Diplomarbeit

Entwurf und Implementierung einer Gruppenfunktionalität für die Lernplattform LAssi

Universität Hamburg
Department Informatik

Freitag, 10. November 2006

Autor:
Heiko Weber
Fritz-Schumacher-Allee 17
22417 Hamburg

Matrikelnr.: 520 16 52
E-Mail: 9weber@informatik.uni-hamburg.de

Erstbetreuer: Dr. Axel Schmolitzky
Zweitbetreuer: Prof. Dr. Horst Oberquelle

Erklärung

Ich, Heiko Weber, bestätige hiermit, dass ich die vorliegende Diplomarbeit allein und selbstständig angefertigt habe.

Hamburg, 10.11.2006, ...
(Heiko Weber)

Danksagung

Ich danke allen, die mich bei der Erstellung dieser Arbeit und während meines gesamten Studiums unterstützt habe. Dazu gehören meine Eltern und meine Freundin. Bei den Mitgliedern des Projektes „Reinventing Education - Werkzeuge für das Lernen“ sowie meinem Erstbetreuer, Axel Schmolitzky, bedanke ich mich für deren hilfreiche Unterstützung. Zudem danke ich Claus Andersen für die Zusammenarbeit am CommSy.

Anmerkung

Aus Gründen der einfacheren Lesbarkeit wird in der Diplomarbeit immer die männliche Form (Mitarbeiter, Lehrer etc.) genutzt. Es sind jedoch sowohl weibliche wie männliche Personen gemeint.

Inhaltsverzeichnis

1 Einführung

Für das Projekt „Reinventing Education - Werkzeuge für das Lernen“ wurden vier Anwendungsprofile entwickelt, die beschreiben, wie Benutzern der Lernumgebung *LAssi* das gemeinschaftliche Lernen ermöglicht werden soll. Die Anwendungsprofile handeln von Kommunikation, Koordination und Gemeinschaftsarbeit kleinerer geschlossener Gruppen. LAssi ist zur Zeit als Einzelplatzsoftware konstruiert und bietet keine Funktionalität in der Hinsicht an.

Diese Diplomarbeit beschäftigt sich daher mit der Frage, wie Gruppenfunktionalität in LAssi umgesetzt werden kann. Die Gruppenfunktionalitäten werden in Anwendungsprofilen beschrieben. Zu den Anwendungsprofilen wurden Lösungsansätze entwickelt, die zeigen, wie LAssi-Instanzen miteinander interagieren können. Es wurden zwei Lösungsansätze konkretisiert. Bei der Ersten Lösung greifen die einzelnen LAssi-Instanzen auf die Web Services des CommSy zu. In der zweiten Lösung interagieren die LAssi-Instanzen mit Hilfe des Eclipse Communication Framewoks direkt miteinander. Die konkretisierten Lösungen wurden in LAssi implementiert.

1.1 Kontext

Die Lernumgebung LAssi

Entwickelt wird die Lernplattform bzw. die Lernumgebung **LAssi** im Rahmen des Projektes „Reinventing Education - Werkzeuge für das Lernen“. LAssi soll den Benutzer in seiner individuellen Art zu lernen unterstützen. Das aktuelle Grundkonzept von LAssi basiert auf so genannten Materialien, die durch Lernwerkzeuge bearbeitet werden können. Die bisherigen Prototypen von LAssi sind auf Basis der *Eclipse Rich Client Platform* (RCP) realisiert. Weiter Informationen zu LAssi sind auf der Website [LAssi01] zu finden.

Das CommSy

CommSy steht für “Community System“ und ist ein webbasiertes System zur Unterstützung von vernetzter Projektarbeit [Com01]. Begonnen wurde das Projekt im Mai 1999 am Fachbereich Informatik der Universität Hamburg. In der Kooperation mit HITeC e.V. wird das CommSy als Open Source-Projekt weiterentwickelt und zur Nutzung angeboten.
Bei der vernetzten Projektarbeit handelt es sich um die Kommunikation und Koordination kleinerer geschlossener Gruppen.

Das Eclipse Communikation Framework

Mit dem *Eclipse Communikation Framework* (ECF) kann synchrone und asynchrone Kommunikation realisiert werden. Eclipse-Rich-Client-Anwendungen können Peer-to-Peer Kommunikation realisieren, indem sie das ECF benutzen. Information über das ECF sind auf der Website [ECF01] zu finden.

1.2 Problemstellung

LAssi ist bisher als Eclipse Rich Client konstruiert. Es ist als Einzelplatz-Software entwickelt, die nur einen Benutzer beim Lernen mit Materialien und Lernwerkzeugen unterstützt. Bisher ist es in LAssi nicht möglich, mit anderen Benutzern, die ebenfalls LAssi zur Lernunterstützung haben, zusammen zu arbeiten oder Informationen auszutauschen. Für eine Lernumgebung ist es jedoch unerlässlich, dass Gruppenfunktionalitäten, wie gemeinsames Kommunizieren, Austausch von Materialien und das gemeinsame Arbeiten, zur Lernunterstützung angeboten werden. In einem ersten Schritt wurden für das Projekt in vier Anwendungsprofilen wünschenswerte Gruppenfunktionalitäten beschrieben. Die Diplomarbeit soll zu den Anwendungsprofilen Lösungsvorschläge liefern und mindestens einen dieser Vorschläge implementieren.

1.3 Aufbau

Zu Beginn wird das Projekt „Reinventing Education - Werkzeuge für das Lernen“, die Ideen und Konzepte, die hinter LAssi stecken und der aktuelle Prototyp näher vorgestellt. Dann wird auf die Problemstellung eingegangen und in diesen Zusammenhang die von den Mitgliedern des Projektes entworfenen Anwendungsprofile erläutert.
Die Kooperationsplattformen BSCW und CommSy, die auch Gruppenarbeit unterstützen, werden kurz vorgestellt, um dann vergleichbare Gruppenfunktionalitäten hervorzuheben, wie sie in den Anwendungsprofilen für LAssi gefordert sind. Für eines der Anwendungsprofile werden zwei Lösungen erklärt. Die erste Lösung beruht darauf, dass LAssi-Clients die Web Services des CommSy benutzen, um die Gruppenfunktionalität aus dem Anwendungsprofil zu realisieren . Die zweite Lösung stellt dar, wie LAssi-Clients mit Hilfe eines ECF-Servers interagieren können, um diese Gruppenfunktionalitäten aus dem Anwendungsprofil umsetzen zu können. Zudem wird verdeutlicht warum für das ausgewählte Anwendungsprofil zwei Lösungen entworfen und implementiert wurden. Außerdem wird beschrieben, wie die Lösungen in LAssi benutzt werden können. Schließlich werden die eingesetzten Techniken, Frameworks und Plattformen erklärt und näher auf die Implementierung der beiden Lösungen eingegangen.

2 Das Projekt „Reinventing Education - Werkzeuge für das Lernen“

2.1 Allgemeines und Grundidee zum Projekt

LAssi ist eine Abkürzung und steht für „Lern-Assistent“ bzw. „Learner's Assistant“. Der Lern-Assistent soll künftig jedem Benutzer zur Verfügung stehen und beim individuellen Lernen unterstützen. Der Lern-Assistent wird im Rahmen des Projektes „Reinventing Education - Werkzeuge für das Lernen“ entwickelt. LAssi soll für den Lernenden vielfältige Lernwerkzeuge, diese werden auf Seite 8 erläutert, in einer einzigen Lernumgebung vereinen. Dabei soll LAssi so individuell einstellbar sein, dass der Lernende selbst entscheiden kann, welche Lernwerkzeuge er für ein erfolgreiches Lernen und eine produktive Wissensarbeit nutzen möchte. LAssi soll neben der Unterstützung zum individuellen Lernen Benutzern auch beim Zusammentragen und Aufbereiten von Wissen behilflich sein. Durch das Konzept der Lernwerkzeuge ist es vorstellbar, dass in Zukunft von einer internationalen Community weitere Lernwerkzeuge für LAssi entwickelt werden, mit denen die Wissensverarbeitung und das individuelle Lernen mit LAssi attraktiver und produktiver gestaltet werden kann. Diese Community könnte z.B. aus Entwicklern und LAssi-Benutzern mit Programmiererfahrung bestehen, die Lust und Interesse haben, LAssi weiter voranzubringen.
Die Lernwerkzeuge sollen die Benutzer bei der aktiven Wissensaneignung unterstützen, indem sie helfen, eigenes gewonnenes Wissen in LAssi systematisch anzuordnen und zu verwalten.
Weiterführende Informationen zum Projekt sind auf der Website [LAssi01] zu finden.

Einordnung von LAssi

LAssi kann dem E-Learning zugeordnet und als spezielle Lernsoftware angesehen werden. Unterscheidet man zwischen „Organizational E-Learning“ und „Personal E-Learning“, so ist LAssi zum jetzigen Entwicklungsstand dem Personal E-Learning zuzuordnen. Mit Personal E-Learning wird das individuelle Lernen bzw. das Lernen an einem persönlichen Arbeitsplatz bezeichnet, wogegen mit Organizational E-Learning gemeinsames Lernen und der Inhaltsaustausch von mehreren Benutzern gemeint ist. Es ist denkbar und wünschenswert, dass LAssi durch weitere Lernwerkzeuge, die das gemeinsame Lernen und den Inhalts- bzw. Wissensaustausch ermöglichen, erweitert wird. Dann wäre LAssi auch dem Organizational E-Learning zuzuordnen. Lernsoftware kann auch dahingehend untersucht werden, ob sie Lerninhalte vermittelt oder ob sie eine methodische Unterstützung zum Lernen bereitstellt. Hier ist LAssi beim momentanen Entwicklungsstand eindeutig zum Letzteren zu zuordnen, da LAssi inhaltsneutral ist und den Benutzer bei seiner Art zu lernen unterstützt, jedoch selbst kein Wissen vermittelt. Es ist aber nicht auszuschließen, dass zu einem späteren Zeitpunkt weitere Lernwerkzeuge entwickelt werden, deren Aufgabe sein könnte, Wissen zu vermitteln.

Anwender von LAssi

Weil in der gegenwärtigen Entwicklungsphase LAssi in Pilotklassen getestet wird, ist die Software hauptsächlich auf die Unterstützung von Schülern der Sekundarstufe ausgelegt.
LAssi soll aber dahingehend entwickelt werden, dass es jeden Lernenden sein ganzes Leben beim Lernen unterstützt. Durch die individuelle Einstellbarkeit und das Mitwachsen von LAssi mit dem Lernenden soll es dann für Schüler jeder Stufe sowie Studenten geeignet sein. So ist es z.B. denkbar, dass ab einer bestimmten Klassenstufe zusätzliche Lernwerkzeuge freigegeben werden. Da LAssi auf dem Konzept der Lernwerkzeuge beruht, ist damit zu rechnen, dass bei einer größeren Verbreitung und einer wachsenden LAssi-Community immer mehr Werkzeuge entstehen, die das Lernen auf vielfältige Art und Weise unterstützen werden. Von der Community könnten dann auch Lernwerkzeuge entwickelt werden, die auch für die Wissensarbeit (und die individuelle Informationsverarbeitung) nützlich sein werden.

Anforderungen an LAssi

Durch das Konzept und die Grundidee, die hinter LAssi steckt, wurden einige Anforderungen an LAssi durch die Gruppenleitung (siehe Seite 9) festgelegt:

- LAssi soll eine individuelle Lernumgebung sein, die dem Lernenden Lernwerkzeuge zur Verfügung stellt.
- Der Lernende soll selber die Werkzeuge wählen können, mit denen er erfolgreich arbeiten kann.
- Die Lernwerkzeuge sollen mit steigender Lernkompetenz des Lernenden an Funktionalität zu nehmen.
- LAssi soll den Lernenden bei der Wissensarbeit, wie Analysieren, Differenzieren, Strukturieren, Visualisieren, Recherchieren und Entdecken unterstützen.
- LAssi soll den Lernenden bei der Entwicklung von individuellen Lernwegen bzw. Lernstrategien helfen.
- LAssi soll Lernwerkzeuge bieten, mit denen Lernziele und Lernvorhaben präzisiert, Lernmaterialien beschafft und analysiert, Lernergebnisse produziert und präsentiert sowie Lernerfolge reflektiert und evaluiert werden können.
- LAssi soll besonders gut geeignet für Lerngemeinschaften sein, die auf selbstorganisiertes und eigenverantwortliches Lernen setzen.
- In LAssi soll das während des Lernens gewonnene Wissen aufbewahrt werden können.
- LAssi übernimmt damit die Funktion einer Schnittstelle, die das eigene erworbene Wissen in die Welt der digitalen Information überführt.
- LAssi selber aber auch alle Lernwerkzeuge sollen als Open-Source entwickelt werden, so dass eine weitgehend kostenfreie Verbreitung möglich ist.

Funktionalitäten, die LAssi als Lernumgebung anbietet

Die Lernumgebung LAssi stellt eine Plattform dar, die das Zusammenspiel zwischen LAssi-Materialien und LAssi-Lernwerkzeugen regelt. Sie sorgt dafür, dass alle Materialien und Lernwerkzeuge in einer einheitlichen grafischen Darstellung wiedergegeben und in einer einheitlichen Benutzerführung benutzt werden können. Zusätzlich bietet LAssi eine Schnittstelle für den Austausch von digitalem Wissen mit anderen Software-Anwendungen auf demselben Rechner an.

LAssi wird als *Eclipse Rich Client Plattform* (Eclipse RCP) zur Verfügung gestellt. Das Eclipse RCP basiert auf einem Plug in-Konzept. Durch dieses Konzept können neue Materialien und Lernwerkzeuge, die mit LAssi kompatibel sind, als PlugIn integriert werden.
Die Lernwerkzeuge können über die LAssi-Plattform auf Materialien zugreifen und diese benutzen bzw. bearbeiten. Lernwerkzeuge können auch über die LAssi-Plattform andere Lernwerkzeuge benutzen.

Lernwerkzeuge in LAssi

Für das Lernen im allgemeinen gibt es viele erprobte Methoden, die Unterstützung beim Lernen bieten. Aus diesen Methoden können Software-Werkzeuge werden, wenn es gelingt, diese in einer computerbasierten Lernumgebung nachzubilden.
Ein Beispiel für so eine Methode ist der sogenannte *Sortierkasten*. Schon in der Vorschule lernen Kinder, wie man z.B. Bauklötze in Sortierkästen nach Farben oder Formen sortiert. Später lernen Kinder oft kompliziertere Sachverhalte in einer Matrix oder Tabelle nach verschiedenen Aspekten anzuordnen. Mit dem Sortierkasten von LAssi steht ein Lernwerkzeug bereit, das diese Methode versucht nachzubilden. Der Sortierkasten bietet die Möglichkeit, LAssi-Karteikarten, eine Sammlung von Texten und Bildern, in eine Matrix einzuordnen. Der Titel sowie die Anzahl der Spalten und Zeilen der Matrix kann den jeweiligen Bedürfnissen angepasst werden.
Das Lernwerkzeug Sortierkasten kann in jedem fachlichen Kontext angewendet werden, um eine Strukturierung oder Ordnung zu erzeugen.

LAssi on my Stick

Ein weiterer wichtiger Aspekt ist *LAssi on my Stick*. LAssi ist so programmiert, dass es von einem USB-Stick gestartet werden kann. Der LAssi Workspace, in dem die Zustände der Lernwerkzeuge und die Materialien selbst gesichert werden, befinden sich ebenfalls auf dem USB-Stick.
Mit LAssi on my Stick ist mobiles und flexibles Lernen möglich. Der LAssiDesktop ist in der Schule, am heimischen PC oder an jedem anderen Computer einsetzbar und der Zugriff auf eigenen Daten gewährleistet.
Dadurch ist es nicht mehr nötig, dass jeder Schüler ein eigenes Notebook zur Verfügung haben muss. Vielmehr können sich mehrere Schüler ein Notebook teilen. Jeder Schüler kann dann durch seinen eigenen USB-Stick sein persönliches LAssi auf dem Notebook starten. Ebenfalls haben die Schüler die Möglichkeit, ihren USB-Stick an den PC oder das Notebook zu Hause anzuschließen,um dort mit ihrem persönlichen LAssi arbeiten zu können.

Bisher entwickelte und geplante Prototypen

Nach den Vorstellungen der Projektleitung sollen bis zum Jahre 2009 weitere Prototypen entwickelt werden, die zur Erprobung von Eigenschaften und Funktionen dienen. Diese Prototypen sollen Lösungen konzeptioneller Überlegungen aufzeigen und die Möglichkeit geben, darüber zu reflektieren, ob und inwiefern Entwicklungsideen sinnvoll sind.
Im Februar 2005 wurde der Prototyp Hypokeimenon, entwickelt von der Firma Innopract, eingesetzt. Im darauf folgenden September wurde LAssiCard und im Februar 2006 ein Prototyp mit dem Namen LAssiDesktop in die Pilotklassen gegeben. Im August wurde der Prototyp LAssiDesktop Version 2 fertig gestellt.
In dem gleichen Rhythmus sollen bis Februar 2008 noch drei weitere Prototypen mit den Namen LAssiPlattform, LAssiWorkspace, LAssiLernumgebung erscheinen. Ab August 2008 soll LAssiLernumgebung als Open-Source für eine Community von Softwareentwicklern, Pädagogen und Nutzern bereitgestellt werden.
Im Februar 2009 soll LAssiLernumgebung dann als marktfähiges Produkt allen Adressaten auf dem Bildungsmarkt zur Verfügung stehen.

Mitglieder des Projektes

Das Projekt „Reinventing Education - Werkzeuge für das Lernen“ besteht aus mehreren Mitgliedern, die zwei Gruppen zugeordnet werden können.

Die Projektleitung hat die Aufgabe, das Projekt zu steuern und so in die richtigen Bahnen zu lenken. Die Projektleitung wird von Michael Töpel gebildet. Für die Idee und das Geamtkonzept von LAssi ist zusätzlich noch Michael Vallendor zu ständig.

Seit dem Februar 2005 wird die LAssi-Software in Kooperation mit HITeC e.V. entwickelt. Das Entwicklungsteam besteht aus Studenten und Mitarbeitern des Arbeitsbereiches Softwaretechnik der Universität Hamburg.

Schuleinsatz

Die Software LAssi-Desktop wird an Schulen eingesetzt. Es sind mehrere Lehrer in das Projekt eingebunden, die LAssi-Desktop im Unterricht in ihren Pilotklassen zum Testen einsetzen. Drei Gymnasien (Alexander-von-Humboldt-Gymnasium, Wilhelm Gymnasium, Gymnasium Kaiser-Friedrich-Ufer), drei Gesamtschulen (Gesamtschule Walddörfer, Gesamtschule Niendorf, Gesamtschule Harburg), die Ganztagsschule Hegholt sowie die Wiechern-Schule setzen die Software im Unterricht ein.
Durch die Lehrer aber auch durch die Schüler der Pilotklassen erhalten die Entwickler der LAssi-Software wertvolles Feedback.

2.2 Die Lernumgebung LAssi

Wie schon im Abschnitt 2.1 erwähnt, wurde im August 2006 der Prototyp mit der Bezeichnung LAssiDesktop Version 2 fertiggestellt. Dieser Prototyp bildet die Grundlage der Diplomarbeit und wird in diesem Kapitel vorgestellt. Es wurden Gruppenfunktionalitäten entworfen und implementiert, die mit dieser Version benutzt werden können.

Die Perspektive Standardansicht

Im Kopfbereich der Oberfläche von LAssiDesktop 2 ist ein Menü, eine Toolbar und eine Auswahl von zur Verfügung stehenden Perspektiven gegeben. Durch die Auswahl einer bestehenden Perspektive wird das Layout der restlichen Oberfläche definiert. Eine Perspektive legt z.B. fest, wo auf der Oberfläche Lernwerkzeuge und Materialien angezeigt werden. Der Benutzer von LAssi kann jederzeit eigene Perspektiven anlegen.
Beim ersten Start von LAssiDesktop 2 ist die Standardansicht ausgewählt. Diese ist so vorkonfiguriert, dass auf der linken Seite der Oberfläche ein neu erzeugter bzw. der zuletzt verwendete Desktop erscheint. Die rechte Oberfläche wird bereitgestellt um Editoren zu öffnen, die das Bearbeiten von Materialien, die auf dem Desktop erzeugt wurden bzw. zur Verfügung stehen, ermöglichen.

Der Desktop

Der Desktop dient in erster Linie dazu, verschiedene Materialien in einen gemeinsamen Kontext zu stellen. Diese Materialien können auf dem Desktop erzeugt, gelöscht, angeordnet, strukturiert und verwaltet werden.

Die Karteikarte

Eines der wichtigsten Materialien in LAssiDesktop 2 ist die *Karteikarte* (siehe Abbildung 1). Die Karteikarte ist eine Sammlung von Texten, Bildern und Lesezeichen. Lesezeichen können auf andere Materialien oder externe Dokumente verweisen. Materialien können mit einen Editor bearbeitet werden.
Über das Kontextmenü des Editors bzw. über die nun sichtbaren Schaltflächen in der Toolbar können die Eigenschaften des Materials aufgerufen, neue Text- / Bildeinträge oder ein neues Lesezeichen zur Karteikarte hinzugefügt werden.

Der Sortierkasten

Als Beispiel für ein Lernwerkzeug kann der Sortierkasten (siehe Abbildung 2) genannt werden. In einem Sortierkasten können Kopien von Materialien in einer Matrix aus mehreren Zeilen und Spalten strukturiert werden. Die einzelnen Zeilen und Spalten können mit unterschiedlichen Titeln versehen werden. Auf diese Weise können die einzelnen Materialien in einem Sortierkasten zueinander in Beziehung gesetzt werden.

Vom Desktop aus oder über eine Sortierhilfe lassen sich Kopien von Materialien in den Sortierkasten ablegen. Innerhalb des Sortierkastens können die Materialien in den einzelnen Zeilen und Spalten der Matrix verschoben werden. Es lassen sich auch jederzeit Zeilen und Spalten hinzufügen bzw. entfernen.

Desktopverwaltung und Sortierhilfe für Sortierkästen

Über das Kontextmenü des Desktops bzw. über das Hauptmenü können weitere Sichten aufgerufen werden, die in der linken Hälfte der Oberfläche über den Desktop angezeigt werden. Z.B. gibt es die Sicht „Desktops“. Über diese Sicht können Desktops geöffnet, angelegt, umbenannt und gelöscht werden. Eine weitere Sicht ist die „Sortierhilfe für Sortierkästen“. Diese Sicht kann dafür genutzt werden um Materialien von verschiedenen Desktops zu sammeln, um diese dann in einen Sortierkasten ablegen zu können. Materialien können aus einem Sortierkasten entfernt werden, in dem diese in die Sortierhilfe für Setzkästen abgelegt werden. Die abgelegten Materialien können dann gelöscht oder in andere Sortierkästen eingefügt werden.

