. Um im weiteren Verlauf festzustellen, ob das HiSAP-Anwendungsbeispiel noch läuft wird die Methode *isRunning()* benutzt. Sie liefert einen booleschen Wert, der true ist, falls das Beispiel noch läuft und ansonsten false. Die Methode *notifyStop()* ist für das Beenden eines HiSAP-Anwendungsbeispiels zuständig.

Anpassung der Simulation und Rollenkonzept

Wenn ein *AppletServer* geöffnet ist, wird die HiSAP-Anwendung durch die *init()*-Methode initialisiert. Während des Initialisierungsvorgangs werden alle Simulationskomponenten, z.B. *Node, Strategy* und *Connection*, initialisiert und alle Rollen an *Nodes* durch *JNodeview* zugewiesen. Da *JNodeView* von der Klasse *JButton* erbt, kann mittels der Methode *setText* die Simulationskomponente *Node* auf der Visualisierungsebene identifiziert werden. Deshalb wurde *JNodeView* um die Methoden *setRole* und *getRole* erweitert, um die Rollen zu identifizieren. Mittels *setRole* wird der Komponente der Name der Rolle zugewiesen, *getRole* kann benutzt werden, um den Namen der Rolle zu erhalten.

Da ein konkretes HiSAP-Anwendungsbeispiel auch verschiedene Kommunikationsprotokolle integrieren kann, ergeben sich für unterschiedliche Kommunikationsalgorithmen unterschiedliche Topologiedarstellungen, die gleiche Rollennamen benutzen. Die Methode *getAllRolesList* gibt alle Rollen eines HiSAP-Anwendungsbeispiels zurück. Dabei können die Rollen auch unterschiedlichen Protokollen zugeordnet sein. Genutzt wird diese Methode von

AppletServer bzw. *AppletClient*, um den jeweiligen Sub-Floor-Server bzw. den Sub-Floor-Client zu erstellen.

Die aktuelle Rollenliste eines einsehnen Beispiels ist durch *getRolesList* erreichbar. Allerdings erscheinen hier nur die Rollen, so genannte aktuelle Rollen, die eine verfügbare Visualisierungskomponente hat.

Anpassung von Visualisierung und Rollenkonzept

Es ist sinnvoll die Visualisierungskomponenten entsprechend ihrer Funktionalität im Sinne der Rollenzugehörigkeit zu gruppieren, also alle Visualisierungskomponenten einer Rolle zuzuordnen, wenn diese Komponenten alle unter den gleichen Umständen freigegeben werden sollen. Ein einfaches Beispiel im Kommunikationsprotokoll, “Zweiwege Authentifikation mittels asymmetrischer Verschlüsselung”: Bei der Rolle Intruder gibt es drei Knöpfe in dem internen Fenster: “catch message”, “send it” und “use key”. Diese Knöpfe sind die Funktionen, die ein Intruder in diesem Kommunikationsprotokoll ausführen kann und daher gleichzeitig aktiv. Um die Zuordnung zu gewährleisten ist die Methode *getCompRole* zu implementieren. Sie liefert zu der ID einer Visualisierungskomponente den Namen der zugehörigen Rolle. Beispielsweise kann eine *HashTable* benutzt werden, um eine n:1-Beziehung zwischen den Komponenten und der Rolle zu speichern.

Anpassung der Kommunikation

Die von solche Visualisierungskomponenten ausgelösten *ActionEvents* sollen durch Kommunikationskanal übertragen. Im Entwurf haben wir festgestellt, daß um die Übertragung der Visualisierungskomponente durch den Kommunikationskanal an Server zu ermöglichen, die Visualisierungskomponente eine ID erhält (*String*). Daher wurde eine Sub-Klasse von *ActionEvent* erstellt (*Sub-*

ActionEvent, Abbildung 7.7), die das ursprüngliche *ActionEvent* und die ID der Komponente (*String*) als Parameter im Konstruktor erhält.

Durch die Methode *getCompID* von *AppletInterface* kann die zugehörige Quelle gefunden werden. Damit kann die Simulation auf der Serverseite laufen, ohne tatsächlich dort Visualisierungskomponenten zu besitzen. Die von der Simulation erzeugten neuen *ActionEvents* werden auf die gleiche Weise an den Client übertragen. Anstatt der ursprünglichen *actionPerformed*-Methode wird dort jetzt die Methode *serverActionPerformed* aufgerufen.