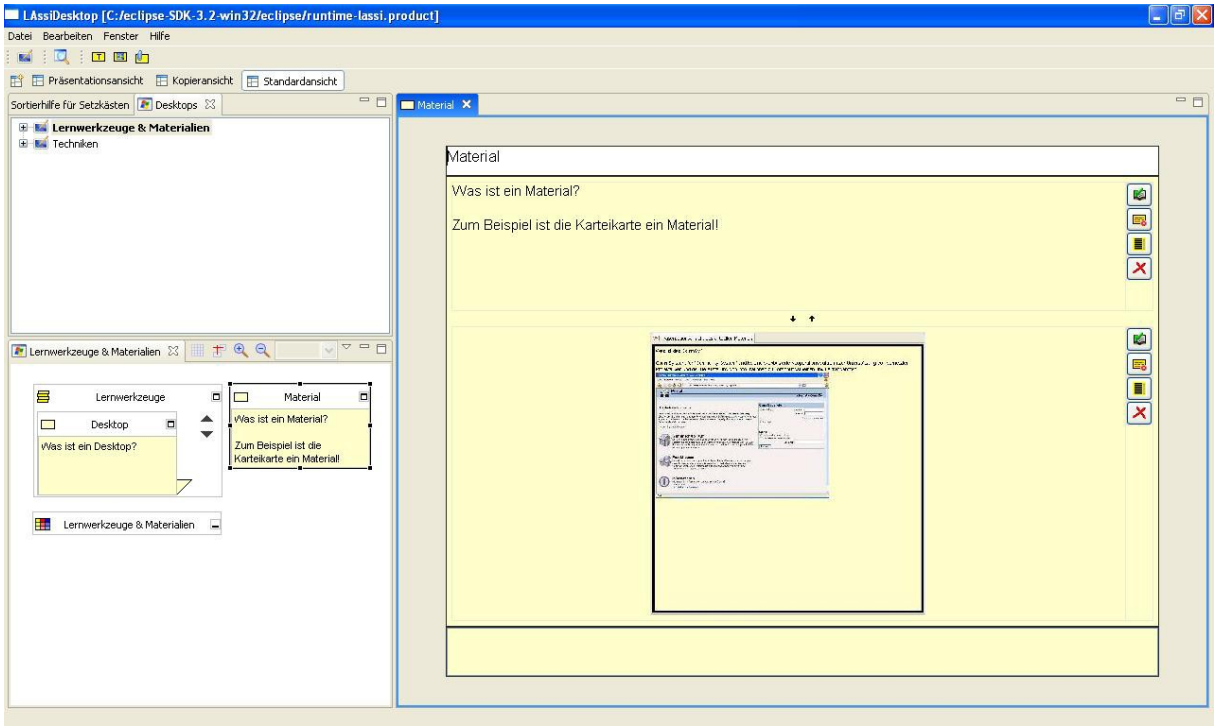

Abb. 1: Desktop, View „Desktops“ und Editor einer Karteikarte in der Standardansicht

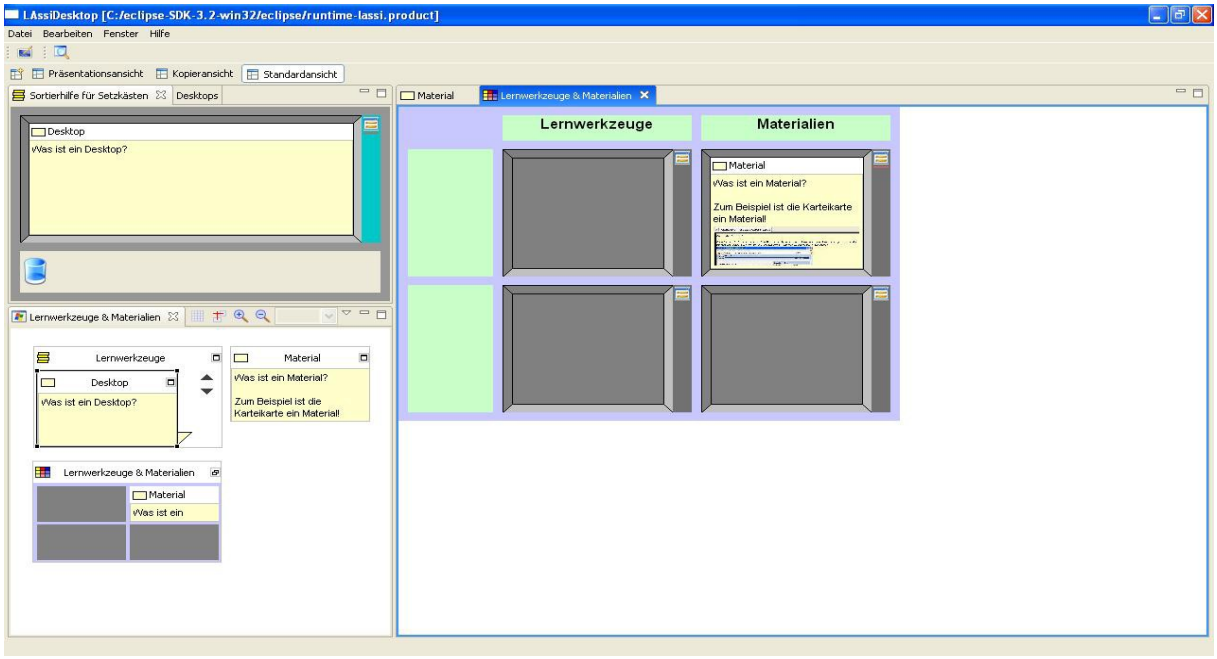

Abb. 2: Desktop, Sortierhilfe und Editor eines Sortierkastens in der Standardansicht

2.3 Vier Anwendungsprofile der Gruppenarbeit für LAssi

Es stellte sich die Frage, ob und was für welche Gruppenarbeiten in LAssi sinnvoll sind. LAssi ist bisher als Einzelplatz-Software entwickelt worden. Momentan ist es nicht möglich, mit anderen Benutzern, die ebenfalls LAssi zur Lernunterstützung zur Verfügung haben, zusammen zu arbeiten oder Informationen auszutauschen. Für den Austausch von Informationen in Form von Lernwerkzeugen und Materialien kann zur Zeit nur eine Import / Export Funktion von LAssi benutzt werden, bei der Zip-Dateien erstellt werden. Die LAssi-Benutzer müssen selber dafür sorgen, dass die Zip-Dateien auf die Rechner der anderen LAssi-Instanzen kopiert werden. Dies kann einen hohen organisatorischen Aufwand bedeuten. Für eine Lernumgebung wie LAssi ist es aber ab einem bestimmten Entwicklungszustand unerlässlich, dass Gruppenfunktionalitäten, wie gemeinsames Kommunizieren, Austausch von Materialien sowie das gemeinsame Arbeiten an einem Lernwerkzeug, zur Lernunterstützung angeboten werden.
Das Entwicklerteam des Projektes „Reinventing Education - Werkzeuge für das Lernen“ hat deshalb folgende Anwendungsprofile entworfen, die beschreiben, was für Gruppenfunktionalitäten für LAssi denkbar sind und wie diese in einer Schulklasse im Unterricht eingesetzt werden könnten:

Lehrer-Schüler-Verhältnis

Im ersten Anwendungsprofil geht es darum, dass ein Lehrer seiner Klasse eine Aufgabe zur Verfügung stellt, die mit Hilfe von LAssi betrachtet und gelöst werden soll. Eine Aufgabe kann in LAssi durch eine normale Karteikarte wiedergegeben werden. Eine Aufgabe kann sich z.B. auf Materialien oder Lernwerkzeuge, wie z.B. weitere Karteikarten oder Sortierkästen, beziehen. Die Materialien können vom Lehrer extra für die Aufgabe erzeugt und die Lernwerkzeuge für die Arbeit mit den Materialien vorbereitet werden. Diese Materialien oder Werkzeuge müssten mit der Aufgabe zusammen den einzelnen Schülern für ihr eigenes LAssi bereitgestellt werden. Die Schüler könnten die bereitgestellte Aufgabe dann abholen und bearbeiten. Die Ergebnisse können wiederum dem Lehrer bereitgestellt werden.

Direkte Zusammenarbeit

Das nächste Anwendungsprofil beschäftigt sich damit, dass mehrere Schüler gemeinsam an einem Notebook mit LAssi arbeiten und gemeinsam an diesem mit LAssi zu einer vom Lehrer gestellten Aufgabe ein Ergebnis erstellen. Ergebnisse könnten z.B. Materialien, die mit Lernwerkzeugen bearbeitet wurden, sein. Dieses Ergebnis würden dann gerne alle Schüler, die mitgearbeitet haben, an ihrem persönlichen LAssi zur Verfügung haben. Wünschenswert an dieser Stelle wäre, dass LAssi mitgeteilt wird, wer an dem Ergebnis mitgearbeitet hat. Der LAssi-Client würde dann diesen Schülern, die mitgearbeitet haben, das Ergebnis in Form von Materialien zur Verfügung stellen.

Direkte verteilte Zusammenarbeit

Das dritte Anwendungsprofil beschäftigt sich ebenfalls mit mehreren Schülern, die gemeinsam an einem Tisch sitzen und sich direkt miteinander verständigen können. Jeder dieser Schüler hat ein eigenes Notebook zur Hand, auf dem LAssi installiert ist und benutzt werden kann. Alle Schüler wollen gemeinsam mit einem Lernwerkzeug Materialien bearbeiten. Jedoch im Unterschied zum Anwendungsprofil „Direkte Zusammenarbeit“ soll jeder Schüler das Lernwerkzeug mit LAssi auf seinem eigenen Notebook sehen können. Dieses bedeutet im Detail, dass ein Schüler mit seinem LAssi über ein Lernwerkzeug mit Materialien arbeiten bzw. Materialien bearbeiten kann. Die anderen Schüler sehen ähnlich, wie bei einer Präsentation, das Lernwerkzeug, die Materialien und die durch die Bearbeitung entstehenden Veränderungen in ihrem eigenen LAssi auf ihrem eigenen Notebook. Sie können zu diesem Zeitpunkt selber aber keine Veränderungen am Lernwerkzeug oder am Material vornehmen. Es ist nur einem Schüler erlaubt, Veränderungen am Lernwerkzeug oder Material durchzuführen. Jedoch kann er das Recht, Veränderungen durchzuführen, an einen anderen Schüler bzw. an dessen LAssi weitergeben und dessen Veränderungen dann wie bei einer Präsentation beobachten. Da sie zusammen an einem Tisch sitzen, können sie direkt miteinander besprechen, welche Veränderungen an dem Lernwerkzeug bzw. dem Materialien vorgenommen werden sollen und wer als nächstes das Recht für die Bearbeitung bekommen soll.

Entfernte verteilte Zusammenarbeit

Die Schüler sitzen in diesem Anwendungsprofil räumlich voneinander getrennt an verschiedenen Orten und können nicht direkt miteinander kommunizieren. Diese Schüler wollen aber, wie bei dem Anwendungsprofil „Direkte verteilte Zusammenarbeit“, gemeinsam mit LAssi über ein Lernwerkzeug Materialien bearbeiten. Diese könnten wie im Anwendungsprofil “Direkte verteilte Zusammenarbeit“ verfahren. Da sie sich jedoch durch die räumliche Trennung nicht untereinander verständigen können, muss in LAssi ein Kommunikationskanal integriert werden. Über diesen Kanal können die Schüler darüber abstimmen, welche Veränderungen am Lernwerkzeug bzw. an den Materialien vorgenommen werden sollen. Ebenfalls können sie sich über den Kanal darüber verständigen, wer als nächstes das Recht bekommt, die Materialien mit den Lernwerkzeug zu bearbeiten.

Gruppenfunktionalitäten, die in den Anwendungsprofilen benutzt werden

In den oben genannten Anwendungsprofilen werden Gruppenfunktionalitäten angesprochen, die in diesem Abschnitt benannt und mit einem Namen versehen werden.

Vor allem in den ersten beiden Anwendungsprofilen müssen Schüler bzw. muss eine Gruppe aus Schülern ausgewählt werden, an die Materialien verschickt werden sollen. Daher wird es eine neue Gruppenfunktionalität die ***Gruppenverwaltung*** geben. In dieser Verwaltung kann der LAssi-Benutzer einzelne Personen z.B. einen Schüler oder eine Gruppe aus Schülern auswählen, um diesen Materialien anzubieten.

Eine weitere Gruppenfunktionalität aus den beiden ersten Anwendungsprofilen ist die ***Materialverwaltung***. Der Benutzer von LAssi muss über die Materialverwaltung die Möglichkeit haben, Materialien zu sammeln und diese bei Bedarf Personen oder Gruppen aus der Gruppenverwaltung zum Abholen bereitzustellen. Ebenfalls muss die Materialverwaltung von anderen LAssi-Clients zum Abholen bereitgestellte bzw. schon abgeholte Materialien anzeigen können und diese zur Weiterverarbeitung in LAssi bereitstellen.

In dem dritten und vierten Anwendungsprofil wird ähnlich einer Präsentation auf einem LAssi-Client mit Lernwerkzeugen und Materialien gearbeitet. Die Lernwerkzeuge und Materialien sowie die Arbeit mit diesen können auf anderen LAssi-Clients betrachtet werden. Diese Gruppenfunktionalität wird als ***Präsentationsmodus*** bezeichnet.

Im vierten Anwendungsprofil wird ein Kommunikationskanal benötigt, damit LAssi-Benutzer, die räumlich voneinander getrennt sind, über LAssi kommunizieren und sich untereinander austauschen können. Diese Gruppenfunktionalität wird daher als ***verteilte Kommunikation*** bezeichnen.

3 Andere verfügbare Lernplattformen

In diesem Kapitel werden die Kooperationsplattformen *CommSy* und *BSCW* kurz beschrieben und genauer auf die Gruppenfunktionalitäten der Systeme eingegangen, welche vergleichbar mit den Gruppenfunktionen der Anwendungsprofile, aus dem Kapitel 2.3, sind. In dem nachfolgenden Kapitel 4 wird dann erläutert, ob die Systeme von LAssi benutzbar sind.

3.1 Die Kooperationsplattform CommSy

Die Kooperationsplattform *CommSy* steht für "Community System" und ist eine webbasierte Kooperationsplattform zur Unterstützung von vernetzter Projektarbeit sowie der Bereitstellung von Informationen zu Forschungsthemen und Lehrangeboten. Weitere Informationen sind auf der Website [Cm01] zu finden.

Begonnen wurde das Projekt im Mai 1999 am Fachbereich Informatik der Universität Hamburg. In Kooperation mit HITeC e.V. wird das CommSy als Open Source-Projekt weiterentwickelt und zur Nutzung angeboten.

Bei der vernetzten Projektarbeit handelt es sich um die Kommunikation und Koordination kleinerer geschlossener Gruppen. Zu jeder geschlossenen Gruppe gehört ein Projektraum. Über diesen können alle Mitglieder der Gruppe ihre Arbeitsergebnisse und Materialien (z.B. Word-Dokumente, Bilder etc.) sichern sowie den anderen Mitgliedern präsentieren und zur Verfügung stellen. Des weiteren können die einzelnen Mitglieder genauer vorgestellt werden. Es besteht ebenfalls die Möglichkeit, Ankündigungen und Termine im Projektraum anzeigen zu lassen. Die Mitglieder können innerhalb der Projekträume Diskussionen führen und miteinander chaten.
Über einen Gemeinschaftsraum können Veröffentlichungen und langfristige Sicherungen von Arbeitsergebnissen, Lehr- und Lernmaterialien vorgenommen werden.

Gestaltung, Aufbau und Benutzung der Benutzeroberfläche des CommSy

Mit dem Portal als Startseite stellt das CommSy eine kleine Beschreibung seiner Community zur Verfügung. Auf dieser Seite besteht die Möglichkeit, sich über einen Login-Bereich zu authentifizieren und sich als Gast oder Mitglied der Community anzumelden. Ebenfalls ist es in diesem Bereich möglich, einen Gemeinschaftsraum oder Projektraum als Zielseite nach erfolgreicher Anmeldung anzugeben.

Mit dem Gemeinschaftsraum und dem Projektraum bietet das CommSy zwei verschiedene Raumtypen an. Der Gemeinschaftsraum ist für alle Mitglieder der CommSy-Community zugänglich. Dagegen ist der Projektraum für kleinere geschlossene Gruppen gedacht. Diese Gruppen können z.B. die Teilnehmer einer Lehrveranstaltung sein. Die Projekträume können mit den Gemeinschaftsräumen verknüpft werden. Auf diese Weise ist es möglich, mehrere Projekträume unter einem Gemeinschaftsraum in einen gemeinsamen Kontext zu stellen und projektübergreifenden Austausch zu ermöglichen.

Gemeinschaftsraum

Der Gemeinschaftsraum ist nicht nur für alle Mitglieder des CommSy zugänglich, sondern auch für Gäste. Die Mitglieder haben in diesem Raum vollen Lesezugriff, für Gäste kann der Lesezugriff für ausgewählte Materialien verboten werden. Unter Materialien sind Arbeitsergebnisse wie Dokumente, Bilder und Präsentationen zu verstehen. Gäste dürfen auch keine Einträge machen oder vorhandene Einträge verändern. Neue Einträge können in so genannten Rubriken erstellt werden.

Die Einstiegsseite zeigt die aktuellsten Einträge im Gemeinschaftsraum unabhängig von den Rubriken an. Dadurch kann sich der Benutzer mit einem Blick auf dem Laufenden halten und die neusten Einträge sofort sehen.

Ein Raum des CommSy ist dreistufig aufgebaut. Die erste Stufe bildet die eben beschriebene Einstiegsseite eines Raumes. In der zweiten Stufe wird eine Übersicht über Einträge in einer Rubrik angezeigt. Als Standard ist festgelegt, dass die zehn aktuellsten Einträge der jeweiligen Rubrik angezeigt werden. Jedoch kann auf der Seite die Anzahl der angezeigten Einträge erhöht, zu weiteren Einträgen geblättert und nach Einträgen sortiert oder gezielt gesucht werden. Die Rubrik wird erstmalig auf der Einstiegsseite über ein Menü ausgewählt. Das Menü bleibt auf jeder Seite des Raumes bestehen, so dass leicht zu anderen Rubriken gewechselt werden kann.

Um in die dritte Stufe zu gelangen, in der eine Detailansicht eines bestimmten Eintrages dargestellt wird, braucht nur auf den Titel des Eintrages in der Übersicht geklickt werden. Auf dieser Detailansicht findet der Benutzer weiterführende Informationen zum Eintrag. Abhängig von der Rubrik, in der die Einträge abgelegt worden sind, stehen auf den Detailansichten zusätzliche Funktionen zur Verfügung. Zum Beispiel wird bei einer Detailansicht für einen Eintrag der Rubrik Projektraum, auch die Möglichkeit gegeben, einen Zugang für den Projektraum zu beantragen oder direkt zum Projektraum zu wechseln.

Für den Gemeinschaftsraum gibt es folgende Rubriken:

- In der Rubrik **Ankündigungen** hat jedes Mitglied des Gemeinschaftsraumes die Möglichkeit, auf zukünftige Ereignisse aufmerksam zu machen, in dem er Ankündigungen anlegt.
- Über die Rubrik **Projektraum** kann jedes Mitglied des Gemeinschaftsraumes ein neues Projekt anlegen. Das Projekt kann näher beschrieben werden und steht unmittelbar nach der Erstellung zur Verfügung.
- In der Rubrik **Materialien** ist eine Übersicht mit sämtlichen Materialien des Gemeinschaftsraumes zu finden. Über diese Rubrik können Materialien erstellt und gelöscht werden. Materialien in einem Gemeinschaftsraum sind allen Mitgliedern des Raumes zugänglich. Zusätzlich kann bestimmt werden, ob auch Gäste auf Materialien in einem Gemeinschaftsraum zugreifen dürfen.
- In der Rubrik **Diskussionen** können alle Mitglieder eines Gemeinschaftsraumes an Diskussionen teilnehmen oder selber durch einen Initialbeitrag Diskussionen eröffnen. Durch weitere Beiträge können Mitglieder des Gemeinschaftsraumes auf den Initialbeitrag oder andere Beiträge antworten.
- In der Rubrik **Personen** werden die persönlichen Seiten der Mitglieder des Gemeinschaftsraumes aufgeführt. Jedes Mitglied kann jedoch entscheiden, ob auch Gäste seine persönliche Seite sehen dürfen. Die persönliche Seite kann z.B. Informationen zur eigenen Person und ein Foto enthalten.

- In der Rubrik **Themen** kann jedes Mitglied des Gemeinschaftsraumes neue Themen anlegen. Diese Themen bestehen aus einem Titel und einer genaueren Beschreibung. Einträge in den anderen Rubriken können einem oder mehreren Themen zugeordnet werden. Jedes Mitglied kann sich Themen zuordnen und so Interesse, Zuständigkeit oder Mitarbeit zeigen.
- In der Rubrik **Institutionen** sind Beschreibungen von Institutionen zu finden, die den Gemeinschaftsraum nutzen. Diese Beschreibungen dienen dazu, anderen CommSy-Nutzern die Institution näher zu bringen und den Nutzern vorzustellen.

Projektraum

In den meisten Fällen finden sich in Projekträumen kleinere geschlossene Gruppen in der Größe von 10 bis 30 Personen zusammen. Dieser spezieller Raum ist auch nur für diese Projektteilnehmer zugänglich.

Auch auf der Einstiegsseite eines Projektraumes ist eine Übersicht der aktuellen Einträge der einzelnen Rubriken zu sehen (siehe Abbildung 3). Die Rubriken Ankündigungen, Diskussionen, Materialien, Personen und Themen sind auch im Projektraum vorhanden. Im Vergleich zu den Rubriken im Gemeinschaftsraum können Rubriken im Projektraum zusätzliche Funktionen bezogen auf Gruppen beinhalten. Unter Gruppen versteht man im Projektraum den Zusammenschluss von mehreren Mitgliedern des Projektraumes. Der Projektraum enthält mit Gruppen, Termin und Chat drei weitere Rubriken:

- Um **Gruppen** zu verwalten, gibt es im Projektraum eine Rubrik Gruppen. Über diese Rubrik kann jeder Projektteilnehmer neue Gruppen anlegen, vorhandenen Gruppen beitreten oder sich Beschreibung und Mitglieder einer Gruppe anzeigen lassen. Gruppen sollen eine bessere Koordination der Arbeit im Projektraum ermöglichen, in dem z.B. Einträge verschiedener Rubriken Gruppen zugeordnet und E-Mails recht einfach an diese Gruppen verschickt werden können.
- In der Rubrik **Termin** können Ereignisse angekündigt werden. Diese Ereignisse können z.B. Besprechungstermine für Projektteilnehmer sein.
- Eine weitere Rubrik im Projektraum ist der **Chat**. Über diese Rubrik können die Projektteilnehmer miteinander über ein Chat synchron Nachrichten austauschen oder Diskussionen führen.

Für jeden Projektraum gibt es mindestens einen Veranstalter (meistens derjenige, der den Projektraum angelegt hat), der als Autor fungiert. Dieser kann den Aufbau des Projektraumes beeinflussen, z.B. in dem er die Reihenfolge und die Anzahl der eben genannten Rubriken bestimmt.

Nähere Informationen zur Administration und Benutzung des CommSy sind in den Handbüchern [Com2], [Com3] zu finden.

Besondere Gruppenfunktionalität im CommSy (Gruppenbildung und Materialien)

Im CommSy ist die Gruppenbildung als besondere Funktionalität zu erwähnen. Anders als bei der Plattform WebCT können Gruppen innerhalb eines Kurses nicht nur vom Autor bzw. Lehrer, sondern auch von den Teilnehmern eines Kurses erstellt werden. Über die Rubrik Gruppen kann eine neue Gruppe erstellt werden. Es können dabei ein Name, eine Beschreibung und ein Bild angegeben werden. Es kann auch noch zwischen zwei Bearbeitungsmodi ausgewählt werden. Zur Verfügung stehen „Für alle bearbeitbar“ oder „Nur von (Name des Erstellers) bearbeitbar“. Die Teilnehmer können auch in der Rubrik Personen, sich selber aber auch jeden anderen Teilnehmer zu den Gruppen zuordnen bzw. aus den Gruppen entfernen.
Im CommSy können digitale Informationen in der Rubrik Material als Datei hoch geladen werden. Beim CommSy können beim Anlegen eines Materials ein Titel, eine AutorIn, das Jahr und der Bearbeitungsmodus eingegeben werden. Zusätzlich können noch Attribute wie Kurzfassung, Schlagwörter, Materialart, Datum und Gruppen angegeben werden. Das besondere hierbei ist, dass die Materialien zugeordnet werden können. Durch diese Zuordnung kann signalisiert werden, dass die dem Material zugeordnete Gruppe für die Erstellung des Materials verantwortlich ist. Im CommSy ist es durch die Zuordnung möglich, in der Rubrik Gruppe eine Auflistung von allen Materialien, die einer bestimmten Gruppe zugeordnet sind, zu bekommen.

Die Gruppenbildung ist mit der Gruppenverwaltung aus den Anwendungsprofilen, aus Kapitel 2.3 Seite 15, vergleichbar. In der Gruppenbildung können wie bei der Gruppenverwaltung Gruppen ausgewählt und diese mit Materialien in Verbindung gebracht werden.
Die Rubrik Materialien kann mit der Materialverwaltung aus den Anwendungsprofilen verglichen werden, da in dieser Rubrik Materialien über das CommSy bereitgestellt werden, ähnlich wie es für die Materialverwaltung gefordert ist.

Abb. 3: Einstiegsseite eines Projektraumes

3.2 Der Shared Workspace BSCW 4

Durch das *BSCW* ist es den Unternehmern möglich, jede digitalisierte Information ihren Partnern dauerhaft über das Internet verfügbar zu machen. Es bietet Funktionen, wie z.B. das Bereitstellen von Dokumenten, ein Dokumentenmanagment, asynchrone und synchrone Kommunikation und einen Terminkalender für Gruppen an. Unter Dokumentenmanagement ist in diesem Fall zu verstehen, dass verschiedene Versionen, Annotations, Sperrungen, der Versand und die Archivierung von Dokumenten berücksichtigt werden. Weitere Informationen sind auf der Website [BSCW01] zu finden.

Gestaltung, Aufbau und Benutzung der Benutzeroberfläche von BSCW

Das Konzept, dass hinter BSCW steht, basiert auf gemeinsamen Arbeitsbereichen für die Kooperation von Geschäftspartnern. Von einem Kooperationspartner kann im BSCW ein gemeinsamer Arbeitsbereich angelegt werden. Er kann dann seine Kooperationspartner, die registrierte Mitglieder des BSCW sein müssen, dazu einladen, dem Arbeitsbereich beizutreten. In diesem Arbeitsbereich können die Kooperationspartner, die dann als eine Arbeitsgruppe angesehen werden , die verschiedensten digitalisierten Informationen in Form von Dokumenten, Tabellen, Notizen und Grafiken ablegen, um eine gemeinsame Aufgabe zu lösen oder zu erfüllen. Ein registrierter BSCW-Benutzer kann dabei Mitglied von mehreren Arbeitsbereichen sein. Alle BSCW-Nutzer, die Mitglieder von Arbeitsbereichen sind, können über einen WWW-Browser auf die im BSCW verwalteten digitalisierten Informationen zugreifen, diese bearbeiten und auf ihren lokalen Rechnern übertragen. Ebenfalls können sich die Mitglieder einer Arbeitsgruppe eines Arbeitsbereiches über die Aktivitäten der anderen Mitglieder der Arbeitsgruppe informieren.

Nachdem Login werden alle für den angemeldeten Benutzer zugänglichen Arbeitsbereiche angezeigt. Nach Auswahl eines von diesen, gelangt der Benutzer auf die Ordnerübersicht (siehe Abbildung 5) des Arbeitsbereiches. Der Aufbau einer BSCW-Seite ist so gestaltet, dass eine Übersicht des Inhaltes eines Arbeitsbereiches bzw. eines Ordners im Mittelbereich angezeigt wird. Im darüber liegenden Kopfteil befinden sich Popup-Menüs und mit Icons versehene Schaltflächen. In einer Ordnerübersicht können eine Vielzahl von Objekten mit unterschiedlichen Objekttypen abgelegt werden. Der Objekttyp wird dabei durch verschiedene Icons dargestellt. Die wichtigsten Objekttypen sind:

- Unter **Dokumenten** versteht man in BSCW von einem lokalen System hochgeladene Dateien mit Texten, Grafiken, Tabellenkalkulationen oder Bildern. Diese Dokumente werden unter anwendungsspezifischen Dateiformaten in einem Arbeitsbereich oder Ordner abgespeichert. Außerdem gibt es für Dokumente eine Versionskontrolle, die sicher stellt, dass ein Dokument, von dem mehrere Bearbeitungsstufen oder Fassungen existieren, nicht versehentlich überschrieben werden kann.
- Das **URL-Objekt** bietet die Möglichkeit, auf Objekte außerhalb des aktuellen Ordners durch eine Verknüpfung zu zugreifen. Diese können nicht nur BSCW-Objekte in einem anderen Ordner auf dem selben BSCW-Server sein, sondern auch Objekte, die sich im Web befinden, wie z.B. eine Datei auf einem FTP-Server.

- Mit einem **Ordner** können BSCW-Objekte organisiert werden. Ein Ordner kann sämtliche BSCW-Objekte enthalten, wie z.B. andere Ordner, Dokumente, Diskussionen oder einen Kalender.
- Mit dem **Kalender** können die Benutzer ihre Termine organisieren. Neben einem persönlichen Kalender können Gruppenkalender in beliebigen Arbeitsbereichen angelegt werden.
- Eine **Diskussion** in BSCW ist ein Ordner, der nur Notizen enthält. Meistens dient die erste Notiz in diesem Ordner zur Bestimmung des Themas oder der Problemstellung. Auf diese Notiz und auf alle weiteren kann dann mit ergänzenden Notizen geantwortet werden. Dadurch entsteht ein Diskussionsstrang ähnlich wie bei Foren.
- Die **Laufmappe** ist ebenfalls ein Ordner und kann Dokumente und Notizen enthalten. Darüber hinaus ist der Laufmappe eine Liste von Aufgaben zugeordnet, die mit dem Inhalt der Mappe gelöst oder durchgeführt werden sollen. Die Aufgabenliste kann auch eine Reihenfolge festlegen, die bestimmt, zu welchem Benutzer die Laufmappe zur Bearbeitung der nächsten Aufgabe weitergeleitet werden soll.
- Durch einen **Auftrag** wird von einem Auftraggeber spezifiziert, was dieser von einem oder mehreren Benutzern erledigt haben möchte. Ein Auftrag kann dabei in Unteraufträge aufgeteilt sein und für die Aufträge relevante Materialien (Objekte) beinhalten. Jeder BSCW-Benutzer hat eine persönliche Auftragsliste, die alle Aufträge enthält, die er als Auftragnehmer erledigen soll.

Direkt über den Einträgen der Objekte befinden sich verschiedene Schaltflächen für Aktionen, die auf die Objekte angewendet werden können.

Im linken Teil des Kopfbereiches sind Popup-Menüs und Schaltflächen mit Icons platziert, die sich auf den aktuellen Ordner beziehen und das Verwalten der Objekte ermöglichen.

Auf der rechten Seite des Kopfbereiches sind eine Reihe von weiteren Schaltflächen mit Icons aufgeführt, die in jedem Ordner bzw. Arbeitsbereich sichtbar sind und z.B. Bereiche oder spezielle Ordner wie den persönlichen Arbeitsbereich, die Zwischenablage, den Papierkorb, das Adressbuch, die Kalender sowie einen Aktenkoffer schnell zugänglich machen. In der Abbildung 4 ist der Kopfbereich dargestellt.

Auf der Homepage stehen die Handbücher [BSCW02], [BSCW03] für die Administration und Benutzung des BSCW zum Download bereit.

Besondere Gruppenfunktionalität im BSCW (Arbeitsbereiche und Dokumente)

Weitere interessante Konzepte sind die Arbeitsbereichen und Dokumente in BSCW. In BSCW gibt es keine Kurse, wie bei den anderen Lernplattformen. Jedes BSCW-Mitglied hat seinen eigenen Arbeitsbereich, zu dem nur er Zugang hat. In diesem Arbeitsbereich kann er Ordner erstellen in denen Dokumente, Diskussionen, Ordner, Kontaktlisten usw. angelegt werden können. Dieser Ordner ist zunächst nur für das BSCW-Mitglied zugänglich, das den Ordner angelegt hat. Dieses BSCW-Mitglied hat nun die Möglichkeit, andere BSCW-Mitglieder dazu einzuladen, seinen Ordner mit ihm zu teilen. Die eingeladenen BSCW-Mitglieder können dann diesen Ordner in ihrem eigenen Arbeitsbereich sehen und damit arbeiten als wäre es ihr eigener Ordner. Den einzelnen eingeladenen BSCW-Mitgliedern können Rollen zugeordnet werden. So wird es möglich, dass einige eingeladene BSCW-Mitglieder nur Leseberechtigung haben, wogegen andere Mitglieder z.B. neue Dokumente und Diskussionen in dem Ordner erstellen und bearbeiten können. Auf diese Art und Weise werden in BSCW Arbeitsgemeinschaften gegründet. Dieses unterscheidet sich doch

wesentlich von den Verfahren in anderen Plattformen, wo ein Autor bzw. Lehrer einen Kurs erstellt und Teilnehmer um Zugang zu diesem bitten.
Im BSCW können digitale Informationen in Form einer Datei, die sich auf einem lokalen Rechner befindet, als Dokument angelegt werden. Dabei können optional ein Dokumenten-Name, eine Beschreibung, eine Bewertung, ein Mime-Typ und eine Kodierung angegeben werden. Zusätzlich stehen noch eine Reihe andere Attribute, die sich auf das Digital Copyright beziehen, wie z.B. Titel, Verfasser, Verleger, Kennung, Quelle, Sprache, Rechte usw. zur Verfügung.

Über die Arbeitsbereiche des BSCW können sich, wie es für die Gruppenverwaltung der Anwendungsprofile, in Kapitel 2.3 Seite 15, gefordert wird, mehrere Personen bzw. BSCW-Mitglieder zusammenfinden. In diese Arbeitsbereiche können Dokumente angelegt werden. Auf diese Dokumente können alle Mitglieder des Arbeitsbereiches zugreifen. Ein ähnliches Verhalten soll auch die Materialverwaltung aus den Anwendungsprofilen realisieren.

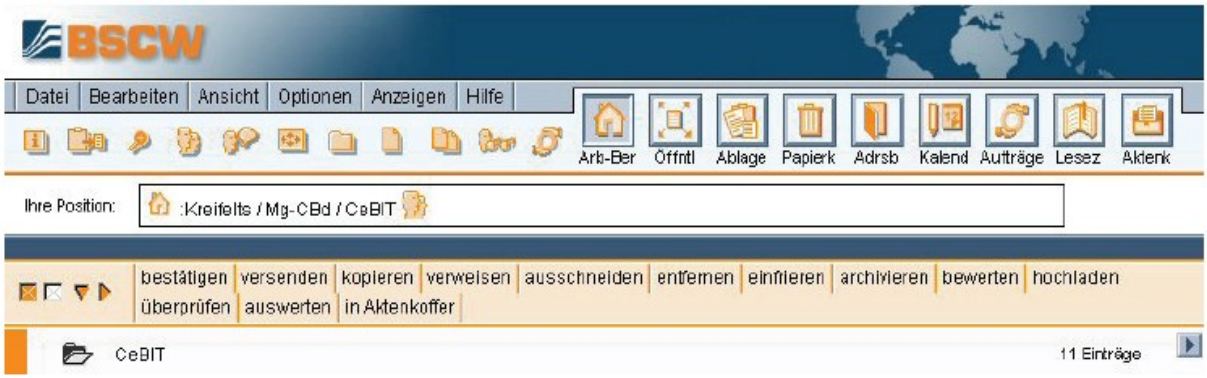

Abb. 4: Kopfbereich des Shared Workspace BSCW 4

Abb. 5: Ordnerübersicht des Shared Workspace BSCW 4

3.3 Gegenüberstellung der drei Lernplattformen

Die folgende Tabelle macht deutlich, welche Gruppenfunktionalitäten aus den Anwendungsprofilen (Kapitel 2.3) die beiden Systeme CommSy und BSCW beinhalten und welche nicht.

	CommSy	BSCW
Gruppenverwaltung		
Von wem können Gruppen angelegt werden ?	Von jedem Teilnehmer können Gruppen angelegt werden.	Jeder Benutzer kann in seinem persönlichen Arbeitsbereich Ordner anlegen. Andere Benutzer können eingeladen werden den Ordner mit ihm zu teilen.
Wer kann Teilnehmer einer Gruppe zuordnen ?	Jeder Teilnehmer kann andere Teilnehmer einer Gruppe zuordnen.	
Wer kann Teilnehmer aus einer Gruppe entfernen ?	Nur der Teilnehmer selber kann sich aus einer Gruppe entfernen.	Die Einladung kann jeder Zeit entzogen werden.
Materialverwaltung		
Können Materialien Gruppen oder Personen zugeordnet werden ?	Ja, das Material kann einer Person sowie Gruppen zugeordnet werden.	Materialien werden in einem persönlichen Ordner abgelegt. Das Material kann von mehreren Benutzern verwendet werden, wenn der Ordner geteilt wird.
Was für Attribute können einem Material übergeben werden ?	Titel, AutorIn, Jahr, Bearbeitungsmodus, Kurzfassung, Schlagwörter, Materialart, Datum und Gruppen	Name, Beschreibung, Bewertung, Mime-Typ, Kodierung, Titel, Verfasser, Verleger, Kennung, Quelle, Sprache, Rechte
Präsentationsmodus		
Enthalten die Lernplattformen etwas Vergleichbares zum Präsentationsmodus ?	Nein, das CommSy enthält keine Funktionalität, die sich mit dem Präsentationsmodus vergleichen lässt.	Nein, das BSCW enthält keine Funktionalität die sich mit dem Präsentationmodus vergleichen lässt.
Verteilte Kommunikation		
Enthalten die Kooperationsplattformen etwas Vergleichbares zur verteilten Kommunikation ?	Ja, die CommSy-Teilnehmer können über ein Chat miteinander kommunizieren.	Ja, die BSCW-Mitglieder können innerhalb eines geteilten Ordners über ein Chat miteinander kommunizieren.
Schnittstelle für Rich-Client-Anwendungen		
Enthalten die Lernplattformen eine Schnittstelle für Rich-Client-Anwendungen ?	Ja, mit dem Projekt JCommSy werden Web Service für Rich-Client-Anwendungen von CommSy angeboten.	Ja, es wird eine Schnittstelle, die auf einem RPC-Protokoll basiert, angeboten.

4 Lösungsansätze für die vier Anwendungsprofile

Während der Konzeptionsphase haben sich zwei verschiedene Lösungen für das Anwendungsprofil „Lehrer Schüler Verhältnis“ ergeben und wurden implementiert. Die erste Lösung beruht auf einen *Client-Server-System*, indem die LAssi-Clients auf die Web Services des CommSy zugreifen, um Materialien auszutauschen.
Die zweite Lösung beruht auf einen *Peer-to-Peer-System*, indem LAssi-Clients direkt miteinander interagieren um Materialien auszutauschen
Die Vor- und Nachteile des Client-Server- und des Peer-toPeer-Systems werden in den nachfolgenden Abschnitten erläutert.
Zudem wird für die erste Lösung deutlich gemacht, warum die Entscheidung auf das CommSy als Server im Client-Server-System gefallen ist und nichtauf BSCW. In dem letzten Abschnitt dieses Kapitels wird dann deutlich gemacht, das beide Lösungen sinnvoll sind.

4.1 Realisierung des Client-Server-Systems mit Hilfe von Web Services

Die Gruppenverwaltung und Materialverwaltung kann durch ein *Client-Server-System* realisiert werden, indem man auf die Kooperationsplattformen BSCW oder CommSy zurück greift. Das BSCW und das CommSy bieten beide, wie in der Tabelle (Seite 25) aufgeführt, Web Services als Schnittstelle für Rich-Client-Anwendungen an. Über diese Web Services werden im wesentlichen die gleichen Funktionen zur Verfügung gestellt wie von den Web-Interfaces der Lernplattformen. Man könnte z.B. den CommSy- oder den BSCW-Server als zentralen Server nehmen. Dieser Server würde dann für Rich-Client-Anwendungen, wie z.B. LAssiDesktop 2, über die Web Services Dienste anbieten. Die Rich-Client-Anwendung, im speziellen LAssiDesktop 2, kann dann über ein Netzwerk oder eine Internetverbindung die Dienste des Servers in Anspruch nehmen.
Die Abbildung 6 veranschaulicht das Zusammenspiel zwischen Server und LAssiDesktop 2.
Die folgenden Zitate sind verschiedene Definitionen von Web Services, die zum allgemeinen Verständnis beitragen sollen:

„Derartige Web-Dienste oder Web Services sind eine über das Internet zugängliche Schnittstelle zu Anwendungsfunktionen, die mit Hilfe von Standardtechniken des Web realisiert wird. Dabei findet die Kommunikation zwischen einer Anwendung (Client) und dem Web Service auf dem Protokoll HTTP statt.“ [Ben01, Seite 267]

„Web Services sind verteilte Softwarekomponenten, die über das Internet in andere Applikationen schnell und kostengünstig integriert werden können. Das Versprechen, zu geringen Kosten rasch die Integration umzusetzen, wird durch den Einsatz von etablierten Standardtechnologien wie Extensible Markup Language (XML) und Hypertext Transfer Protocol (HTTP) gestärkt.“ [SFT01, Seite 100]

Für weiterführende Informationen zu Web Services siehe [SFT01, Kapitel 2], [Ben01, Kapitel 3.4] und [Tay01, Kapitel 3.2].

In BSCW wird als Basis Protokoll für die Web Services das XML-RPC benutzt. XML-RPC ist ein einfaches und flexibles RPC-Protokoll, das XML benutzt, um die Datenpakete zu codieren. Die Entwickler der Web Services haben auch dieses Protokoll gewählt, um ohne großen Aufwand später die Möglichkeit zu haben auf SOAP (Simple Object Access Protocol) [SOAP01] umsteigen zu können.

Das CommSy ist ein System, dass mit PHP geschrieben ist und bietet normalerweise keine Web Services für Rich-Client-Anwendungen an. Jedoch wurde ein Projekt mit dem Namen JCommSy gegründet, das als Ziel hat, das bisherige CommSy in Java umzusetzen. Die Migration von PHP auf Java ist zwar nicht völlig vollzogen, es sind jedoch schon einige Rubriken in Java implementiert worden. Diese können auch in ein PHP-CommSy integriert und damit auch benutzt werden. Für diese Rubriken bietet das JCommSy auch Web Services für Rich-Client-Anwendungen an. Als Basis-Protokoll für die Web Services wurde Apache Axis [Axis01] benutzt. Apache Axis wird in einem Open-Source-Projekt entwickelt und verwendet den Sax-XML-Parser. Es ist im wesentlichen ein SOAP-Prozessor und kann daher SOAP-Nachrichten verarbeiten. Apache Axis 1.4 unterstützt die SOAP Spezifikation in der Version 1.1.

Wie die Tabelle, auf Seite 25, zeigt bieten der BSCW und der CommSy-Server an ihrer Schnittstelle Dienste an, um die Gruppenverwaltung sowie die Materialverwaltung aus den ersten beiden Anwendungsprofilen realisieren zu können. Der Lehrer und die Schüler könnten sich mit LAssiDesktop 2 über die Schnittstelle am Server anmelden und dort Materialien ablegen bzw. herunterladen. Schwierig wird es jedoch, das dritte und das vierte Anwendungsprofil, in dem es um einen Präsentationsmodus geht, mit den beiden Servern umzusetzen. Zur Erinnerung, es geht in diesen Anwendungsprofilen darum, dass ein Schüler auf seinem LAssi-Client arbeitet, und dass andere Schüler auf ihren LAssi-Clients diese Arbeit als Präsentation sehen. Wie die Tabelle 1 verdeutlicht besitzen der CommSy- und der BSCW-Server keine Gruppenfunktionalität, die sich mit einem Präsentationsmodus vergleichen läßt.
Um dieses zu ermöglichen müsste der LAssi-Client, auf dem gerade gearbeitet wird, jede Veränderung, z.B. an einem Desktop oder Material, allen anderen Clients, die an der Präsentation teilnehmen, mitteilen.
Bei dem BSCW-Server sowie bei dem CommSy-Server ist aber keine direkte Kommunikation zwischen den einzelnen LAssi-Clients möglich. Der LAssi-Client, auf dem gearbeitet wird, müsste jede Veränderung, die für die Präsentation wichtig ist, dem Server mitteilen. Die LAssi-Clients, die an der Präsentation teilnehmen, müssten dann ständig beim Server nachfragen, ob eine Veränderung für die jeweilige Präsentation vorliegt. Dieses Verfahren könnte für den Server etliche unnötige Anfragen zur folge haben, die diesen stark belasten könnten. Ebenfalls ist nicht sichergestellt, dass jeder Client alle Veränderungen rechtzeitig für eine Echtzeit-Präsentation erhält. Daher ist für das dritte und vierte Anwendungsprofil das Peer-to-Peer-Prinzip eher eine Lösung als ein Client-Server-System mit einem BSCW- oder CommSy-Server.

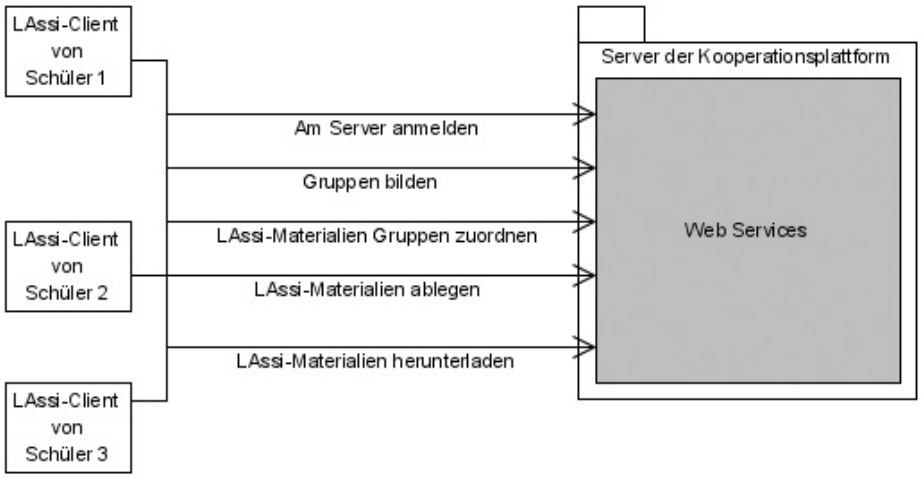

Abb. 6: Zusammenspiel zwischen Lernplattform-Server und LAssi-Client

Entscheidung für das CommSy

Die Entscheidung fiel auf das CommSy bzw. das JCommSy, da es gegenüber dem BSCW als Open-Source-Projekt angeboten wird und so kostenfrei einsetzbar ist. Das BSCW ist dagegen nur kostenpflichtig zu bekommen. Im Prinzip bietet das BSCW sowie das JCommSy an ihren Schnittstellen Dienste an, die der zentrale Server für die Realisierung des Anwendungsprofiles „Lehrer-Schüler-Verhältnis“ benötigt. Des weiteren wird das JCommSy sowie das CommSy an der Universität Hamburg entwickelt. Mir ist dadurch die Möglichkeit gegeben, Wünsche betreffend der Schnittstelle an die Entwickler heran zutragen und diese gegebenenfalls in Zusammenarbeit schnell in das JCommSy zu integrieren.

Ein weiterer Aspekt ist, dass sich die Metapher des Klassenraums besser auf das CommSy anwenden lässt. Die CommSy-Räume können mit Klassenräumen verglichen werden. Die Teilnehmer der CommSy-Räume würden dann die Schüler sein und der Autor des CommSy-Raumes wäre der Lehrer. Schüler könnten sich im CommSy selbständig zu Gruppen zusammen finden. Aber auch der Lehrer hat die Möglichkeit, den Schülern Gruppen zuzuordnen. Ebenfalls können Lehrer und Schüler Materialien in den CommSy-Räumen ablegen.
Die Metapher des Klassenraum ist nicht so einfach auf das BSCW zu übertragen. Im BSCW lässt sich auf den ersten Blick kein Vergleich zu einem Klassenraum herstellen. Es gibt nur die persönlichen Arbeitsbereiche der BSCW-Mitglieder. Der Lehrer als BSCW-Mitglied könnte einen Ordner in seinem persönlichen Arbeitsbereich erstellen und alle Schüler, die ebenfalls BSCW-Mitglied sein müssen, dazu einladen, den Ordner mit ihm zu teilen. Dieser Ordner würde auch als Ordner in den persönlichen Arbeitsbereichen der Schüler zu finden sein. In diesem Ordner können keine Gruppen angelegt werden. Es ist nur möglich, z.B. einen Unterordner zu erstellen, der nur für eine Teilmenge der Schüler zugänglich ist. Es können daher in BSCW nur Ressourcen geteilt werden, aber keine Gruppen angelegt werden. Die Struktur des BSCW lässt sich wesentlich schlechter auf die Metapher des Klassenraums abbilden als der CommSy-Raum. Daher ist anzunehmen, dass der Umgang mit dem CommSy für Schüler einfacher und intuitiver ist.

4.2 Realisierung des Peer-to-Peer-Systems mit Hilfe eines ECF-Servers

Wie schon in der Einleitung zum Kapitel 4 erwähnt, gibt es eine zweite Lösung, die in diesem Abschnitt erläutert wird. Für die Eclipse-Plattform gibt es das *Eclipse Communication Framework* (ECF). Mit diesem Framework kann ein anderes Konzept verfolgt werden. Mit dem Eclipse Communication Framework kann ein ECF-Server konstruiert werden, der als Vermittlungsserver zwischen den LAssi-Clients dient. Die einzelnen LAssi-Clients, die z.B. an einer Präsentation teilnehmen wollen, müssten sich dem ECF-Server bekannt geben. Der Client, auf dem gearbeitet wird, müsste eine Nachricht mit der Veränderung an den ECF-Server schicken, dieser würde dann umgehend dafür sorgen, dass die Nachricht an alle anderen LAssi-Clients weiter gegeben wird. So werden unnütze Anfragen an den Server vermieden und es ist sichergestellt, dass alle LAssi-Clients umgehend die Veränderungen erhalten. Ein System mit einem ECF-Server kann mit einem *Peer-to-Peer-System* verglichen werden.