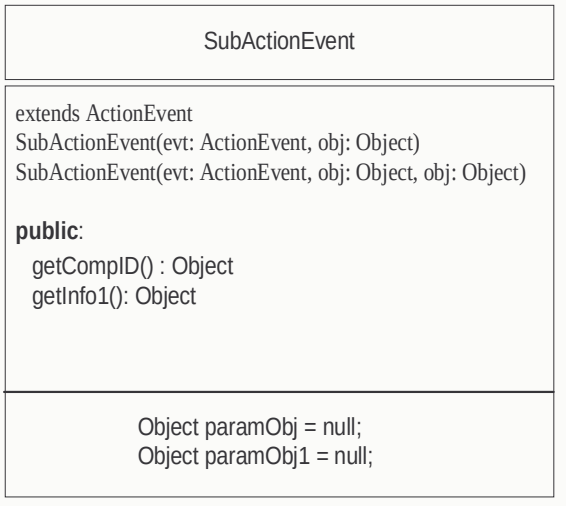


Abbildung 7.7: Klasse SubActionEvent

7.5 Bewertung

In dieser Diplomarbeit wurde eine prototypische Testversion anhand eines konkreten Kommunikationsprotokolls entwickelt. Als ein Testanwendungsbeispiel nehmen wir das Applet: “Zweiwege Authentifikation mittels asymmetrischer Verschlüsselung”, das den HiSAP-Baukasten benutzt und den Algorithmus “asymmetrische Zweiwege Authentifikation” darstellt. Wir integrieren dieses Applet in SASCIA. Der Applikationsname ist “HiSAP”. Der Appletname ist “AsymTwoWayAuthenticationTrainee”. 4 Sub-Floors wurden bei

der Initialisierung dieses Applets erstellt, nämlich User A, User B, Intruder und Authentication Server. Nach der Darstellung der Topologie des Algorithmus “with key” sind drei Rollen sichtbar: User A, User B und Intruder. Die Teilnehmer können in diesem Fall nur die drei Rollen beantragen.

Diese Testversion bietet die Basisfunktionen: Identifikation der Rollen, Floor- / Sub-Floor-Verwaltung, Transformierung des *ActionEvents*, Übertragung der transformierten *ActionEvents*, Kommunikation zwischen Simulationsserver und Visualisierungsclient.

Lediglich die Serialisierung der Nachricht (*Message*) wurde noch nicht implementiert. Die Simulation des Intruders nutzt allerdings diese Nachrichten (*Message*). Auf der Visualisierungsebene wird lokal eine gehackte Nachricht (*Message*) erzeugt. Die Nachricht (*Message*) ist in einer Baumstruktur aufgebaut, bei der jede Nachricht beliebige viele Kindernachrichten (im Englischen: *Child Messages*) haben kann. Die Informationen, die von den Simulationskomponenten gebraucht oder erzeugt werden, müssen zuerst isoliert und dann über den Kommunikationskanal übertragen werden. Aufgrund der umfangreichen Analyse, kam es zu zeitlichen Problem im Hinblick auf die Serialisierung innerhalb der vorgegebenen Zeit. Später könnte eine eigene Subklasse von *Message* erstellt werden, um die von den Simulationskomponenten gebrauchten und erzeugten Informationen zu speichern und serialisieren.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

Thema dieser Diplomarbeit war die Integration des HiSAP-Baukastens in das Projekt SASCIA. Damit sollte das gemeinsame Lernen von Kommunikationsprotokollen durch elektronisch unterstütztes Rollenspiel ermöglicht werden. Für die Integration dieser zwei Projekte bedurfte es der Einführung des Rollenbegriffs und der Anpassung der im HiSAP-Baukasten vorhandenen Synchronisationsmechanismen sowie einer Erweiterung der in SASCIA vorhandenen Floorkontrollmechanismen und -strategien.

Wie sich während der Ist-Analyse des HiSAP-Baukasten herausstellte muß, um die Synchronisationsmechanismen anzupassen, die Simulation zentral durchgeführt werden. Da der gegenwärtige HiSAP-Baukasten ereignisgesteuerte Simulation verwendet, müssen die Ereignisse durch den Kommunikationskanal zwischen Server und Client übertragen werden. Für die Integration dieses Verfahrens in das SASCIA-Projekt mußte eine abgeänderte Variante (*SubActionEvent*) entworfen werden.