Es ist auch über den ECF-Server möglich, eine ad hoc Verbindung zwischen LAssi-Clients aufzubauen. Da keine Datenbank und auch kein zusätzlicher HTTP-Server, wie beim CommSy oder BSCW, benötigt wird, ist es denkbar, dass jeder LAssi-Client einen ECF-Server beinhaltet und jederzeit starten kann. Im Gegensatz zum ECF-Server ist das Einrichten eines BSCW-Servers bzw. eines CommSy-Servers wesentlich komplizierter und zeitaufwendiger. Andere LAssi-Clients könnten sich dem ECF-Server bekannt machen. Ab diesem Zeitpunkt würde es dann möglich, dass Nachrichten, die auch Materialien enthalten können, zwischen den einzelnen LAssi-Clients ausgetauscht werden. Damit sich ein LAssi-Client dem ECF-Server bekannt geben kann, muss die IP-Adresse des Rechners auf dem der Server läuft, veröffentlicht werden. Das Veröffentlichen von anderen IP-Adressen ist nicht nötig.
Die Abbildung 7 veranschaulicht das Zusammenspiel zwischen LAssi-Clients und einem ECF-Server.

Der Vorteil, den der BSCW-Server und der CommSy-Server gegenüber dem ECF-Server haben, ist dass Materialien auf einem zentralen Server abgelegt werden können. Die abgelegten Materialien können von jedem LAssi-Client heruntergeladen werden. Dafür muss kein anderer LAssi-Client mit dem Server verbunden sein. Es ist ein asynchroner Austausch von Materialien möglich. Beim ECF-Server müssen beide LAssi-Clients mit diesem verbunden sein, wenn diese Materialien austauschen wollen. Es ist daher nur ein synchroner Austausch von Materialien möglich. Ein entscheidender Vorteil für das System mit dem ECF-Server ist, dass auf Ausfälle des Servers besser reagiert werden kann. Da jeder LAssi-Client als Server fungieren kann, kann ein anderer Client die Rolle des Servers übernehmen, falls der bisherige als Server fungierende Client ausfällt. Zudem kann ein System aus mehreren LAssi-Clients, von denen einer die Rolle des Servers übernimmt, in wenigen Minuten eingerichtet werden. Das Aufsetzen und Einrichten eines CommSy- bzw. BSCW-Servers benötigt dagegen mindestens eine Stunde (bei guten Kenntnissen). Falls ein System benutzt werden soll, dass ad hoc also in wenigen Minuten einzurichten sein soll, ist ein System mit einem ECF-Server zu empfehlen.

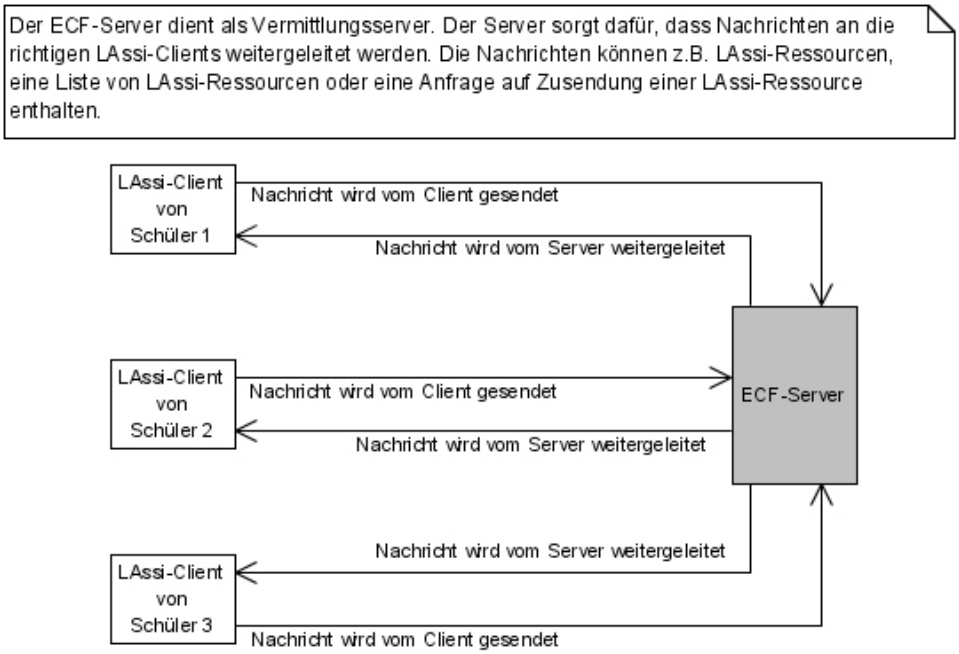

Abb. 7: Zusammenspiel zwischen ECF-Server und LAssi-Clients

4.3 Vor- und Nachteile des Client-Server- und des Peer-to-Peer-Systems

Wenn die Gruppenverwaltung und Materialverwaltung, aus Kapitel 2.3, realisiert werden, indem die LAssi-Clients Materialien auf einem Server abgelegen, handelt es sich um ein Client-Server-System. In einem solchen System werden von einem Server Dienste angeboten, die von den Clients in Anspruch genommen werden können. Von dem Server können Dienste, wie z.B. das Speichern von Materialien und das Abholen angeboten werden. Die Clients würden die Dienste in Anspruch nehmen und die Materialien für andere Clients auf dem Server ablegen bzw. speichern. Die anderen Clients könnten die Materialien dann vom Server abholen. Das folgenden Zitat ist eine Definition des Client-Server-Systems, die zum allgemeinen Verständnis beitragen soll:

„Ein Client-Server-System, bezeichnet mit C+S, besteht aus zwei logischen Teilen:
- einem oder mehreren Clients, der die Services oder Daten des Servers in Anspruch nimmt und somit anfordert.
- einem Server, der Services oder Daten zur Verfügung stellt.“ [Ben01, Seite 29]

Für weiterführende Informationen zum Client-Server-System siehe [Ben01, Kapitel 2].

Für das System, bei denen die LAssi-Clients direkt Materialien untereinander austauschen, müssen die LAssi-Clients untereinander eine Peer-to-Peer bzw. Rechner-Rechner Verbindung aufbauen. Jeder Peer (in diesem Fall ein LAssi-Client) könnte dann über einen Dienst eigene Materialien zum Abholen bereitstellen. Zugleich ist es jedem Peer aber auch möglich, die Dienste von anderen Clients in Anspruch zu nehmen, um von diesen bereitgestellte Materialien abzuholen.

„Mit den Begriff Peer-to-Peer ist die Vorstellung verbunden, dass in einem Verbund Gleichberechtigter („Peers“), die sich wechselseitig Ressourcen wie Informationen, CPU-Laufzeiten, Speicher und Bandbreite zugänglich machen, kollaborative Prozesse unter Verzicht auf Bandbreite auf zentrale Koordinationsinstanzen durchgeführt werden.“ [SFT01, Seite 3]

Mehrere Peers würden dann ein P2P-Netz aufstellen. Bei einem P2P-Netz handelt es sich um gleichberechtigte Computer, die Daten speichern, senden und empfangen können. In einem P2P-Netz ist kein zentraler Datenbestand vorhanden. Jeder Peer (Computer) stellt vorhandene Daten in Form von Diensten bereit. Kein Peer hat einen Überblick über alle angebotenen Daten. Es gibt keine zentrale Instanz, die Interaktionen im Netz steuert oder koordiniert. Die einzelnen Peers können, wenn sie direkt vernetzt sind, in Echtzeit miteinander interagieren. Es ist in einem P2P-Netz nicht sichergestellt, dass ein Peer dem Netz ständig zur Verfügung steht, vielmehr muss das Netz tolerieren, dass die Peers nicht immer online sind. In einem P2P-Netz sind die Rollen aus dem Client-Server-Prinzip aufgehoben, denn jeder Peer kann zugleich die Funktionen des Clients sowie des Servers übernehmen.
Für weiterführende Informationen zu P2P siehe [SFT01] und [Tay01].

Es stellte sich die Frage, welches Prinzip für die Interaktion der LAssi-Clients, wie sie speziell in den ersten beiden Anwendungsprofilen beschrieben ist, angewendet werden sollte.

Für eine Lösung mit dem Client-Server-Prinzip spricht, dass die zum Abholen bereitgestellten Materialien auch von anderen Clients noch abgeholt werden können, wenn der Client, der die Materialien bereitstellt, nicht mehr im Netz verfügbar ist. Bei dem Peer-to-Peer-Prinzip können Materialien nur im Netz zum Abholen bereitgestellt werden, solange der Peer im Netz verfügbar ist. Dieses Verhalten ist eine wesentliche Einschränkung, zumal bei dem Client-Server-Prinzip die Materialien längerfristig, d.h. zur Archivierung, auf dem Server gespeichert werden können. Andererseits ist beim Client-Server Prinzip der Materialaustausch nur dann möglich, wenn der Server im Netz verfügbar ist. Sollte der Server ausfallen, können die Clients keine Materialien mehr austauschen. Bei dem Peer-to-Peer-Prinzip können bei einem Ausfall eines Peers die anderen Peers trotzdem untereinander Materialien austauschen. Daher kann eine Entscheidung für einen der beiden Prinzipien davon abhängen, ob ein Server eingesetzt werden kann und wie groß der Aufwand für das Aufsetzen und Verwalten des Servers ist.
Ein anderer Entscheidungsgrund für die Auswahl eines der beiden Prinzipien ist die Sichtbarkeit der Partner im Netz. Bei dem Client-Server-Prinzip muss für die einzelnen Clients nur der Server sichtbar sein. Für den Server brauchen die Clients nicht sichtbar sein, da nur die Clients die Dienste des Servers in Anspruch nehmen und nicht umgekehrt. Bei dem Peer-to-Peer-Prinzip dagegen müssen sich die Peers, die miteinander interagieren wollen, sich gegenseitig kennen. Es kann einen erheblichen Aufwand bedeuten, die Peers untereinander in einem Netz bekannt zu machen.

4.4 Rechtfertigung warum zwei Lösungen entworfen wurden

Für das Anwendungsprofil “Lehrer-Schüler-Verhältnis“ wurden zwei Lösungen, aus den vorangegangenen Abschnitten (siehe 4.1 und 4.2), entworfen und implementiert, da beide ihre Vor- und Nachteile besitzen. Für jedes System sind Situationen vorstellbar, in dem es besser als das andere System geeignet ist.
Das System mit dem CommSy-Server ist gut geeignet, wenn während einer Schulstunde mehrere Schüler an einem LAssi ein Ergebnis erstellen. Damit alle Schüler das Ergebnis auf ihrem eigenen LAssi zur Verfügung bekommen, muss das Ergebnis nur auf den CommSy-Server abgelegt werden. Die Schüler können dann zu irgendeinem Zeitpunkt auf den CommSy-Server von ihrem LAssi, das sich auf einem Notebook in der Schule oder dem PC zu Hause befindet, über eine Internetverbindung zugreifen und das Ergebnis abholen.
Ein System mit dem ECF-Server ist dann gut geeignet, wenn sich mehrere Schüler mit ihren Notebooks in einem Raum zusammenfinden und über LAssi Materialien austauschen wollen. Da in diesem Raum keine Internetverbindung zur Verfügung steht, können die Schüler mit ihren Notebooks nur ein lokales Netzwerk einrichten. Durch die fehlende Internetverbindung haben sie keine Verbindung zu einem CommSy-Server. Über das lokale Netzwerk kann jedoch ein System mit einem ECF-Server aufgebaut und so über LAssi Materialien ausgetauscht werden.
Zudem lässt sich der Präsentationsmodus aus Kapitel 2.3 Seite 15 nur mit einem ECF-Server realisieren, da nur bei dieser Lösung Zustandsänderungen der Präsentaion direkt an andere LAssi-Clients versendet werden können. Bei der Lösung mit dem CommSy-Server können die Zustandsänderungen zwar auf dem Server abgelegt werden, der Server kann aber keine aktive Rolle einnehmen und die Zustandsänderungen anderen LAssi-Clients zuschicken. Die LAssi-Clients, die an der Präsentation teilnehmen, müssten dann ständig beim Server nachfragen, ob eine Veränderung für die jeweilige Präsentation vorliegt. Dieses Verfahren könnte für den Server etliche unnötige Anfragen zur folge haben, die den Server stark belasten könnten. Ebenfalls ist nicht sichergestellt, dass jeder Client alle Veränderungen rechtzeitig für eine Echtzeit-Präsentation erhält. Für eine zukünftige Entwicklung des Präsentationsmodus kommt daher nur die Lösung mit dem ECF-Server in Frage. Dieses rechtfertigt die ECF-Lösung.
Die zukünftige Entwicklung der verteilten Kommunikation ist mit beiden Lösungen möglich. Wie in der Tabelle Seite 25 deutlich wird, beinhaltet das CommSy ein Chat über den die verteilte Kommunikation realisiert werden kann. Da bei der ECF-Lösung von vornherein Nachrichten ausgetauscht werden, kann auch die verteilte Kommunikation mit Hilfe eines ECF-Servers umgesetzt werden.

5 Gruppenarbeit mit LAssi während einer Schulstunde

Für die CommSy- sowie für die ECF-Lösung, aus Kapitel 4, wurde in LAssi je eine neue Sicht entwickelt. Dieser Abschnitt greift das Anwendungsprofil „Lehrer-Schüler-Verhältnis“ aus Kapitel 2.3 auf und beschreibt, wie dieses mit den beiden neuen Sichten umgesetzt werden kann. Wobei einige offene Fragen aufgetreten sind, die im Abschnitt 5.3 diskutiert werden. Für die offenen Fragen werden mehrere Lösungsansätze vorgeschlagen.

Für beide Lösungen wird LAssi in einer Klasse mit ca. 20 Schülern benutzt. Der Lehrer und jeder Schüler haben während einer Schulstunde ein Notebook zur Verfügung. In der Klasse ist ein WLAN eingerichtet. Alle Notebooks haben Zugriff auf das WLAN und sind somit untereinander verbunden. Jeder Schüler hat auf seinem Notebook sein persönliches LAssi installiert, mit dem er während der Schulstunde produktiv arbeiten soll.

5.1 Gruppenarbeit mit LAssi auf Basis der CommSy-Lösung

Das lokale WLAN des Klassenraumes muss mit dem Internet verbunden sein. Der CommSy-Server kann so irgendwo außerhalb des Klassenraumes aufgestellt werden. So wird es möglich, dass mehrere Klassen oder sogar Schulen auf ein und demselben Server zugreifen. Für jede Schule kann ein CommSy-Portal angelegt werden. Innerhalb des Portals wird dann für jede Klasse der Schule ein Projektraum errichtet.
Der Lehrer und die Schüler können mit dem LAssi auf ihrem Notebook über die Internetverbindung auf den CommSy-Server zugreifen. Der Lehrer und die Schüler können in diesem Fall je eine vom WLAN automatisch zugeordnete IP-Adresse besitzen. Die IP-Adressen können auch wechseln.
Um sich an dem CommSy-Server anzumelden, muss in einer Preference Page die Internetadresse des CommSy-Servers, das Portal der Schule, der Projektraum der Klasse sowie Benutzername und das Kennwort angegeben werden.

Eine Schulstunde beginnt damit, dass der Lehrer und die Schüler ihre Notebooks anschalten und ihr LAssi starten. Nachdem LAssi gestartet ist, melden sich die Schüler und der Lehrer z.B. am Raum „Klasse 7a“ des CommSy-Servers an. Der Lehrer erstellt eine neue Karteikarte Aufgabe 1, die eine Aufgabe für die Schüler enthält. Die Aufgabe 1 zieht er per Drag and Drop in die Übersicht „Eigene bereitgestellte Materialien:“. Da eine Verbindung zum CommSy besteht wird auf dem CommSy sofort ein neues Material, dass die Aufgabe 1 beinhaltet, angelegt (siehe Abbildung 8). Der Lehrer kann in der Übersicht die Aufgabe 1 auswählen und durch betätigen des Button „Materialien einer Gruppe zuordnen“ dieses Material der Gruppe Schüler zuordnen, indem er in dem dann erscheinenden Dialogfenster die Gruppe Schüler auswählt und mit OK bestätigt.
Ein Schüler (hier als Beispiel Max Petersen), der ebenfalls am CommSy-Raum Klasse 7a angemeldet ist, kann nun in der Übersicht „CommSy-Raum: Klasse 7a“ die Gruppe „Schüler“ auswählen. Durch betätigen des Buttons „Materialien aktualisieren“ werden dann die Materialien, die dieser Gruppe zugeordnet sind, in der Übersicht „Materialien von der Gruppe: Schüler“ angezeigt. Falls eine andere Gruppe oder Person ausgewählt ist, werden die Materialien dieser Person oder Gruppe angezeigt, wenn der Button betätigt wird. Die Überschrift der Übersicht wird jeweils der ausgewählten Person oder Gruppe angepasst. Der Schüler kann nun z.B. die angezeigte Aufgabe 1 auf einen Desktop ziehen und dann betrachten. Er erstellt dann zu der Aufgabe 1 ein

Ergebnis und stellt es auf dem CommSy bereit, indem er es in die Übersicht „Eigene bereitgestellte Materialien“ zieht (siehe Abbildung 9).
Der Lehrer kann nun in der Übersicht „CommSy-Raum: Klasse 7a“ den jeweiligen Schüler (z.B. Max Petersen), von dem er das Ergebnis betrachten will, auswählen. Durch betätigen des Buttons „Materialien aktualisieren“ werden die von dem Schüler auf dem CommSy bereitgestellten Materialien in der Übersicht „Materialien von der Person: Max Petersen“ angezeigt. Der Lehrer kann dann das Ergebnis auf einen Desktop ziehen und in LAssi benutzen (siehe Abbildung 10).

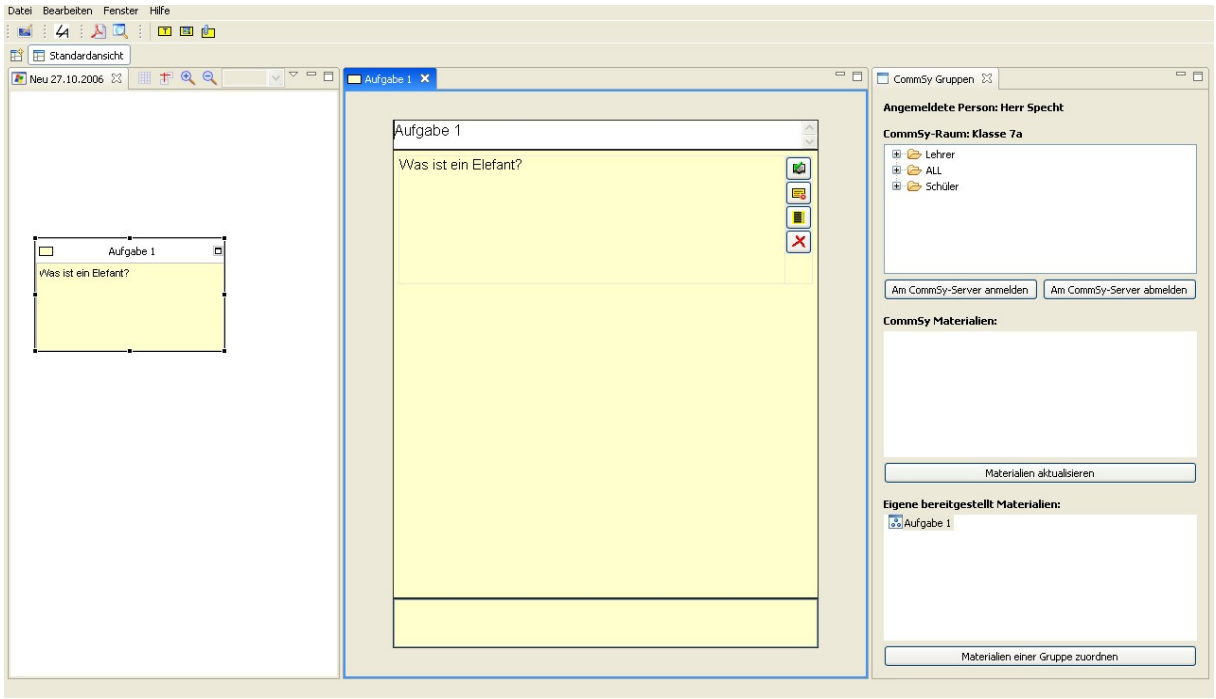

Abb. 8: LAssi-Client des Lehrers, bei dem die Aufgabe auf dem CommSy für die Schüler bereitgestellt wurde

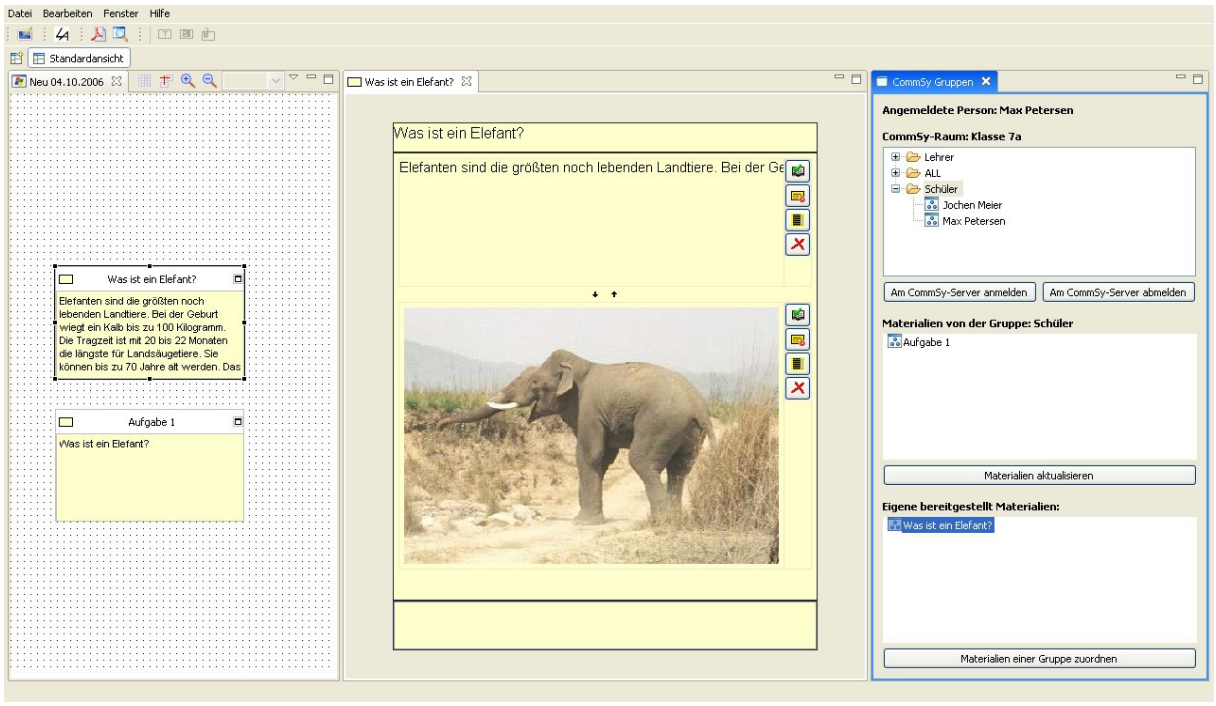

Abb. 9: LAssi-Client eines Schüler, bei dem die Aufgabe vom CommSy abgeholt und ein Ergebnis auf dem CommSy für den Lehrer bereitgestellt wurde.

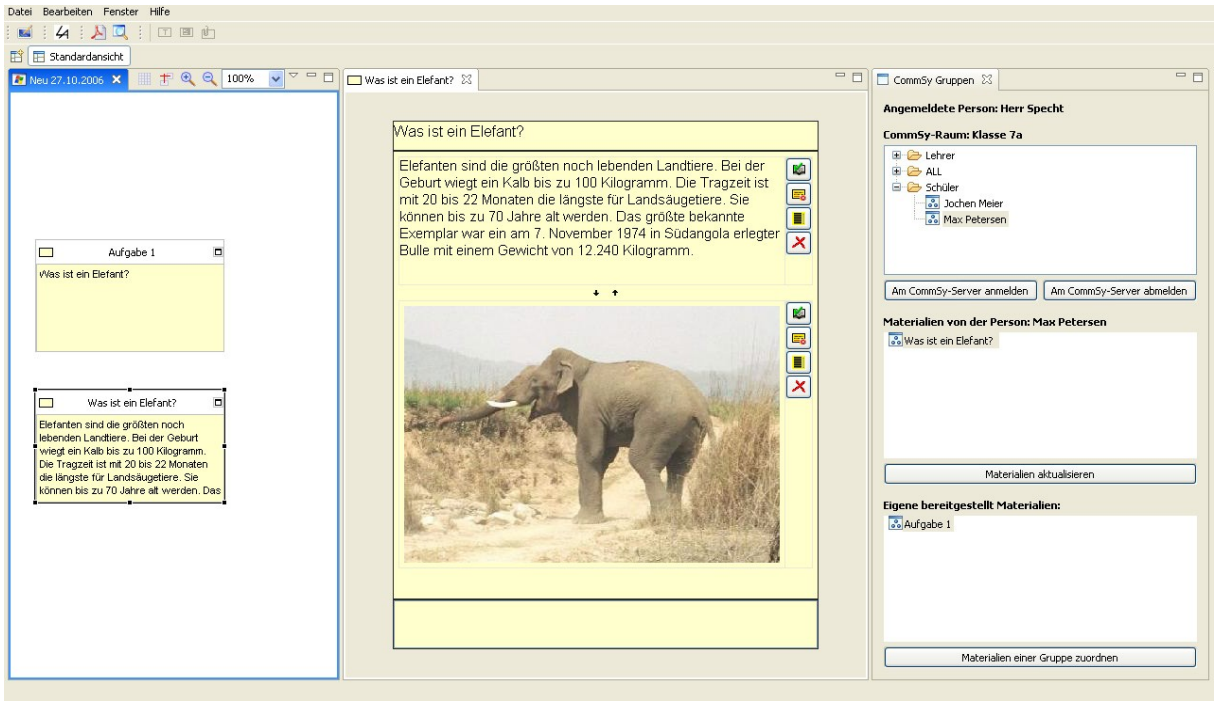

Abb. 10: LAssi-Client des Lehrers, bei dem die Ergebnisse, die von einem Schüler auf dem CommSy bereitgestellt wurden sind, auf einen Desktop kopiert wurden

5.2 Gruppenarbeit mit LAssi auf Basis der ECF-Lösung

Die Schulstunde beginnt damit, dass der Lehrer und alle Schüler ihr Notebook anschalten. Nachdem das Betriebssystem gestartet ist, startet der Lehrer sein LAssi. Zusätzlich startet er noch einen ECF-Server, der in LAssi integriert ist. Automatisch wird das LAssi des Lehrers am ECF-Server angemeldet. Die Schüler starten ebenfalls ihr LAssi und können sich nun über LAssi am ECF-Server anmelden. Die für die Anmeldung benötigten Informationen werden aus einer Preference Page geholt, die über das Menü von LAssi aufrufbar und editierbar ist.

Wenn alle Schüler und der Lehrer am ECF-Server angemeldet sind, kann mit dem Unterricht begonnen werden. Dafür erstellt der Lehrer eine Karteikarte „Aufgabe 1“, die eine Fragestellung beinhaltet. Der Lehrer wählt in der Übersicht „Gruppen und Personen“ (siehe Abbildung 11) die ganze Klasse aus. Dann zieht er die Karteikarte in der Übersicht „Abholbereit für ...“ (die Punkte stehen für die gerade ausgewählte Gruppe, hier die Schulklasse, bzw Person).

Jeder Schüler kann auf seinem eigenen LAssi über die „Gruppen und Personen“-Übersicht den Lehrer auswählen. Die Karteikarte „Aufgabe 1“ steht dann in der Übersicht „Abholbereit von der Person: Herr Specht“ zum Abholen bereit. Die Überschrift wird an die ausgewählte Gruppe oder Person angepasst. Die Schüler ziehen die Karteikarte zur Weiterverarbeitung auf einen Desktop und können so entweder eine Antwort der Karteikarte hinzufügen oder eine neue Karteikarte erzeugen, die eine Antwort auf die Aufgabe 1 enthält. Da der Lehrer noch in der Übersicht „Gruppen und Personen“ ausgewählt ist, kann die Karteikarte mit ihrer Antwort speziell dem Lehrer durch Ziehen in die Übersicht „Abholbereit für die Person: Herr Specht“ zum Abholen bereitgestellt werden (siehe Abbildung 12). Falls in der Übersicht „Gruppen und Personen“ eine andere Person oder Gruppe ausgewählt wird, werden diesen die Materialien zum Abholen bereitgestellt. Auch die Überschrift dieser Übersicht passt sich der ausgewählten Person oder Gruppe an.

Der Lehrer kann dann, nachdem die Schüler die Ergebnisse zum Abholen bereitgestellt haben, in der Übersicht „Gruppen und Personen“ die einzelnen Schüler auswählen und so in der Übersicht „Abholbereit von ...“ das Ergebnis jedes Schülers anzeigen lassen. Er kann die Ergebnisse durch einen Doppelklick direkt in LAssi öffnen oder auf einen Desktop ziehen und sie bewerten (siehe Abbildung 13) . Erst wenn die Schulstunde zu Ende ist, melden sich die Schüler und der Lehrer am ECF-Server ab. Der Lehrer muss bis dahin die Ergebnisse entweder bewertet oder auf seinem LAssi z.B. auf einem Desktop gesichert haben. Das LAssi auf den Notebooks der einzelnen Schüler muss so lange laufen, bis der Lehrer die Ergebnisse abgeholt hat. Dies ist nötig, da der ECF-Server nur als Vermittlungsrechner dient und die Ergebnisse nicht zwischenspeichern kann.

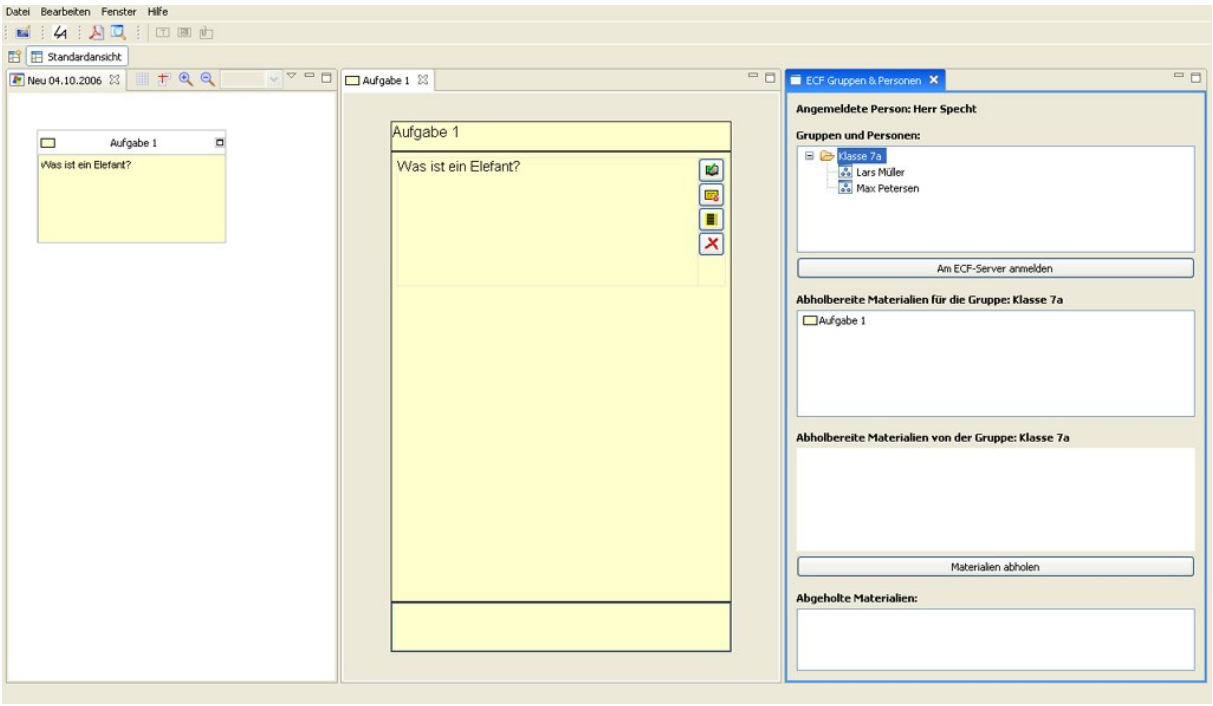

Abb. 11: LAssi-Client des Lehrers, bei dem eine Aufgabe der Klasse 7a zum Abholen bereit gestellt wurde

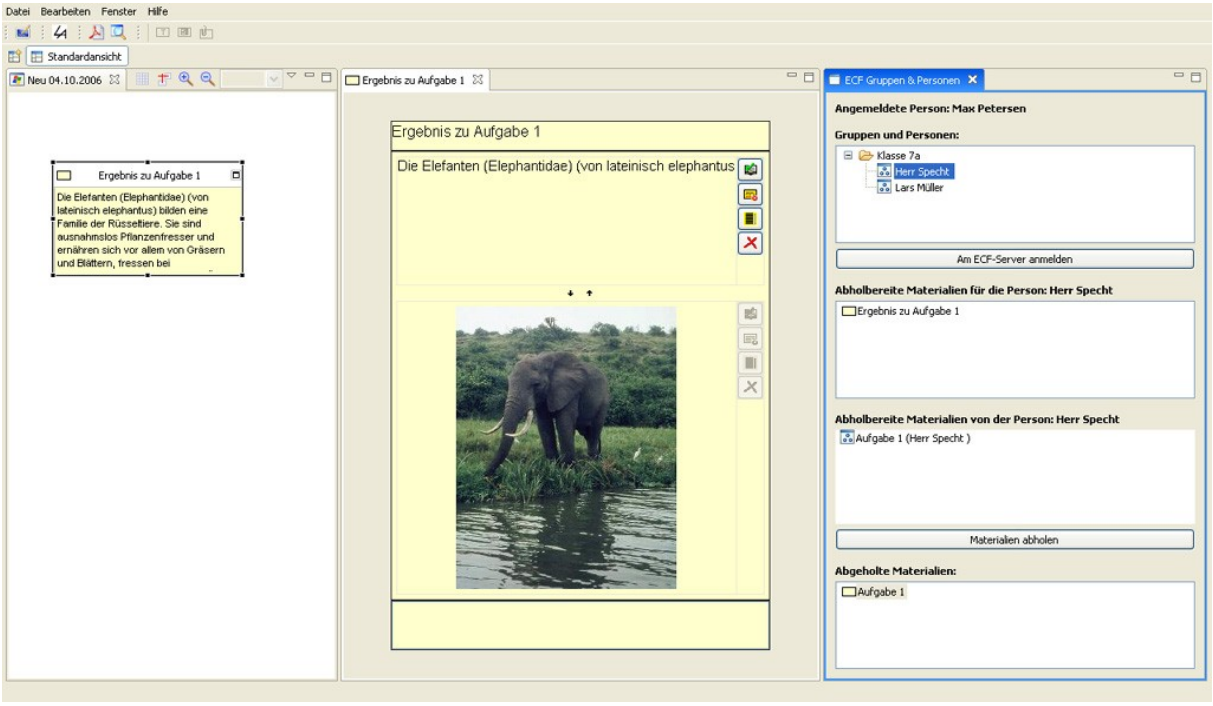

Abb. 12: LAssi-Client eines Schülers, bei dem die Aufgabe abgeholt und ein Ergebnis dem Lehrer zum Abholen bereit gestellt wurde

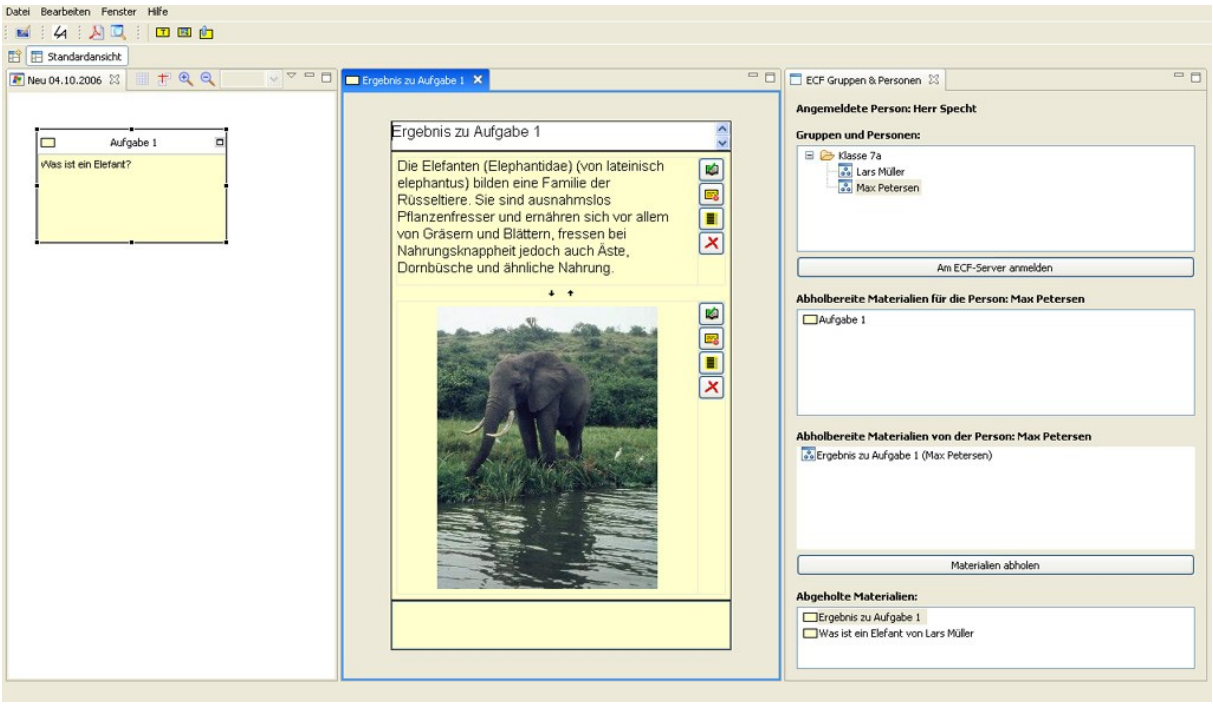

Abb. 13: LAssi-Client des Lehrers, bei dem die Ergebnisse der Schüler abgeholt sind

5.3 Offene Fragen

Nach der Implementierung der ECF-Lösung bleiben einige Fragen bzw. Problemstellungen offen, die in diesem Abschnitt andiskutieren werden. Für die offenen Fragen werden Lösungsansätze geliefert, die den Mitgliedern des Projektes „Reinventing Education - Werkzeuge für das Lernen“ als Grundlage einer weiterführenden Diskussion dienen können.
Eine offene Frage, die noch diskutiert werden muss, ist die Verwaltung der Zieladresse des ECF-Servers. In meiner Implementierung wird davon ausgegangen, dass der ECF-Server in einem lokalen Netzwerk betrieben wird. Dem Notebook, auf dem der Server gestartet wird, wird eine feste IP-Adresse in dem lokalen Netzwerk zugewiesen. Diese Adresse ist als Standardadresse in den LAssi-Clients vorgegeben. Die Clients können deshalb nur eine Verbindung zu einem ECF-Server aufbauen, der die vorgegebene IP-Adresse besitzt. Diese Variante hat den Vorteil, dass in einem LAssi-Client eines Schülers keine Zieladresse des Servers eingegeben werden muss. Der Nachteil an dieser Lösung ist jedoch, dass nur auf dem Rechner mit der vorgegebenen IP-Adresse ein ECF-Server gestartet werden kann. Falls dieser Rechner mal nicht zur Verfügung steht, könnte auf die Schnelle kein neuer Server gestartet werden.
Eine andere Variante könnte darin bestehen, dass bei jedem LAssi-Client eine Eigenschaftsseite für die ECF-Verbindung aufrufbar ist, in der die Zieladresse des Servers eingegeben werden kann. Durch diese Variante könnte im Prinzip auf jedem Notebook, das in einem lokalen Netzwerk verfügbar ist, ein ECF-Server gestartet werden. Der Nachteil dieser Variante ist, dass den Schülern die IP-Adresse des ECF-Servers mitgeteilt werden muss. Die Schüler müssten zusätzlich noch wissen, wo sie diese Adresse in ihrem LAssi eingeben können. Diese Variante ermöglicht es, dass jeder Schüler einen eigenen ECF-Server starten kann, jedoch müsste er die IP-Adresse seines Notebooks kennen und diese an die anderen Schüler weitergeben. Es stellt sich die Frage , ob dieses für Schüler jeder Altersklasse zumutbar ist.

Eine weitere Variante könnte z.B. ein Polling über den gesamten IP-Bereich des lokalen Netzwerkes sein. So könnten in diesem Fall z.B. den Notebooks des Lehrers und der Schüler eine beliebige lokale IP-Adresse zugewiesen werden. Von jedem LAssi-Client würde dann bei dem Versuch, eine Verbindung zu einem ECF-Server aufzubauen, nacheinander jede IP-Adresse abgefragt werden, ob auf diesem Notebook ein ECF-Server läuft. Läuft dort ein ECF-Server, wird eine Verbindung zwischen dem Notebook und dem Server hergestellt.
Diese Variante verbindet die beiden positiven Aspekte der ersten beiden Varianten. Denn zum einen müssen die Schüler keine Zieladresse des Servers eingeben, zum anderen kann jeder Schüler einen eigenen ECF-Server starten. Um die einzelnen ECF-Server unterscheiden zu können, kann jedem Server ein Name zugeordnet werden. Zu einem speziellen ECF-Server kann dann unter Angabe des Namens eine Verbindung hergestellt werden. So können in einem lokalen Netzwerk durchaus mehrere ECF-Server betrieben werden.
Es ist noch eine vierte Variante denkbar, beidem die im Kapitel 2 Seite 8 beschriebene Idee „LAssi on my Stick“ aufgegriffen wird. Bei dieser Variante teilen sich mehrere Schüler ein Notebook. Jeder Schüler hat einen USB-Stick mit seinem persönlichen LAssi. Die verschiedenen USB-Sticks können gleichzeitig an das Notebook angeschlossen werden. Dadurch wird es möglich, dass auf diesem Notebook die persönlichen LAssi-Clients der einzelnen Schüler parallel benutzt werden können. Zwischen diesen LAssi-Clients können dann Materialien ausgetauscht werden, in dem von einem Client aus, ein ECF-Server gestartet wird und die anderen Clients sich an diesem anmelden. Um diese Variante auch für Notebooks zu ermöglichen, die sich nicht in einem lokalen Netzbefinden und somit keine lokale IP-Adresse besitzen, kann beim Polling noch die Loopback-Adresse 127.0.0.1 benutzt werden. Dadurch können mehrere LAssi-Clients auf einem Rechner über einen ECF-Server, der ebenfalls auf diesem Rechner gestartet wurde, Materialien austauschen. Die letzten beiden Varianten könnten sogar dahingehend erweitern, dass, wenn sich ein LAssi-Client an einem ECF-Server anmelden möchte, jedoch beim Polling kein ECF-Server gefunden wird, der LAssi-Client selber einen ECF-Server startet.

Außerdem ist zu diskutieren, wann die Kopien der Materialien erzeugt werden sollen, die bei der ECF-Lösung zum Abholen bereitgestellt werden. Bei der jetzigen Implementierung wird eine Kopie zu dem Zeitpunkt erstellt, wenn per Drag and Drop das Material in die Übersicht „Abholbereit für ...“ hinzugefügt wird. Änderungen am Originalmaterial werden auch bei einem späteren Abholen der Kopie nicht berücksichtigt. Einerseits kann dieses Verhalten von den Benutzern erwünscht sein, andere Benutzer würden jedoch eher ein Verhalten erwarten, wo alle vorgenommenen Änderungen bis zum Abholzeitpunkt mitübertragen werden. Jedoch ist darauf hinzuweisen, dass der Benutzer nicht mitgeteilt bekommt, wann die Materialien abgeholt wurden. Somit hat er auch keinen Überblick, welche Änderungen in dem abgeholten Material enthalten sind. Deshalb sollte die Kopie zum Zeitpunkt des Bereitstellens erstellt werden. Möchte der Benutzer später gemachte Änderungen mitübertragen, muss er die alte Kopie aus der jeweiligen Übersicht löschen und eine neue Kopie hinzufügen.

Eine weitere Frage, die diskutiert werden sollte, ist, dass einem abgeholten Material kein Besitzer bzw. LAssi-Client, von dem das Material versandt worden ist, zugeordnet werden kann, wenn es nach dem Abholen in der Übersicht „Abgeholte Materialien“ angezeigt wird. Selbst wenn die bisher implementierte Lösung dahingehend erweitert wird, dass in der Übersicht „Abgeholte Materialien“ der sendende LAssi-Client bzw. dessen Benutzer angezeigt wird, bleibt das Problem, dass spätestens beim Einfügen dieser Materialien auf einen Desktop der Benutzer bzw. die Bezeichnung des LAssi-Clients verloren geht. Die Informationen gehen verloren, da es in LAssi bisher keine Möglichkeit gibt, den Besitzer bzw. den Ersteller eines Materials zu bestimmen und dem Material

später wieder zuordnen zu können. Mein Vorschlag wäre daher, jedem LAssi-Client eine eindeutige Identifikation z.B. eine ID zu geben. Jedem LAssi-Client wird dann genau ein Benutzer zugeordnet. Die ID und der Benutzer könnten dann zur Bestimmung des Erstellers dem Material hinzugefügt werden. Des weiteren sollte ein Material auch die ID und den Benutzernamen des LAssi-Clients enthalten, von dem das Material als letztes versandt wurde. Auf diese Weise ist es zu jedem Zeitpunkt möglich, anhand des Materials den Ersteller und den letzten LAssi-Client, von dem das Material gesendet wurde, bzw. dessen Benutzer zu ermitteln. Vielleicht wäre es noch sinnvoll zu überlegen, ob eine Liste mit der ID und dem Benutzer der LAssi-Clients, die das Material zu irgendeinem Zeitpunkt versandt haben, dem Material hinzugefügt werden sollte. Aus meiner Sicht reicht es aus, wenn nur der Benutzer und die ID des letzten LAssi-Clients, berücksichtigt werden. Mit diesen Informationen ist es auch ohne Probleme möglich, den sendenden LAssi-Client bzw. dessen Benutzer in der Übersicht „Abgeholte Materialien“ anzuzeigen oder nur die Materialien, die von einem bestimmten LAssi-Client abgeholt wurden, aufzulisten.

Bei der jetzigen implementierten ECF-Lösung wurden alle Benutzer eines LAssi-Clients an einem ECF-Server mit demselben Namen angemeldet. Diese Benutzer bilden eine Gruppe, die als Bezeichnung den Namen des Servers trägt. Es können mehrere ECF-Server mit unterschiedlichen Namen gestartet werden. Die angemeldeten Benutzer, die sich am selben ECF-Server angemeldet haben, bilden jeweils eine Gruppe. Auf diese Weise können in einem lokalen Netzwerk mehrere Gruppen gebildet werden. Es ist sogar möglich, dass sich ein Benutzer eines LAssi-Clients an mehreren ECF-Servern anmeldet und sich so zu mehreren Gruppen zuordnet. Dadurch können mehreren Gruppen und Personen, aus unterschiedlichen Gruppen, parallel Materialien zum Abholen bereitgestellt werden. Damit dieses Verhalten möglich wird, müsste die implementierte ECF-Lösung, also das PlugIn org.lassitools.ecf, dahingehend erweitert werden, dass in der Übersicht „Gruppen und Personen“ mehrere Gruppen mit ihren zugehörigen Mitgliedern angezeigt werden können.

6 Eingesetzte Techniken, Frameworks bzw. Plattformen

Bei der Implementierung, der Lösungen, die in den Kapiteln 4 und 5 beschrieben sind, wurden verschiedene Techniken, Frameworks und Plattformen eingesetzt. Die hier im Kapitel 6 kurz vorgestellt werden.
Da LAssi als Eclipse-Rich-Client-Anwendung konzipiert ist, wird in 6.1. kurz erläutert, was die Eclipse-Rich-Client-Plattform ist.
Da sich eine Lösung auf die Web Services CommSy bzw. des JCommSy bezieht, wird in 6.3 beschrieben, welche Funktionalitäten die Web Services zur Verfügung stellen.
Die zweite Lösung beruht auf dem Eclipse-Communication-Framework. Deswegen wird in 6.2 beschrieben, wofür das Framework eingesetzt werden kann.