In verschiedenen Kommunikationsprotokollen gibt es unterschiedliche Rollen. Um das Rollenkonzept durchzuführen, ist die Identifikation der Rollen in HiSAP notwendig. Dadurch wird das elektronisch unterstützte Rollenspiel überhaupt erst möglich.

Ohne Integration der Floor-Control konnte die Identifikation der Rollen im HiSAP-Baukasten allein das gemeinsame Lernen von Kommunikationsprotokoll noch nicht ausreichend unterstützen. Weil unter einem Kommunikationsprotokoll viele Rollen vorhanden sind, mußten wir eine Sub-Floor-Control hinzufügen. Vor der Integration der Floor-Control wurde also die existierende Floor-Control erweitert, um die Sub-Floor-Verwaltung zu ermöglichen. Das Rollenzuteilungsproblem und der Rollenwechsel sind zwei wichtige Themen im Sub-Floor-Control. Gemäß den unterschiedlichen Testszenarien wurden unterschiedliche Strategien ausgewählt.

Anschließend wurden alle benötigten Komponenten in den Anwendungsrahmen integriert. Damit ergibt sich eine benutzerfreundliche Benutzungsschnittstelle für alle Teilnehmer.

Nach dem Entwurf wurde eine prototypische Implementierung mit Hilfe eines Anwendungsbeispiels durchgeführt.

8.2 Ausblick

Diese Diplomarbeit zielt auf das kollaborative Lernen von Kommunikationsprotokollen durch elektronisch unterstütztes Rollenspiel. Wie in der Anforderungsanalyse gezeigt, konnten nicht alle Komponenten und Anforderungen, die in der Systemarchitektur und der Anforderungsanalyse erwähnt werden, auch tatsächlich umgesetzt werden. Aufgrund der umfangreichen Anforderungen, kam es zu zeitlichen Problemen im Hinblick auf die vollständige Realisierung

innerhalb der vorgegebenen Zeit. Daher beschränkte sich diese Diplomarbeit auf die Implementierung der Sub-Floors von SASCIA, Rollenidentifikation in HiSAP-Baukasten, Serialisierung der ActionEvent und die Implementierung der Benutzungsoberfläche.

Insbesondere ist noch das Serialisierungsproblem der Klasse *Message* zu lösen. Es gibt zwei Möglichkeit, um die Serialisierung der *Messages* zu realisieren. Erstens können wir die Informationen in *Message*, die von Simulationskomponenten gebraucht werden oder erzeugt wurden, isolieren und kapseln. Danach werden diese Informationen per Hand serialisiert. Aber bei hierarchischen Nachrichten (*Messages*) muß dafür gesorgt werden, daß die Kindernachrichten (child Messages) mit übertragen werden. Eine andere Möglichkeit wäre, daß die Klasse *Message* künftig als *Serializiable* implementiert wird.

Zur Zeit stehen die Sitzungsverwaltung [Somm01] und Protokollierungsdatenbank [Schm01] bereits zur Verfügung. Gleichzeitig wird an weiteren Einsatzgebieten des SASCIA-Systems, die prototypischer Realisierung von elektronischer Tafel und Leinwand in der Lehre, gearbeit [Ober02]. Die Integration des HiSAP-Bakasten mit den anderen Komponenten aus dem Projekt SASCIA (Sitzungsverwaltung, Protokollierungsdatenbank, andere Anwendungen) wird in kurzer Zeit erfolgen können.

Literaturverzeichnis

[Bonn97] *D. Bonnemann* : Gedanken zu otentialen und Grenzen des Teleteaching : Das Internet als Lehr- und Lernmedium, 1997

[Bran97] *O. Brand, M. Zitterbart :* Steuerung von Konferenz- und Kollaborations-Anwendungen : Praxis der Informationsverarbeitung und Kommunikation (PIK), 97(4), 1997

[Brod99] *Torsten Brodbeck :* Grundlegende Überarbeitung der Teile des Java Visualisierungsbaukastens für Protokolle : Studienarbeit, Universität Stuttgart, 1999

[Brod00] *Torsten Brodbeck :* Erweiterung des Animationssystems HiSAP um Synchronisationsmechanismen auf der Grudlage logischer Zeit : Diplomarbeit, Universität Stuttgart, 2000

[Burg97] *Cora Burger :* Groupware-Kooperationsuntersützung für verteilte Anwendungen : dpunkt-Verl. für digitale Technologie, 1. Auflage 1997