6.1 Eclipse-Rich-Client-Platform

„Zusammen mit Java und auf dem Markt verfügbaren Java-Packages stellt das Eclipse-RCP inzwischen eine veritable Ablaufplattform für alle Arten von Anwendungen dar.“ [Dau01, Seite 5]

Das oben genannte Zitat macht deutlich was die Eclipse-RCP im Laufe der Zeit geworden ist und weist auf den folgenden Werdegang hin.
Eclipse war zuerst als eine Plattform für Softwareentwicklung geplant. Bis zur Eclipse Version 2.1 wurde Eclipse nur als Plattform für Entwicklungswerkzeuge genutzt. Aufgrund der Struktur, die auf PlugIns beruhte, wurde zur selben Zeit deutlich, dass Eclipse auch zu einer allgemeinen Plattform für Desktop- und Rich-Client-Anwendungen werden könnte. Was zu der Entwicklung einer allgemeinen Plattform noch störte, war, dass viele Komponenten von Eclipse noch eng mit dem Eclipse-Workspace verknüpft waren. Der Eclipse-Workspace dient als Datenpool für Entwicklungsartefakte, wie z.B. Klassendateien von Java Projekten. Durch eine Neuorganisation der gesamten Plattform wurde das Problem mit der Eclipse Version 3.0 gelöst. Bei dieser Neuorganisierung wurden bei fast allen PlugIns Umstellungen nötig. Heute ist die Umstellung vollzogen und Eclipse kann spätestens mit der Version 3.1 als eine allgemeine Plattform für Rich -Client-Anwendungen angesehen werden. So ist auch die Eclipse-IDE nur noch eine Rich-Client-Anwendung, die auf der Eclipse Plattform aufgesetzt ist.

Mit Eclipse 3.0 fand noch eine weitere bedeutende Veränderung an der Plattform statt. Eclipse wurde von einer proprietären Ablaufumgebung auf die OSGI-Plattform umgestellt. Die OSGI-Plattform ist eine standardisierte Ablaufumgebung für Java-Module, die auch als Bundles bezeichnet werden. Ein wesentlicher Vorteil in dieser Ablaufumgebung liegt darin, dass der Austausch von Bundles im laufenden Betrieb, das so genannte Hot Plugging, möglich wird. Der Großteil der Eclipse-RCP besteht aus Komponenten für die Benutzeroberfläche. Die beiden Komponenten SWT (Standard Widget Toolkit) und das JFace stellen grundlegende Funktionen für die Erstellung der grafischen Benutzeroberfläche zur Verfügung. Eine weitere Bibliothek, die für die Gestaltung von grafischen Benutzeroberflächen genutzt werden kann, ist die Forms API. Diese wurde ebenfalls mit Eclipse 3.0 eingeführt. Auf diese Bibliothek baut auch die generische Workbench von Eclipse auf. Mit den Komponenten der generischen Workbench können globale Menüs, Werkzeugleisten, Texteditoren, Views, Präferenzbäume, Wizards usw. genutzt werden. Diese Komponenten stehen nicht nur jeder Eclipse-Rich-Client-Anwendung zur Verfügung,

vielmehr stellen diese Komponenten die Kernfunktionen der Eclipse-RCP dar.

Optional können für eine Eclipse-Rich-Client-Anwendung noch Komponenten wie das Eclipse Hilfesystem, der Update-Manager und die Textverarbeitung eingesetzt werden. Die restlichen Komponenten der Eclipse-Plattform werden für die Eclipse-IDE gebraucht, die wie schon erwähnt als Eclipse-Rich-Client-Anwendung implementiert ist. Auf „www.eclipse.org“ ist zum einen eine Rich-Client-Plattform-Binärdistribution, die alle für den Ablauf einer Eclipse-Rich-Client-Anwendung notwendigen Komponenten enthält, und zum anderen das Eclipse-SDK, das alle Komponenten der gesamten Eclipse-Plattform beinhaltet, zu finden.

In Eclipse basiert alles auf der Basis von PlugIns. Sämtliche Funktionen, die einem Eclipse-Benutzer zur Verfügung stehen, sind in Form von PlugIns implementiert. Der eigentliche Kern von Eclipse ist sehr klein und beinhaltet nur die Komponenten, die zum Ablauf von PlugIns gebraucht werden. Module, die im Rahmen von Rich-Client-Anwendungen verwendet werden sollen, müssen daher in Form eines PlugIns vorliegen. Selbst die Anwendungslogik einer Rich-Client-Anwendung muss als PlugIn in die Eclipse-Rich-Client-Plattform eingebettet werden. PlugIns für die RCP werden in der Programmiersprache Java geschrieben. Ein PlugIn besteht daher in der Regel aus einem oder mehreren Java-Archiven. Diese Archive enthalten die benötigten Java-Klassen.

Zum weiteren Verständnis kann jeder eine einfache RCP-Anwendung erstellet werden, in dem ein neues PlugIn-Projekt in Eclipse angelegt und die Frage „Would you like to create a rich Client application?“ mit „Yes“ beantwortet wird. Daraufhin besteht die Möglichkeit, aus so genannten Templates sich eine Beispielanwendung generieren zu lassen. Die einfachste Beispielanwendung ist das Hello-RCP-Template. Dieses Template ist gut geeignet, um einen Eindruck zu bekommen, wie eine minimale RCP-Anwendung aufgebaut ist bzw. was diese alles enthalten muss.

Bis zu dieser Stelle sollte ein Eindruck vermittelt worden sein, worum es sich bei der Eclipse-Rich-Client-Plattform handelt, was ein PlugIn und was eine RCP-Anwendung ist.
Die beiden Lösungen, die implementiert wurden, sind zwei PlugIns, die in der Eclipse-Rich-Client-Anwendung LAssi eingebettet werden können. Auf die Implementierung der beiden PlugIns wird im Kapitel 7 näher eingegangen.

Weiterführende Informationen zu der Eclipse RCP sind auf der Webseite [Dau01] zu finden.

6.2 Die Web Services des CommSy

In diesem Abschnitt werden die angebotenen Funktionalitäten der Web Services des CommSy erläutert, worauf in der Einleitung schon hingewiesen wurde.
Das ursprüngliche CommSy wurde in PHP geschrieben. Das CommSy wurde im Laufe der Zeit immer weiterentwickelt, so dass zur Zeit das CommSy mit der Versionsnummer 4.4.0 zu bekommen ist. Auch die Weiterentwicklungen des CommSy werden in PHP verfasst[Com01].

Das aktuelle CommSy ist so konstruiert, dass auf das System über das Internet mit HTTP zugegriffen werden kann. Das CommSy kann von jedem Rechner mit Zugang zum Internet über einen beliebigen Web-Browser (Internet Explorer oder Mozilla Firefox) benutzt werden. Um dieses zu erreichen, wurde das CommSy auf einen Apache-Server aufgesetzt. Die Informationen über registrierte Benutzer, eingerichtete Räume und die in den einzelnen Rubriken eines Raumes abgelegten Inhalte werden vom CommSy-System in einer MySQL-Datenbank gespeichert.

Im Jahre 2005 wurde ein Projekt mit dem Namen JCommSy gegründet, dass als Thema die Migration des CommSy in die Programmiersprache Java hat. Zur Zeit gibt es einen Prototypen, der einige Rubriken in der Programmiersprache Java nachbildet. Das normale und das JCommSy sind so konstruiert, dass sich die schon in Java implementierten Rubriken in ein PHP-CommSy einbinden lassen und die jeweiligen Rubriken des PHP-CommSy ersetzen. Dabei bemerkt ein CommSy-Benutzer kaum, dass er auf beide Systeme zugreift.
Die Migration der Rubriken wird nacheinander vorgenommen, d.h. dass zuerst eine Rubrik vollständig in Java implementiert wird, bevor die nächste Rubrik in Angriff genommen wird. Dies ist möglich, da die einzelnen Rubriken unabhängig von einander sind.

Auch das JCommSy ist ein webbasiertes System, das auf einem Apache Tomcat Server der Version 5.0 aufgesetzt ist und die 2-Schichten-Architektur realisiert. Eine Server-Schicht stellt Dienste zur Verfügung, die von einem Client (Client-Schicht) benutzt werden können. Zu jeder Rubrik gibt es in der Server-Schicht einen eigenen Service, der die Dienste der Rubrik für die Client-Schicht anbietet. Die Services sorgen ebenfalls dafür, dass von der Client-Schicht übergebene Inhalte in der Datenbank gespeichert werden. Die Services bilden sozusagen die Schnittstelle zwischen der Client-Schicht und der Datenbank. Die Java-Klassen der Services enthalten Methoden, über die Inhalte der einzelnen Rubriken in Form von Java-Objekten geholt bzw. abgelegt werden können. Die Kommunikation zwischen den Services und der Datenbank erfolgt über Hibernate.

Das JCommSy ist aus mehreren Packages aufgebaut. Im Rahmen dieser Diplomarbeit ist im wesentlichen nur das Package „Service“ von Bedeutung gewesen. Dieses Package enthält die Web Service-Klassen, die auf den bestehenden Service-Klassen aufbauen und deren Methoden benutzten. Die Dienste, die von diesen Services angeboten werden, können auch von LAssi benutzt werden, um Gruppenfunktionalitäten bereit zu stellen.

Im JCommSy gibt es ein Servlet, das durch das Axis-Framework bereitgestellt wird. Alle HTTP-Anfragen, die SOAP-Anfragen beinhalten, werden an das Axis-Servlet delegiert. Das Axis-Servlet verarbeitet diese weiter und ruft die Dienste der Web Service-Klassen auf. Nach dem die HTTP-Anfrage verarbeitet wurde, wird vom Axis-Servlet eine HTTP-Antwort, die eine SOAP-Antwort enthält, generiert (siehe Abbildung 14).

In der CommSy-Lösung wurde LAssi dahingehend erweitert, dass es solche Anfragen erzeugen kann, die von dem Axis-Servlet mit Hilfe der Web Services verarbeitet werden. Nach der Verarbeitung wird dann eine HTTP-Antwort mit einer SOAP-Antwort an LAssi zurückgegeben.

Das Axis-Framework ist im Rahmen eines Open-Source-Projektes mit dem Namen „Apache Axis“ entstanden. Diese Projekt wurde Ende 2000 ins Leben gerufen. Am 22. April 2006 wurde die neueste Version unter der Nummer Apache Axis 1.4 veröffentlicht. Unter der Internetadresse sind weitere Informationen zu Apache Axis 1.4 zu finden und die aktuelle Version zum Downloaden bereitgestellt[Axis01].

Das JCommSy stellt bisher vier Web Services zur Verfügung, die für LAssi folgende wichtige Methoden bereitstellen:

MaterialWebService:
- Vector<Integer> getMaterialIDs(String username)
  liefert einen Vektor mit IDs für die Materialien, die sich in dem Raum befinden, in dem sich der Benutzer angemeldet hat.
- WSMaterial getMaterial(String username, int materialID)
  liefert das zu einer Material-ID passende WSMaterial-Objekt zurück. Das WSMaterial-Objekt enthält alle Informationen, wie z.B. Name und Autor, und Daten eines CommSy-Materials.
- void createMaterial(String username, WSMaterial mws)
  legt auf dem CommSy ein Material, das durch ein WSMaterial-Objekt beschrieben wird, in dem Raum an, an dem sich der Benutzer angemeldet hat.
- void linkMaterialToGroup(String username, int materialID, int groupID)
  ordnet das zur Material-ID passende Material der zur Gruppen-ID gehörende Gruppe zu.

GroupWebService
- Vector<Integer> getGroupsByRoom(String username, int roomID)
  liefert einen Vektor mit IDs für alle Gruppen zurück, die in dem zur roomID passenden Raum zur Verfügung stehen.
- WSGroup getGroup(String username, int groupID)
  liefert das zu einer Gruppen-ID passende WSGroup-Objekt zurück. Das WSGroup-Objekt enthält alle Informationen einer Gruppe , wie z.B. den Gruppennamen oder eine Beschreibung der Gruppe.
- Vector<Integer> getGroupMembers(String username, int groupID)
  liefert einen Vektor mit den IDs der Benutzer, die zu der übergebenen Gruppen-ID passenden Gruppe zugeordnet sind.
- Vector<Integer> getLinkedMaterials(String username, int groupID)
  liefert einen Vektor mit den Ids für die Materialien, die zu der übergebenen Gruppen-ID passenden Gruppe zugeordnet sind.

RoomWebService
- Session login(String username, String passwd, int portalID)
  meldet den übergebenen Benutzer unter Angabe eines Passwortes an das Portal, das durch die Portal-ID identifiziert wird, an und liefert eine Session zurück, die Informationen, wie die ID des Benutzers und des Raumes, an dem der Benutzer angemeldet ist, enthält.

- Session logonToRoom(String username, int roomID)
  meldet den übergebenen Benutzer an dem Raum, der zu der übergebenen Raum-ID gehört, an und gibt eine Session zurück.
- WSRoom getRoom(String username, int roomID)
  gibt das zu der übergebenen Raum-ID gehörende WSRoom-Objekt zurück. Das WSRoom-Objekt enthält alle Informationen eines CommSy-Raumes, wie z.B. Name und Beschreibung der Gruppe..

UserWebService
- WSUser getUser(String username, int userID)
  liefert das WSUser-Objekt zu der übergebenen User-ID. Das WSUser objekt enthält alle Informationen über einen CommSy-Benutzer, wie z.B. Vor- und Nachname.

Der username mit dem man sich am CommSy angemeldet hat, muss jeder anderen Methode als Parameter übergeben werden,damit das CommSy überprüfen kann, ob der Benutzer angemeldet ist und somit die angeforderten Informationen bekommen darf. Außerdem ist für einige Methoden die Angabe des Benutzernamens nötig, damit der aktuelle Raum, an dem sich der Benutzer angemeldet hat, ermittelt werden kann.

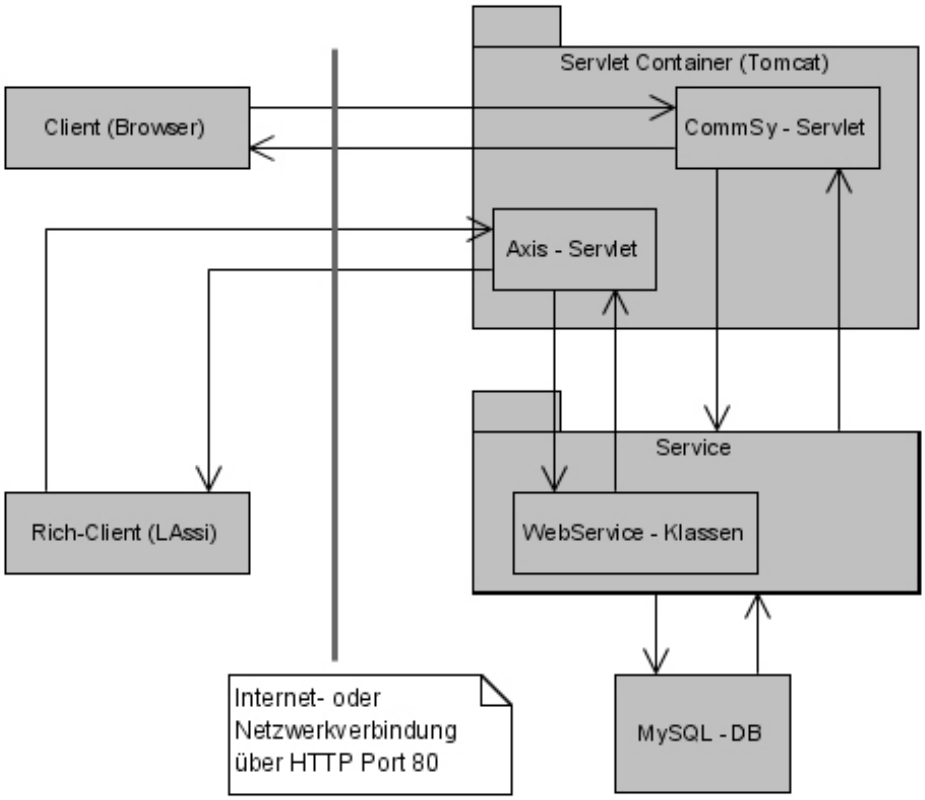

Abb. 14: Übersicht der CommSy-Architektur

6.3 Eclipse Communication Framework

Wie schon in der Einleitung zum Kapitel 6 erwähnt, wird in diesem Unterpunkt das ECF beschrieben.
Über das ECF können Peer-to-Peer Kommunikation oder auch Gruppenkommunikation, in der von einem Kommunikationspartner Informationen veröffentlicht werden, die von den anderen Partnern bezogen werden können, realisiert werden. Weitere Informationen sind auf den Webseiten [ECF01] und [ECF02] sowie in den Tutorials [ECF03] und [ECF04] zu finden.

Das ECF beinhaltet einen Standard-Server „ECF Generic Collaboration Server“, auf dem sich ECF-Benutzer zu privaten Gruppen zusammenfinden können, um gemeinsam zu arbeiten. Die Gruppenmitglieder können über den Server z.B. Ressourcen aus dem Eclipse-Workspace teilen, Nachrichten austauschen, peer file sharing und URL sharing betreiben sowie Views auf den Eclipse-Instanzen der anderen Gruppenmitglieder öffnen.

Eine Eclipse-Rich-Client-Anwendung kann eine Verbindung zu einem ECF-Server aufbauen, in dem es zuerst eine Container-Instanz über die API des ECF erzeugt wird.

Um eine Verbindung mit dem „ECF Generic Collaboration Server“ herzustellen, muss als Container-Typ `ecf.generic.client` angegeben werden. Das `IContainer`-Interface bietet zwei Methoden an. Zum einen die Methode `connect(ID targetID, IConnectContext connectContext)` und zum anderen die Methode `disconnect()`.
Der Parameter `tagetID` identifiziert den ECF-Server.

Über die Methode `getAdapter()` der `IContainer`-Instanz muss nun ein `IChannelContainer`-Objekt geholt werden. Durch die Methode `createChannel` der `IChannelContainer`-Instanz wird ein Kanal, repräsentiert durch eine `IChannel`-Instanz, erstellt, über den Daten versendet werden können. Dem Kanal wird noch ein Listener übergeben, der horcht, ob neue Daten über den Kanal versendet werden. Der Listener kann die Daten entgegen nehmen und weiter verarbeiten.

Durch den Aufruf der Methode `connect` wird eine Verbindung zu dem ECF-Server, der durch die `targetID` identifiziert wird, aufgebaut. Mit `disconnect` kann die Verbindung geschlossen werden.
Ab diesen Zeitpunkt können mit der Methode `sendMessage(byte[] data)` des `IChannel`-Objektes Daten als ByteArray an jeden Client, der mit dem ECF-Server verbunden ist, versendet werden.
Über den Aufruf von `sendMessage(ID clientID, byte[] data)` können Daten an den Client versendet werden, der durch die `clientID` identifiziert wird.

Der folgende Code zeigt, wie eine Verbindung zu einem „ECF Generic Collaboration Server“ hergestellt wird, um Daten an andere Eclipse Rich Client-Anwendungen verschicken zu können:

```
IContainer client = ContainerFactory.getDefault().createContainer(ecf.generic.client);
IChannelContainer channelContainer = (IchannelContainer)
       client.getAdapter(IChannelContainer.class);
final ID targetID = IDFactory.getDefault().createID(client.getChannelNamespace(),
       „ecftcp://localhost:3282/server1“);
ChannelListener clistener = new ChannelListener();
IChannel channel = channelContainer.createChannel(channelID,clistener, new HashMap());
client.connect(targetID, null);
channel.sendMessage(message.toByteArray());
```

7 Implementierung der Lösungen

Im diesem Kapitel der Diplomarbeit, gibt es einen Einblick in die Architektur von LAssi und es wird erläutert, wie sich die neu implementierten PlugIns in die Architektur von LAssi einfügen lassen. Zudem wird die Implementierung der CommSy- und ECF-Lösungen dargestellt..

7.1 Allgemeines zu der Architektur von LAssi

Wie schon öfters in der Arbeit erwähnt und als Erinnerung ist LAssi als Eclipse-Rich-Client Anwendung konstruiert. LAssi greift das PlugIn-Konzept von Eclipse auf und besteht daher auch aus mehreren PlugIns. Das PlugIn, über das LAssi gestartet werden kann, heißt `org.lassitools.main`. Dieses PlugIn legt fest, welche weiteren PlugIns gestartet werden sollen, und startet die Workbench von LAssi.

Zwei der wichtigsten PlugIns für LAssi sind `org.lassitools.platform` und `org.laqssitools.platform.ui`.
Das erste PlugIn beinhaltet die Logik, die benötigt wird, um zum Beispiel Materialien in LAssi zu speichern. So stellt dieses PlugIn ein `IResourceOperationService` bereit. Nur über die Schnittstelle des Services ist es möglich, LAssi-Materialien zu bekommen, zu speichern oder zu erzeugen.
Das zweite PlugIn ist für den Aufbau der Workbench zuständig. In diesem PlugIn werden z.B. das Menü und die Toolbar von LAssi erzeugt.

Zusätzlich gibt es für jedes Lernwerkzeug und Material in LAssi weitere PlugIns. Die PlugIns der Lernwerkzeuge können auf den `IResourceOperationService` des PlugIns `org.lassitools.platform` zugreifen und zunächst Referenzen auf LAssi-Materialien bekommen. Mit diesen Referenzen können dann über den `IResourceOperationService` auch die Material-Objekte geholt werden. Die Referenzen werden durch Java-Objekte mit dem Namen `ILassiResource` gebildet, die Materialien durch Java-Objekte mit dem Namen `ILassiMaterial` repräsentiert.

Um z.B. eine Karteikarte einem Sortierkasten zu übergeben, wird zuerst nur die Referenz der Karte an eines der Oberflächenelemente des Sortierkastens übergeben. Dieses Oberflächenelement würde sich über den `IResourceOperationService` das Material holen. Erst dann kann das Lernwerkzeug, hier der Sortierkasten, das Material anzeigen.

7.2 Die zwei neuen PlugIns von LAssi

Für die beiden Lösungen wurden zwei neue PlugIns entwickeln. Das `org.lassitools.commsy`-PlugIn beinhaltet die Logik und die Sicht für die CommSy-Lösung, wogegen das `org.lassitools.ecf`-PlugIn die Logik und die Sicht für die Lösung mit dem ECF-Server enthält. Beide PlugIns werden wie die Lernwerkzeuge den `IResourceOperationService` benutzen, um auf die `ILassiResource`- und `ILassiMaterial`- Objekte zugreifen zu können.

Die beiden PlugIns enthalten mehrere Packages. Das Package `View` wird die Klassen, die die Sicht erzeugen, beinhalten. In dem Package `Service` wird ein Service bereitgestellt, über den alle benötigten Informationen, die zur Verwendung des PlugIns benötigt werden, gespeichert und abgefragt werden können. Solche Informationen sind z.B. andere am Server angemeldete Personen. Über den Service können auch Methoden aufgerufen werden, die weitere Ereignisse auslösen. Die beiden Services sind nach dem Singelton-Entwurfsmuster gestaltet [Gam01, Seite 127]. Das Package `Listeners` enthält mehrere Listener, die auf Zustandsänderungen reagieren, in dem sie Ereignisse auslösen. Ein weiteres Package wird PreferencePages enthalten. Diese PreferencePages sind über das Menü von LAssi aufrufbar. Auf den PreferencePages können Informationen, die zum Anmelden am ECF- oder CommSy-Server benötigt werden, vom LAssi-Benutzer eingetragen werden.
Darüber hinaus enthält das PlugIn `org.lassitools.ecf` noch ein Package `Messages`. In diesem Package sind die Nachrichten zu finden, die über den ECF-Server verschickt werden können. Schließlich enthält das Package intern weitere Klassen, die von dem PlugIn benötigt werden. Die Abbildungen 15 und 16 zeigen die Klassendiagramme der beiden PlugIns.

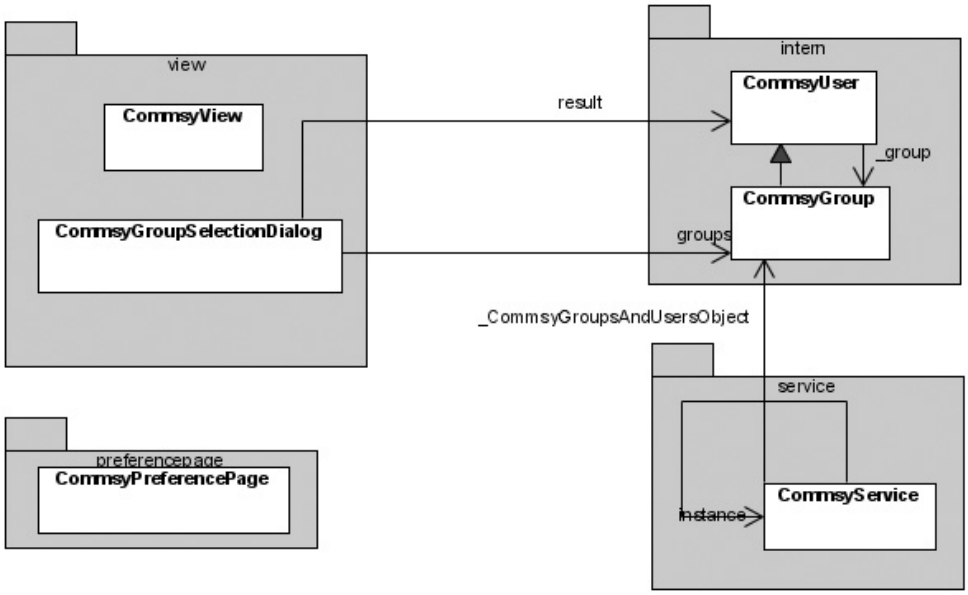

Abb. 15: Vereinfachtes Klassendiagramm für das PlugIn org.lassitools.commsy

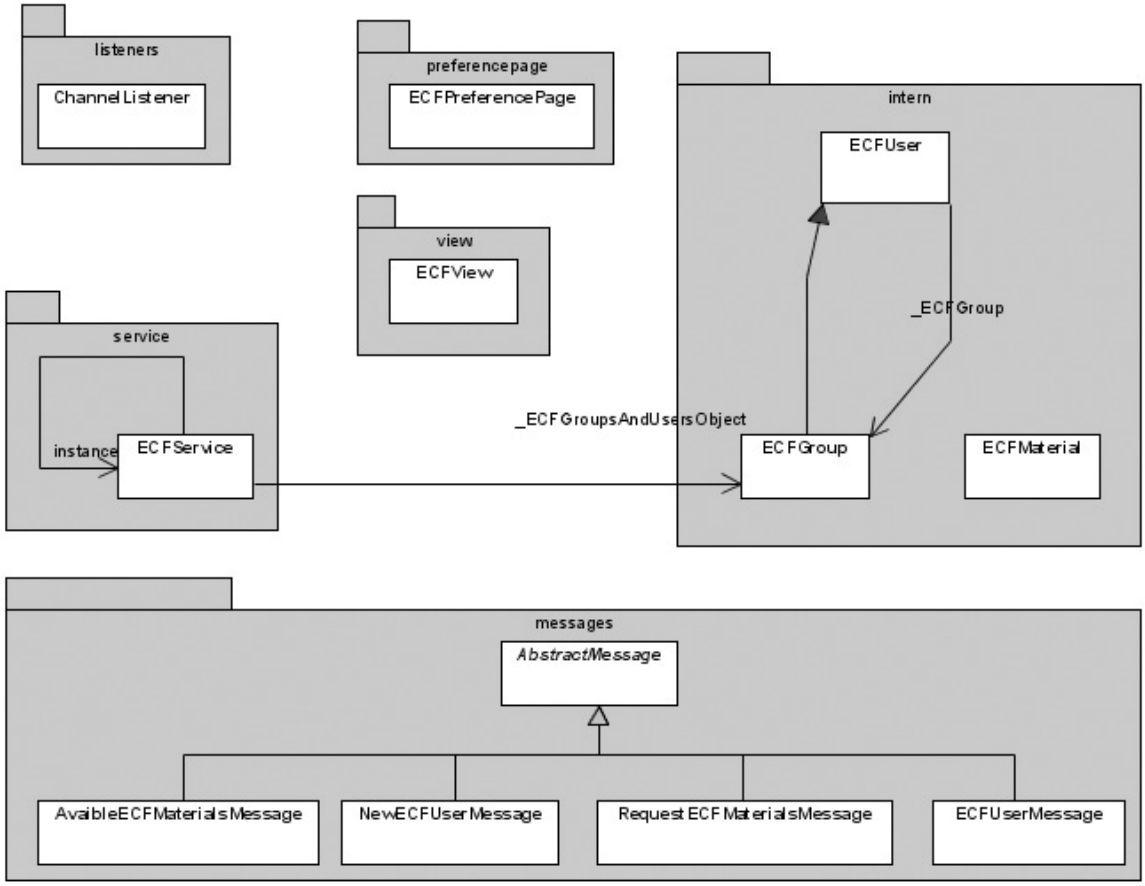

Abb. 16: Vereinfachtes Klassendiagramm für das PlugIn org.lassitools.ecf

7.2.1 Implementierung der CommSy-Lösung

Aufbauend zudem Kapitel 5.1 wird hier erklärt, welche Methoden von den Web Services aufgerufen werden müssen, um das Anwendungsprofil „Lehrer-Schüler-Verhältnis“ zu realisieren.
Um den CommSy-Raum einer bestimmten Klasse, z.B. der Klasse 7a, zu betreten, muss der LAssi-Client sich zuerst als CommSy-Benutzer dem CommSy mit Benutzername und Passwort authentifizieren. Dieses geschieht über den Aufruf der Methode `login( String username, String passwd, int portalID)` des RoomWebService. Um sich dann an einem bestimmten Raum anzumelden, muss die Methode `logonToRoom( String username, int roomID)` aufgerufen werden, die als Parameter neben den ständig benötigten Benutzernamen noch die ID für den Raum Klasse 7a übergeben bekommt.
Um in der Übersicht „CommSy-Raum: ...“ die Gruppen und Personen dieses Raumes anzeigen zu können, muss der LAssi-Client die Gruppen über die Methode `getGroupsByRoom( String username, int roomID)` abfragen und erhält einen Vektor mit den Gruppen-IDs. Um zu den Gruppen-IDs die `WSGroup`-Objekte zu bekommen muss der Client die Methode `getGroup( String username, int groupID)` für jede einzelne Gruppen-ID aufrufen.
Über die Methode `getGroupMembers( String username, int groupID)` kann sich der LAssi-Client für jede einzelne Gruppe einen Vektor mit den User-IDs der zugeordneten CommSy-Benutzer holen. Zu jeder einzelnen User-ID können die `WSUser`-Objekte über die Methode `getUser( String username, int userID)` geholt werden. Nach dem Erhalt der `WSGroup`-Objekte und der `WSUser`-Objekte ist der LAssi-Client nun in der Lage, die Gruppen und Personen des aktuellen Raumes in der Übersicht „CommSy-Raum: ...“ anzuzeigen.

Realisiert wird diese in der Methode `initialCommsyConnection()` des `CommsyService` aus dem PlugIn `org.lassitools.commsy`. In dieser Methode werden zuerst die vier CommSy-Services initialisiert. Dann werden die Informationen, die zum Anmelden benötigt werden, wie der Benutzername, das Passwort, das CommSy-Portal und der CommSy-Raum ermittelt und der Methode `mws.login(username, passwd,portal)` des MaterialWebService übergeben. Um einen CommSy-Raum zu betreten, muss die Methode `logonToRoom(username, room)` des RoomWebService aufgerufen werden. Mit der Methode `getGrooupsByRoom(username, room)` erhält man alle Gruppen des Raumes. Über alle Gruppen wird dann iteriert, um die Gruppenmitglieder zu bekommen. Die Gruppen und die Mitglieder werden dem `CommsyService` zur Aufbewahrung übergeben. Zuletzt wird noch durch die `contextChanged()` und `labelChanged()` Methoden die CommSy-Sicht aktualisiert, die dann den angemeldeten Benutzer, den CommSy-Raum und die Gruppen mit ihrem Mitgliedern anzeigt.

```
public void initialCommsyConnection () {

        rwss = new RoomWebServiceImplServiceLocator();
        mwss = new MaterialWebServiceImplServiceLocator();
        gwss = new GroupWebServiceImplServiceLocator();
        uwss = new UserWebServiceImplServiceLocator();

        rws = rwss.getCommsyRoomService();
        mws = mwss.getCommsyMaterialService();
        gws = gwss.getCommsyGroupService();
        uws = uwss.getCommsyUserService();
        String username = Activator.getDefault().getPreferenceStore()
                .getString(CommsyPreferencePage.USERNAME);
         String passwd = Activator.getDefault().getPreferenceStore()
                .getString(CommsyPreferencePage.PASSWD);
         int portal = Integer.parseInt(Activator.getDefault().getPreferenceStore()
```

```
                .getString(CommsyPreferencePage.PORTAL));
        int room = Integer.parseInt(Activator.getDefault().getPreferenceStore()
                .getString(CommsyPreferencePage.ROOM));
        Session mySession = rws.login(username, passwd, portal);
        mySession = rws.logonToRoom(username , room);
        WSRoom wsroom = rws.getRoom(username, room);
        _commsyRoom = wsroom.getName();
        Vector<Integer> groups = gws.getGroupsByRoom(username, mySession.getCurrentRoomID());
        Iterator itergroups = groups.iterator();
        while (itergroups.hasNext()) {
                Integer groupID = (Integer) itergroups.next();
                WSGroup wsgroup = gws.getGroup(username, groupID);
                addCommsyGroupToCommsyGroupsAndUsersObject(new CommsyGroup(wsgroup.getName()
                        ,wsgroup.getItemID(),wsgroup.getItemID(),CommsyUser.Typ.Group));
                Vector<Integer> users = gws.getGroupMembers(username, groupID);
                Iterator iterusers = users.iterator();
                while (iterusers.hasNext()) {
                        Integer userID = (Integer) iterusers.next();
                        WSUser wsuser = uws.getUser(username, userID);
                        addCommsyUserToCommsyGroupsAndUsersObject(new CommsyUser(wsuser.getUserName()
                                ,wsuser.getItemID(),wsgroup.getItemID(),CommsyUser.Typ.User));
                }
        }
        getCommsyViewInputChangedListener().contextChanged();
        setAnnouncedPerson(Activator.getDefault().getPreferenceStore()
                .getString(CommsyPreferencePage.USERNAME));
        _COMVChangedListener.contextChanged();
        _StatusLabelChangedListener.labelChanged();
        _CGAULabelChangedListener.labelChanged();
}
```

Um in der Übersicht „Materialien von der Person/Gruppe: ...“ Materialien, die von einer in der Übersicht „CommSy-Raum: ...“ ausgewählten Person auf dem Commsy bereitgestellt wurden, anzeigen zu können, muss zuerst die Methode `getMaterialIDs( String username)` aufgerufen werden. Von der Methode werden die IDs der Materialien, die im Raum, bei dem der Benutzer angemeldet ist, verfügbar sind, zurückgegeben. Für jede einzelne ID wird dann mit der Methode `getMaterial( String username, int materialID)` das `WSMaterial`-Objekt geholt. Ist als Autor im `WSMaterial`-Objekt der Benutzer angegeben, den man in der Übersicht „CommSy-Raum: ...“ ausgewählt hat, so wird das `WSMaterial`-Objekt berücksichtigt. Die gefilterten `WSMaterial`-Objekte werden dann dafür benutzt, die Materialien des ausgewählten Benutzers anzuzeigen.

Sollen die Materialien, die einer ausgewählten Gruppe zugeordnet wurden, angezeigt werden, so müssen die IDs der Materialien, die der Gruppe zugeordnet sind, über die Methode `getLinkedMaterials( String username, int groupID)` aufgerufen werden. Als Parameter wird die ID der Gruppe übergeben. Zu den IDs der Materialien können über die Methode `getMaterial( String username, int materialID)` die `WSMaterial`-Objekte geholt werden. Eine Filterung ist nicht nötig, da die Methode nur die IDs der gewünschten Materialien zurück gibt. Auch hier kann der LAssi-Client durch den Erhalt der `WSMaterial`-Objekte die Materialien, die zu der ausgewählten Gruppe zugeordnet sind, anzeigen.

Materialien einer Person können über die Methode `getLassiMaterialsFromCommsyForPerson (String person)` des `CommsyService` vom CommSy-Server geholt werden. In dieser Methode wird zunächst einmal ein Vector mit IDs von Materialien, die eine Person bereitgestellt hat, über den MaterialWebService geholt. Über diese IDs wird dann iteriert und zu jeder ID das `WSMaterial`-Objekt geholt. Dieses Objekt enthält Informationen über ein Material, wie z..B. den Name des Materials und den Author Diese `WSMaterial`-Objekte werden in einer Liste, die über den `CommsyService` abrufbar ist, gespeichert. Durch den Aufruf der `context.Changed()`-Methode wird die Übersicht „Bereitgestellte Materialien von der Gruppe/Person: ...“ aktualisiert.

```
public void getLassiMaterialsFromCommsyForPerson(String person) {
        String username = Activator.getDefault().getPreferenceStore()
                .getString(CommsyPreferencePage.USERNAME);
        Vector<Integer> materialIDs = mws.getMaterialIDs(username);
        Iterator materials = materialIDs.iterator();
        _CommsyMaterials = new LinkedList();
        while (materials.hasNext()) {
                Integer materialID = (Integer) materials.next();
                WSMaterial wsmaterial = mws.getMaterial(username, materialID);
                if (person.matches(wsmaterial.getAuthor())) _CommsyMaterials.add(wsmaterial);
        }
        _CMViewInputChangedListener.contextChanged();
}
```

Per Drag and Drop kann aus der Übersicht „Materialien von der Person/Gruppe: ...“ ein Material z.B. auf dem Desktop eingefügt werden. Während des Drag and Drop-Vorganges wird die Methode `getLassiMaterialFromCommsy()` des `CommsyService` aufgerufen. In dieser Methode wird durch den Aufruf der `mws.getMaterial(username, Material.getItemID()) Methode` das zu einem `WSMaterial`-Objekt passende LAssi-Material geholt. Das LAssi-Material wird dann innerhalb der for-Schleife in einem speziellen Context gespeichert. Von diesem Context kann das LAssi-Material auf den Desktop kopiert werden. Die Abbildung 17 zeigt in einem Sequenzdiagramm, welche Methoden der Web Services aufgerufen werden müssen, um die Gruppen und Personen eines CommSy-Raumes sowie die Materialien einer dieser Person zu erhalten.

```
public List<ILassiResource> getLassiMaterialFromCommsy (WSMaterial material) {
        List<ILassiResource> resourceIds = new ArrayList<ILassiResource>();
        IResourceOperationsService ros = LassiPlatform
                .getResourceOperationsService();
        IContext packageContext = ros.createContext(TempContextType
                .getInstance(), CommsyService.getInstance().printCalendar());
        String username = Activator.getDefault().getPreferenceStore()
                .getString(CommsyPreferencePage.USERNAME);
        WSMaterial wsm1 = mws.getMaterial(username, material.getItemID());
        DataHandler dh1 = wsm1.getFile();
        FileOutputStream fos = new FileOutputStream(wsm1.getFileName());
        dh1.writeTo(fos);
        fos.close();
        FileInputStream fis = new FileInputStream(wsm1.getFileName());
        packageContext = new ContextPacker().unpack(fis, null);
        File file = new File(wsm1.getFileName());
        File dest = new File(wsm1.getFileName() + ".old");
        file.renameTo(dest);
        file.delete();
        ILassiResource rootResource = ros.getRootResource(packageContext);
        ILassiMaterial rootMaterial = ros.getMaterial(rootResource);
        ILassiContainerMaterial folder = (ILassiContainerMaterial) rootMaterial;
        IContext context = this.getIContext("CommSy");
        ILassiResource rootresource = ros.getRootResource(context);
        Set<ILassiResource> originalResources = new HashSet();
        for (ILassiResource ILR : folder.getResources()) {
                ILassiResource ILR2 = ros.copy(ILR, context);
                ((ILassiContainerMaterial) ros.getMaterial(rootresource)).addResource(ILR2);
                originalResources.add(ILR2);
        }
        resourceIds.addAll(originalResources);
        ros.deleteContext(packageContext);
        fis.close();
        return resourceIds;
}
```

Alle Materialien die in der Übersicht „Eigene bereitgestellte Materialien“ eingefügt werden, werden sofort, soweit eine Verbindung zu einem CommSy-Raum hergestellt ist, auf den CommSy-Server übertragen und für andere Benutzer bereitgestellt. Um dieses zu ermöglichen, muss ein neues

`WSMaterial`-Objekt erzeugt werden. Diesem wird dann das LAssi-Material übergeben. Wenn das `WSMaterial`-Objekt erzeugt worden ist, wird es als Parameter der Methode `createMaterial( String username, WSMaterial mws)` übergeben und damit auf dem CommSy bereitgestellt.

Bei dem Vorgang wird die Methode `sendMaterialToCommsy(ILassiResource material)` des `CommsyService` aufgerufen. In dieser Methode werden die LAssi-Materialien in einen temporären Context kopiert. Von diesem Context wird ein DataHandler erzeugt. Der DataHandler wird einem neuen `WSMaterial`-Objekt übergeben. Mit der Methode `createMaterial(username , wsm)` des MaterialWebService wird auf dem CommSy ein neues Material angelegt, dass das zu bereitstellende LAssi-Material enthält.

```
public void sendLassiMaterialToCommsy(ILassiResource resource) {
        ByteArrayOutputStream output = new ByteArrayOutputStream();
        IResourceOperationsService ros = LassiPlatform
                .getResourceOperationsService();
        IContext packageContext = ros.createContext(TempContextType
                .getInstance(), printCalendar());
        ILassiResource copiedResource = ros.copy(resource,packageContext);
        StackMaterial containerMaterial = new StackMaterial();
        containerMaterial.addResource(copiedResource);
        ros.createRootResource(packageContext, containerMaterial);
        new ContextPacker().pack(packageContext, output, null);
        ros.deleteContext(packageContext);
        String username = Activator.getDefault().getPreferenceStore()
        .getString(CommsyPreferencePage.USERNAME);
        String date = printCalendar();
        FileOutputStream fos = new FileOutputStream("l" + date + ".lsi");
        fos.write(output.toByteArray());
        fos.close();
        File f = new File("l" + date + ".lsi");
        DataHandler dh = new DataHandler(new FileDataSource(f));
        WSMaterial wsm = new WSMaterial();
        wsm.setAuthor(username);
        wsm.setDescription(resource.getTitle());
        wsm.setFileName("l" + date + ".lsi");
        wsm.setPublicity(true);
        wsm.setTitle(resource.getTitle());
        wsm.setTyp("zip");
        wsm.setFileName(dh.getName());
        wsm.setFile(dh);
        mws.createMaterial(username, wsm);
}
```

Um ein eigenes bereitgestelltes Material einer Gruppe zuzuordnen, muss die Methode `linkMaterialToGroup( String username, int materialID, int groupID)` aufgerufen werden. Als Parameter wird die ID des Materials und die ID der Gruppe, die einander zugeordnet werden sollen, übergeben.

Während dieses Vorganges wird die Methode `linkMaterialToGroup(List<WSMaterial> list)` des `CommsyService` aufgerufen. In dieser Methode wird zuerst ein Dialog, in dem der Benutzer die Gruppe auswählen kann zu der die Materialien zugeordnet werden sollen, erzeugt und angezeigt. Dann wird über die übergebenen `WSMaterial`-Objekte iteriert und für jedes `WSMaterial`-Objekt die Methode `linkMaterialToGroup(username, material.getItemID(), group.getID())` des MaterialWebService aufgerufen.

```
public void linkedLassiMaterialsToGroup(List<WSMaterial> list) {
        String username Activator.getDefault().getPreferenceStore()
                .getString(CommsyPreferencePage.USERNAME);
        CommsyGroupSelectionDialog dialog = new
                CommsyGroupSelectionDialog(Display.getDefault().getActiveShell());
        CommsyUser group = null;
```

```
        if (dialog.open() == CommsyGroupSelectionDialog.OK) {
                group = dialog.selectedGroup();
                if (group != null) {
                        Iterator<WSMaterial> iterator = list.iterator();
                        while (iterator.hasNext()) {
                                WSMaterial material = iterator.next();
                                mws.linkMaterialToGroup(username, material.getItemID()
                                , group.getID());
                        }
                }
        }
}
```

Über die Methode `logout(username)`, die durch das Betätigen des Buttons „Am CommSy-Server abmelden“ aufgerufen wird, wird der Benutzer am CommSy-Server abgemeldet.

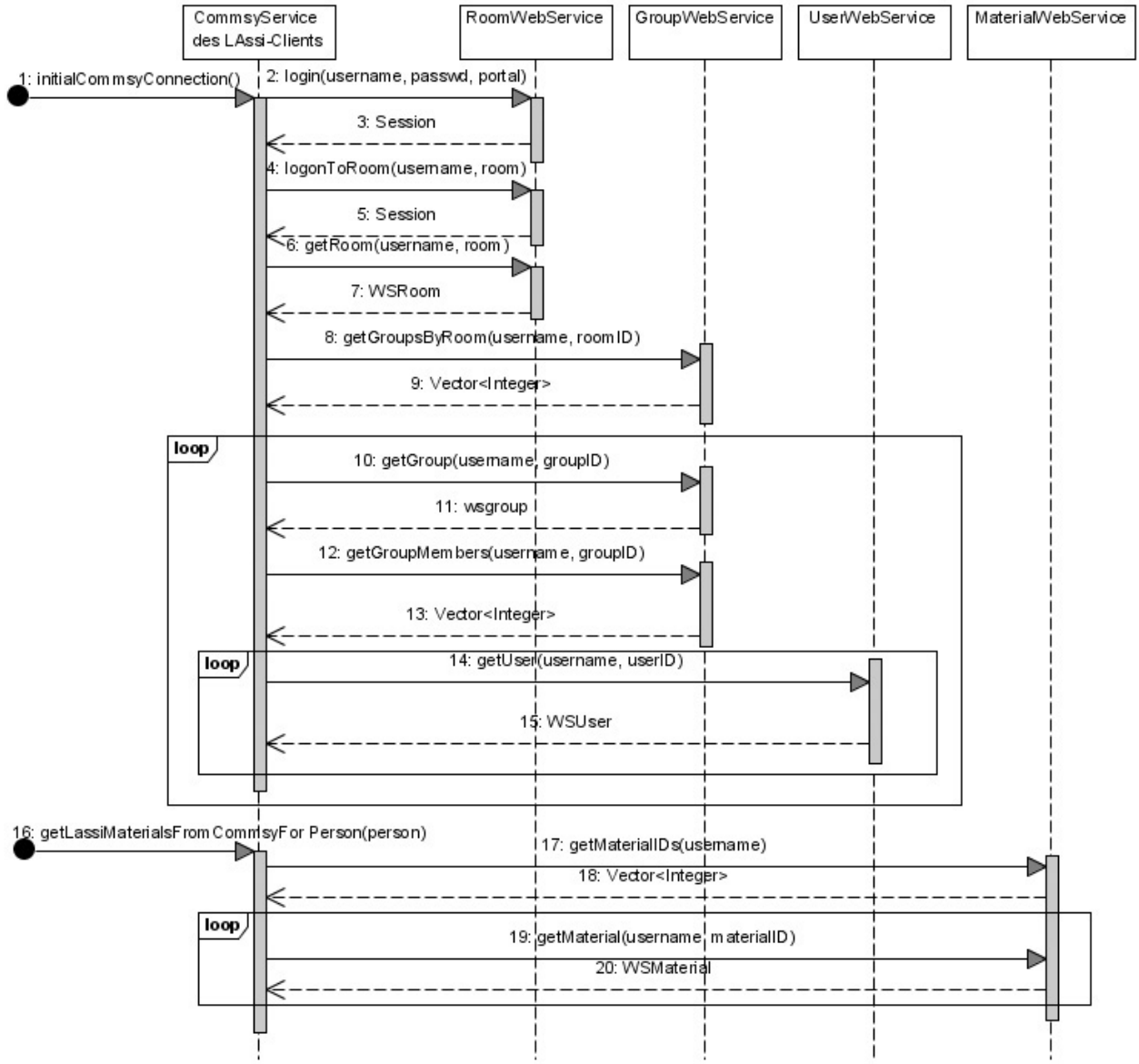

Abb. 17: Das Sequenzdiagramm 1 zeigt, welche Methoden der Web Services aufgerufen werden müssen, um die Gruppen und Personen eines CommSy-Raumes sowie die Materialien einer dieser Person zu erhalten

7.2.2 Implementierung der ECF-Server Lösung

In der ECF-Lösung, die im Kapitel 5.2 beschrieben wurde, müssen die LAssi-Clients miteinander kommunizieren. Bei der ECF-Lösung geschieht dieses, indem Nachrichten über den ECF-Server an die anderen LAssi-Clients versendet werden. Es stellt sich die Frage, wie diese Nachrichten aussehen müssen und in welchen Situationen diese dann verschickt werden müssen. Zusätzlich wird erläutert, wie das Versenden der Nachrichten in den einzelnen Situationen im PlugIn `org.lassitools.ecf` realisiert ist.