[Crow90] *T. Crowley, P.Miazzo, E. Baker, H. Forsdick, R. Tomlinson*: MMConf: An Infranstructure for Bilding Shared Multimedia Applications Proceedings of the ACM Conference on CSCW: Los Angeles, 1990: ACM-Press

[DoGa97] *H.-P. Dommel, J.J. Garcia-Luna-Aceves :* Floor control for multimedia con-ferencing and collaboration: Mutimedia Systems (1997)

[Eng93] *Hermann Engsesser* : Hrst. Duden Informatik, Dudenverlag, Mannheim, zwite Auflage, 1993

[FESTIVAL] *Projekt FESTIVAL* : Verteilte Systeme, Universität Stuttgart: http://Festival.informatik.uni-stuttgart.de/

[Hilt01] *V. Hilt* : Automatisierte Erzeugung von Online-Lehreinheiten aus Tele-Veranstaltungen: Jahrestagung der GI und der OCG, Informatik 2001: Beitrag in Taugungsband in Informatik 2001

[HiKuVo01] *V. Hilt, C. Schremmer, C. Kuhmüch, J. Vogel* : Erzeugung und Verwendung multimedialer Teachware im synchronen und asynchronen Teleteaching : Praktische Informatik IV, Mannheim: http://www.wirtschaftsinformatik.de/wi/archiv/2001_1inh.htm

[JAVA] *Java 2 Platform Standard Ed, v1.3.1, API Specification :* http://java.sun.com/j2se/1.3/doc/api/index.html

[JSDT] *Java Shared Data Toolkit :* http://java.sun.com/products/java-media/jsdt/index.html

[KoKöBü97] *M. Koch, D. Köhler, M. Bürger* : Workpaper Awareness Information in Wide Area Networks : Informatik XI, Technische Universität München, Germany, Mar. 1997: http://www11.informatik.tu-muenchen.de/publications/abs/koch1997c.html

[Ober01] *Peter Oberparleiter :* Vergabe von Zugangsberechtigungen (floor control) für die gemeinsame Nutzung von Smartboards in der Vorlesung : Studienarbeit, Universität Stuttgart, 2001

[Ober02] *Peter Oberparleiter :* Eine Architektur zur gemeinsamen Nutzung von Anwendungen in der Lehre mit prototypischer Realisierung von elektronischer Tafel und Leinwand : Diplomarbiet, Universität Stuttgart, 2002

[Pank98] *Uta Pankoke-Babatz* : Awareness: Spannungsfeld zwischen Beobachter und Beobachtetem : Von Groupware zu GroupAware der D-CSCW 1998

[Rein94] *W. Reinhard, J. Schweitzer, G.Völkensen, M. Weber* : CSCW Tools: Conceptes and Architectures : Communications of the ACM 27, 1994

[Saut95] *C. Sauter, O. Morger, T. Mühlherr, A. Hutchison, S. Teufel :* CSCW for Strategic Management in Swiss Enterprises : an Empirical Study. Proc. 4th European Conference on Computer-Supported Cooperative Work. Proc. of ESCW'95, H. Marmolin, Y. Sundblad, K. Schmidt (Eds.), Kluwer Academic Publishers 1995, S. 117-132

[Schm01] *Andreas Schmid :* Prototypische Erstellung einer Protokollierung für den Einsatz in Lehrveranstaltungen : Studienarbeit, Universität Stuttgart, 2001

[Somm01] *Marcus Sommer :* Prototypische Erstellung einer Sitzungsverwaltung für den Einsatz in Lehrveranstaltungen : Studienarbeit, Universität Stuttgart, 2001

[SöRe99] *M. Schöppler, T. Reicherzer* : Awareness: Basis für kollaboratives Lernen : Universität Karlsruhe, 1999, Teleseminar: “Distance Learning”

[SwaFiVa01] *G. Schwabe, C. Filk, M. Valerius* : Warun Kooperation neu erfinden? : Institut für Wirtschafts- und Verwaltungsinformation, Uni. Koblenz-Landau, 2001: http://dominosrv.uni-koblenz.de/iwi3/Literaturverwaltung.nsf

[VirtuGrad] *Projekt VirtuGrade / Evaluation :* Universität Tübingen: http://www.virtugrade.uni-tuebingen.de/ : August, 2001

[Walt98] *M.. Walter:* Kollaborative Lehrsysteme : Technische Universität München, Germany, 1998, Haupseminar “Intelligente Lehrsysteme”