Wenn ein LAssi-Client sich an dem ECF-Server anmeldet, wird ein Kanal auf den Nachrichten versendet werden können, zwischen diesen eingerichtet. Der LAssi-Client kann über diesen Kanal Nachrichten an den ECF-Server senden. Der ECF-Server empfängt die Nachricht und leitet diese an alle anderen LAssi-Clients, die sich am Server angemeldet haben, weiter. Jeder Kanal, der zwischen einem ECF-Server und einem LAssi-Client aufgebaut wird, kann durch eine eindeutige ID identifiziert werden. Ein LAssi-Client kann eine Nachricht an einen einzelnen LAssi-Client versenden, indem er neben der Nachricht noch die ID des Kanals, der zwischen dem ECF-Server und dem Ziel-Client aufgebaut ist, an den ECF-Server übergibt. Der Server leitet die Nachricht nur an den LAssi-Client weiter, der über den Kanal, der durch die ID identifiziert wird, erreichbar ist. Jedoch ist nach der Anmeldung an einem ECF-Server nur der eigene Kanal zum ECF-Server bekannt. Die Kanäle, die von anderen LAssi-Clients und dem ECF-Server gebildet werden, sind nicht bekannt. Daher kann ein LAssi-Client zunächst nur Nachrichten an alle anderen LAssi-Clients versenden. Um Nachrichten speziell von einem zum anderen Client versenden zu können, müssen die Clients in einer Nachricht, die an alle versendet wird, ihre eigene Kanal-ID zusammen mit dem Namen des Benutzers verbreiten. Daher enthält die erste Nachricht, die an alle anderen LAssi-Clients über den ECF-Server verschickt wird, den Benutzernamen der Person, die mit dem LAssi-Client arbeitet, und die eigene Kanal-ID. Den anderen LAssi-Clients soll diese Nachricht mitteilen, dass sich der LAssi-Client vom angegebenen Benutzer am ECF-Server angemeldet hat und Nachrichten über den angegebenen Kanal an diesen geschickt werden können. Zudem ist die Nachricht als Aufforderung für die anderen LAssi-Clients gedacht, ihren eigenen Benutzernamen und die Kanal-ID zurück zu schicken. Im folgenden werde ich diese erste Nachricht als `NewECFUserMessage` bezeichnen.

Die Herstellung der Verbindung zum ECF-Server (siehe Kapitel 6.3) und das Verschicken der `NewECFUserMessage` wird in der Methode `loginToECF()` des `ECFService` realisiert.
Während der Herstellung der Verbindung wird ein `ChannelListener` und eine `channelID`, der den Kanal zwischen dem ECF-Server und dem LAssi-Client identifiziert, erzeugt. Der Listener nimmt die Nachrichten, die über den ECF-Server verschickt werden, entgegen, wertet diese aus und leitet weitere Schritte ein. Mit dem Aufruf der Methode `connect(ID targetID, IConnectContext connectContext)` des `IContainer`-Objektes wird die Verbindung zum ECF-Server hergestellt. Ab diesem Zeitpunkt können über die Methode `sendMessage(byte[] message)` des `Ichannel`-Objektes Nachrichten als ByteArray verschickt werden. In der Methode `loginToECF()` wird ein `NewECFUserMessage`-Objekt erzeugt. Dem Konstruktor dieses Objektes wird der Name des LAssi-Benutzers und die `channelID` übergeben. Das `NewECFUserMessage`-Objekt bildet die Nachricht, die mit der Methode `sendMessage(byte[] message)` versendet wird.

```
public IChannel loginToECF() throws Exception {
        _ECFGroupsAndUsersObject = new ECFGroup("ECF",null,null,ECFUser.Typ.Root);
        presenceContainer = ContainerFactory.getDefault().createContainer(
        Activator.getDefault().getPreferenceStore().getString(ECFPreferencePage.CONTAINER_TYPE));
        IChannelContainer channelContainer = (IchannelContainer)
        presenceContainer.getAdapter(IChannelContainer.class);
        final ID channelID = IDFactory.getDefault().createID(channelContainer.getChannelNamespace(),
                Activator.getDefault().getPreferenceStore().getString(ECFPreferencePage.CHANNEL_ID));
        ChannelListener clistener = new ChannelListener();
        presenceChannel = channelContainer.createChannel(channelID,clistener, new HashMap());
        presenceContainer.connect(IDFactory.getDefault().
                createID(presenceContainer.getConnectNamespace(),
                Activator.getDefault().getPreferenceStore().
                getString(ECFPreferencePage.TARGET_SERVER)), null);
        addECFGroupToECFGroupsAndUsersObject(new
        ECFGroup(GroupName,null,GroupName,ECFUser.Typ.Group));
        getGViewerInputChangedListener().contextChanged();
        NewECFUserMessage message = new NewECFUserMessage(
                Activator.getDefault().getPreferenceStore().
                getString(ECFPreferencePage.LOCAL_NAME),presenceContainer.getID());
        presenceChannel.sendMessage(message.toByteArray());
        _isConnectedWithECF = true;
        _LabelChangedListener.labelChanged();
        return presenceChannel;
}
```

Die anderen LAssi-Clients empfangen die `NewECFUserMessage` und wissen somit, dass der LAssi-Client für den übergebenen Benutzer unter der angegebenen Kanal-ID erreichbar ist. Die Clients müssen ihren eigenen Benutzer und die eigene Kanal-ID zurückschicken. Dies wird mit der Nachricht `ECFUserMessage` realisiert. Nachdem die LAssi-Clients die `NewECFUserMessage` erhalten haben, können sie in der Übersicht „Gruppen und Personen“ den Benutzer der neuen LAssi-Client anzeigen.
Da die LAssi-Clients Materialien bereitstellen, die von den anderen LAssi-Clients abgeholt werden können, müssen sie zusätzlich noch eine weitere Nachricht mit einer Liste von den bereitgestellten Materialien an den neuen LAssi-Client versenden. Neben der Liste enthält die Nachricht noch die Kanal-ID des LAssi-Clients, von dem die Materialien abgeholt werden können. Diese Nachricht wird im folgenden als `AvaibleECFMaterialsMessage` bezeichnet.

Nachrichten können von einem LAssi-Client durch den erwähnten `ChannelListener`, der bei der Herstellung der Verbindung zum ECF-Server erzeugt wird, empfangen werden. So wird die Nachricht `NewECFUserMessage` von den `ChannelListener` der einzelnen LAssi-Clients als ByteArray entgegen genommen und überprüft, ob es eine Instanz eines `NewECFUserMessage`-Objektes ist. Ist dieses der Fall, so wird der Name und die ID aus dem `NewECFUserMessage`-Objekt ausgelesen und ein neues `ECFUser`-Objekt erzeugt. Dieses wird der Methode `addECFUserToECFGroupsAndUsersObject(ECFUser ecfUser)` übergeben. Die Methode übergibt das `ECFUser`-Objekt an ein Objekt, dass aus `ECFUser`- und `ECFGroup`-Objekten besteht. Mit Hilfe des Objektes können die Gruppen und Personen der Übersicht „ECF Gruppen und Personen“ angezeigt werden. Die `ECFUser`- und `ECFGroup`-Objekte enthalten Informationen, die von den Benutzern und Gruppen, die über das ECF erreichbar sind, bekannt sind. Dazu zählen z.B. der Name einer Person, die Kanal-ID, über die der LAssi-Client der Person erreichbar ist, und Referenzen auf Materialien, die von der Person angeboten werden. Durch den Aufruf der Methode `getGViewerInputChangedListener().contextChanged()` wird die Aktualisierung der Übersicht „Gruppen und Personen“ veranlasst. Nun ist der neue ECF-User in der Übersicht zu sehen. Dem LAssi-Client vom neuen ECF-User muss nun noch der Name des eigenen Benutzers und die eigene Kanal-ID übermittelt werden. Dieses geschieht, in dem ein `ECFUserMessage`-Objekt, mit dem

eigenen Benutzernamen und der `ChannelID` erzeugt wird und über die Methode `sendMessage(byte[] message)` versendet wird. Über die Methode `sendECFListWithAvailableMaterialsToPerson(String name, ID id)` wird dann noch eine Liste mit den zum Abholen bereitgestellt Materialien für den Benutzers des neuen LAssi-Clients versendet. Diese Nachricht wird durch ein `AvaibleECFMaterialsMessage`-Objekt repräsentiert. Dieses Objekt enthält die Liste mit den zum Abholen bereitgestellten Materialien und die ID des Clients, von dem die Materialien bereitgestellt werden. Im folgenden Quellcode ist ein Ausschnitt der ChannelListener-Klasse zu sehen, der zeigt, wie die empfangene Nachricht daraufhin überprüft wird, ob sie eine Instanz des `NewECFUserMessage`-Objektes ist und bei Erfolg weitere Schritte eingeleitet werden.

```
if (message instanceof NewECFUserMessage) {
        String User = ((NewECFUserMessage) message).getUser();
        ECFService ecfService = ECFService.getInstance();
                ecfService.getInstance().addECFUserToECFGroupsAndUsersObject(
                new ECFUser(User,((NewECFUserMessage) message).getID(), ecfService
                .getPresenceChannel().getID().getName(),ECFUser.Typ.User));
        ecfService.getGViewerInputChangedListener().contextChanged();
        ECFUserMessage usermessage = new ECFUserMessage(Activator
                .getDefault().getPreferenceStore().getString(ECFPreferencePage.LOCAL_NAME),
                ecfService.getPresenceContainer().getID());
        ecfService.getPresenceChannel().sendMessage(usermessage.toByteArray());
        ecfService.sendECFListWithAvailableMaterialsToPerson(User,((NewECFUserMessage)
        message).getID());
}
```

Wenn der neue LAssi-Client die `ECFUserMessage` empfängt, weiß er über welche Kanal-ID der LAssi-Client des in der `ECFUserMessage` angegebenen Benutzern erreichbar ist und kann ebenfalls durch die `AvaibleECFMaterialsMessage` diesem LAssi-Client mitteilen, welche Materialien er für dessen Benutzer anbietet.
Für den Fall, dass sich auf irgendeinem LAssi-Client die zum Abholen bereitgestellten Materialien für eine Person ändern, muss der LAssi-Client eine neue `AvaibleECFMaterialsMessage` an den LAssi-Client des Benutzer schicken, für den sich die bereitgestellten Materialien geändert haben. Erst durch den Erhalt der `AvaibleECFMaterialsMessage` können in der Übersicht „Abholbereite Materialien für die Person: ...“ die Materialien, die von der jeweiligen Person angeboten werden, angezeigt werden.

Das Empfangen der Nachrichten `ECFUserMessage` und `AvaibleECFMaterialsMessage` wird ebenfalls im `ChannelListener` realisiert. Ist die empfangene Nachricht eine Instanz des `ECFUserMessage`-Objektes, dann wird wie schon oben beschrieben ein `ECFUser` erstellt und dem `ECFService` übergeben sowie die Übersicht „Gruppen und Personen“ aktualisiert. Durch die unterschiedlichen Message-Typen wird ein zyklischer Nachrichtenversandt verhindert. Schließlich wird wieder eine `AvaibleECFMaterialsMessage`-Nachricht erzeugt, die eine Liste mit bereitgestellten Materialien für den Benutzer enthält, von dessen LAssi-Client die `ECFUserMessage` gesendet wurde.

```
if (message instanceof ECFUserMessage) {
        String User = ((ECFUserMessage) message).getUser();
        ECFService ecfService = ECFService.getInstance();
        ecfService.addECFUserToECFGroupsAndUsersObject(
                new ECFUser(User, ((ECFUserMessage) message).getID(), ecfService
                .getPresenceChannel().getID().getName(),ECFUser.Typ.User));
        ecfService.getGViewerInputChangedListener().contextChanged();
        ecfService.sendECFListWithAvailableMaterialsToPerson(User, ((ECFUserMessage)
        message).getID());
}
```

Wenn die empfangene Nachricht eine Instanz des `AvaibleECFMaterialsMessage`-Objektes ist, dann wird die Liste und die ID aus der Message ausgelesen und als Parameter der Methode `setECFListWithAvaibleMaterialsFromUser(Set<String> set, ID id)` übergeben. Diese Methode sorgt dafür, dass die Liste mit den angebotenen Materialien in dem `ECFUser`-Objekt gespeichert wird, dessen ID mit der ID aus der Nachricht übereinstimmt.

```
if (message instanceof AvaibleECFMaterialsMessage) {
	AvaibleECFMaterialsMessage ECFRM = (AvaibleECFMaterialsMessage) message;
	ECFService.getInstance().setECFListWithAvaibleMaterialsFromUser(ECFRM.getListWithObject()
		,ECFRM.getid());
}
```

Wenn der Übersicht „Abholbereit für ...“ Materialien hinzugefügt werden, muss dem LAssi-Client des Benutzers, für den die Materialien bereitgestellt werden sollen, eine `AvaibleECFMaterialsMessage` zugeschickt werden, die eine aktualisierte Liste mit den angebotenen Materialien enthält.

Das Versenden der Nachricht `AvaibleECFMaterialsMessage` wird in der Methode `drop(DropTargetEvent event)` realisert, wenn eine oder mehrere Materialien per Drag and Drop in die Übersicht „Abholbereit für ...“ eingefügt werden. Der Aufruf der Methode sorgt dafür, dass die per Drag and Drop überlieferten Materialien an ein Context übergeben werden. Ein Context ist ein von LAssi bereitgestelltes Objekt, in dem Materialien gespeichert werden können. Für jeden bekannten Benutzer eines LAssi-Clients gibt es einen solchen Context, in dem die für ihn zum Abholen bereitgestellten Materialien gespeichert werden. Diese Contexte werden auch benutzt, um den Inhalt der Übersicht „Abholbereit für ...“ darzustellen. Da sich der Context verändert hat, muss auch die Übersicht durch den Methodenaufruf `getAMFViewerInputChangedListener().contextChanged()` aktualisiert werden. Zudem wird in der Methode `sendECFListWithAvailableMaterialsToPerson(ecfUser)` eine neue `AvaibleECFMaterialsMessage` -Nachricht erzeugt. Diese Nachricht erhält eine aktualisierte Liste mit den abholbereiten Materialien für die Gruppe oder Person, die in der Übersicht „ECF Gruppen und Personen“ ausgewählt ist.

```
public void drop(DropTargetEvent event) {
	ECFService ecfService = ECFService.getInstance();
	List<ILassiResource> resources = new LinkedList<ILassiResource>();
	ILassiResource[] attachments = (ILassiResource[]) internalTransfer
		.nativeToJava(event.currentDataType);
	IResourceOperationsService ros = LassiPlatform
		.getResourceOperationsService();
	IContext context = null;
	IStructuredSelection ISS = (IStructuredSelection) _GroupViewer.getSelection();
	ECFUser ecfUser = ((ECFUser) ISS.getFirstElement());
	context = ecfService.getIContext(ecfUser.getName());
	ILassiResource rootresource = ros.getRootResource(context);
	for (ILassiResource attachment : attachments) {
		ILassiResource ILR = ros.copy(attachment, context);
		((ILassiContainerMaterial) ros.getMaterial(rootresource)).addResource(ILR);
	}
	ecfService.getAMFViewerInputChangedListener().contextChanged();
	ecfService.sendECFListWithAvailableMaterialsToPerson(ecfUser);
}
```

Falls für einen bestimmten Benutzer bereitgestellte Materialien über dessen LAssi-Client abgeholt werden sollen, muss der LAssi-Client eine `RequestECFMaterialsMessage` an den LAssi-Client, auf dem sich die Materialien befinden, versenden. Die `RequestECFMaterialsMessage` enthält eine Liste mit den Materialien, die abgeholt werden sollen, und die Kanal-ID des LAssi-Clients, zu dem die Materialien geschickt werden sollen.

Diese wird durch den Aufruf der Methode `sendRequestECFMaterialsMessageToPerson (Set<String> set, ECFUser ecfUser)` des `ECFService` realisiert. Diese Methode erzeugt ein `RequestECFMaterialsMessage`-Objekt mit der Kanal-ID des eigenen LAssi-Clients, dem Namen des eigenen Benutzers und eine Liste mit Referenzen auf die ausgewählten Materialien.
Über den Methodenaufruf `sendMessage(ecfUser.getECFID(),RM.toByteArray())` wird das `RequestECFMaterialsMessage`-Objekt als Nachricht an den LAssi-Client desjenigen Benutzers geschickt, von dem die Materialien abgeholt werden sollen.

```
public void sendRequestECFMaterialsMessageToPerson(Set<String> set, ECFUser ecfUser) {
        RequestECFMaterialsMessage RM = new
                RequestECFMaterialsMessage(getPresenceContainer().getID(),
                getAnnouncedPerson(),set);
        if (ecfUser.getTyp().equals(ECFUser.Typ.User)) {
                getPresenceChannel().sendMessage(ecfUser.getECFID(),RM.toByteArray());
        } else if (ecfUser.getTyp().equals(ECFUser.Typ.Group)) {
                getPresenceChannel().sendMessage(RM.toByteArray());
        }
}
```

Wenn der LAssi-Client, von dem die Materialien bereitgestellt werden, die Nachricht `RequestECFMaterialsMessage` empfängt, sendet er die gewünschten Materialien als Nachricht an den LAssi-Client zurück, dessen Kanal-ID in der `RequestECFMaterialsMessage` angegeben ist.

Der `ChannelListener` des Ziel-Clients führt, wenn die empfangene Nachricht eine Instanz der `RequestECFMaterialsMessage` ist, die Methode `sendECFMaterialsToTarget(Set<String> set, String name, ID id)` aus. Als Parameter wird die empfangene Nachricht übergeben. In dieser Methode wird zuerst der `IResourceOperationService` des PlugIns `org.lassitools.platform` geholt. Über die Methode `getIContext(name)` wird der Context geholt, in dem die Materialien gespeichert werden, die für die Person, die die `RequestMessage`-Nachricht gesendet hat, zum Abholen bereitgestellt sind. Da nicht alle Materialien, sondern nur die Gewünschten, aus diesem Context versendet werden sollen, müssen die Materialien gefiltert und in einen neuen Context kopiert werden. Über den Methodenaufruf `ContextPacker().pack(packageContext, output, null)` kann ein Context in ein ByteArrayOutputStream umgewandelt werden. Dieser ByteArrayOutputStream kann als Nachricht über die Methode `sendMessage(id, output.toByteArray())` dem LAssi-Client, von dem die Anfrage kam, zugeschickt werden.

```
public void sendECFMaterialsToTarget(Set<String> set, String name, ID id) {
        IResourceOperationsService ros = LassiPlatform.getResourceOperationsService();
        IContext context = null;
        context = getIContext(name);
        Iterator<String> iter = set.iterator();
        Set<ILassiResource> resources = new HashSet<ILassiResource>();
        while (iter.hasNext()) {
                resources.add(ros.getResource(context, iter.next()));
        }
        ByteArrayOutputStream output = new ByteArrayOutputStream();
        IContext packageContext = ros.createContext(TempContextType
                .getInstance(), printCalendar());
        List<ILassiResource> copiedResources = ros.copy(resources,
```

```
            packageContext, new NullProgressMonitor());
        StackMaterial containerMaterial = new StackMaterial();
        containerMaterial.addResources(copiedResources);
        ros.createRootResource(packageContext, containerMaterial);
        new ContextPacker().pack(packageContext, output, null);
        ros.deleteContext(packageContext);
        FileOutputStream file = new FileOutputStream("C:/Lassi1.txt");
        file.write(output.toByteArray());
        getPresenceChannel().sendMessage(id, output.toByteArray());
        }
```

Die Nachricht mit den Materialien wird von dem anfordernden LAssi-Client erhalten. Der LAssi-Client kann die Materialien nun in der Übersicht „Abgeholte Materialien“ anzeigen. Von dieser Übersicht können die angezeigten Materialien z.B. auf einen Desktop gezogen werden. Durch einen Doppelklick wird das Material in einem Editor angezeigt. Das Sequenzdiagramm in der Abbildung 18 macht nochmal deutlich, welche Nachrichten zwischen den LAssi-Clients ausgetauscht werden.

Wird die Nachricht mit den Materialien empfangen, wird der empfangene ByteArrayOutputStream als Parameter der Methode `saveECFMaterials(ByteArrayOutputStream bout)` des `ECFService` übergeben. In dieser Methode wird der versendete ByteArrayOutputStream als ByteArray der Methode `ContextPacker().unpack(Byte[] bout)` übergeben und so ein neuer Context mit den angeforderten Materialien erzeugt. Die Materialien des neuen Contextes werden in einen speziellen Context kopiert, der alle empfangenen Materialien speichert. Der spezielle Context repräsentiert auch den Inhalt der Übersicht „Abgeholte Materialien“. Durch den Aufruf der Methode `contextChanged()` des `_FMViewerInputChangedListener`-Objektes wird die Übersicht „Abgeholte Materialien“ aktualisiert. Die neuen Materialien sind nun in der Übersicht „Abgeholte Materialien“ aufgelistet und sind damit in LAssi zur Weiterverarbeitung bereitgestellt.

```
public void saveECFMaterials(byte[] b) {

        ... Konvertierung des ByteArrays in ein FileInputStream

        IContext packageContext = new ContextPacker().unpack(fis,null);
        IResourceOperationsService ros = LassiPlatform
                .getResourceOperationsService();
        ILassiResource rootResource = ros.getRootResource(packageContext);
        ILassiMaterial rootMaterial = ros.getMaterial(rootResource);
        ILassiContainerMaterial folder = (ILassiContainerMaterial) rootMaterial;
        IContext context = ContextUtil.RECIEVERESOURCESDIRECTORY_CONTEXT;
        ILassiResource rootresource = ros.getRootResource(context);
        for (ILassiResource ILR : folder.getResources()) {
                ILassiResource ILR2 = ros.copy(ILR, context);
                ((ILassiContainerMaterial) ros.getMaterial(rootresource)).addResource(ILR2);
        }
        ros.deleteContext(packageContext);
        _FMViewerInputChangedListener.contextChanged();
}
```

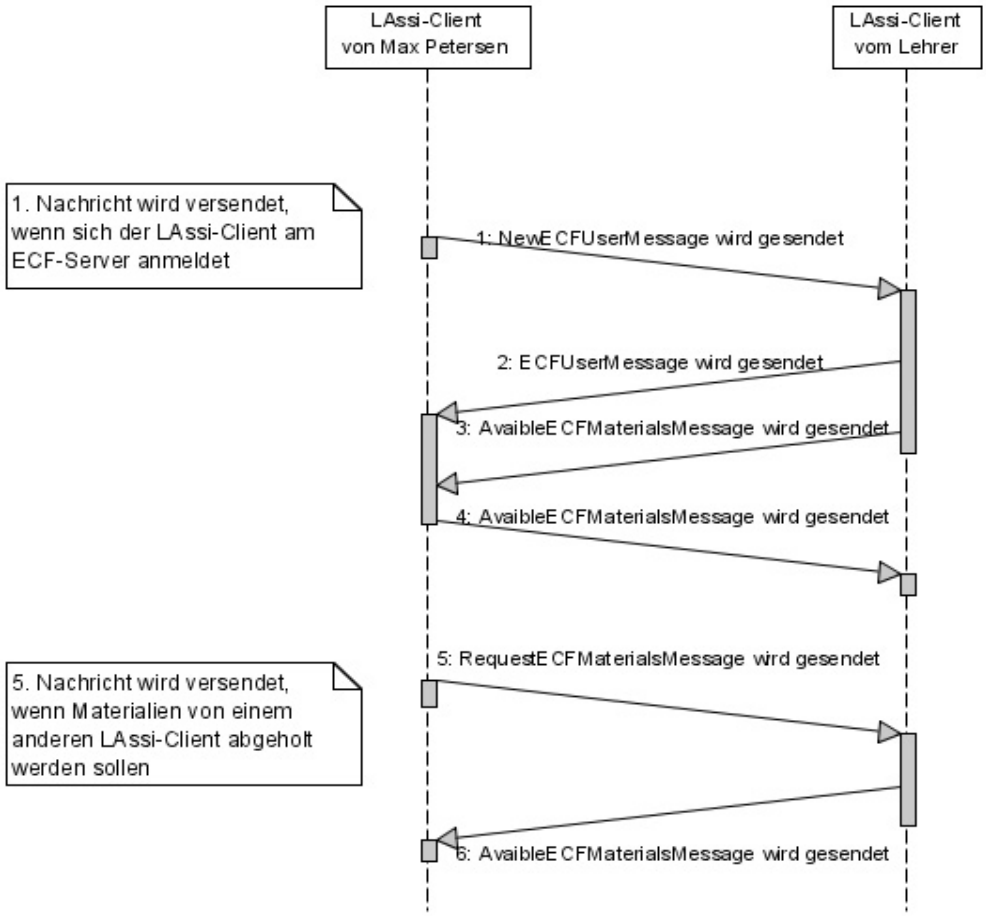

Abb. 18: Das Sequenzdiagramm zeigt, welche Nachrichten zwischen zwei LAssi-Clients ausgetauscht werden

8 Schlusswort

Am Ende der Diplomarbeit wird auf die Entstehung der Aufgabenstellung, die Entwicklungsphase, das anwendbare Ergebnis und die Probleme bei der Diplomarbeit eingegangen. Zudem werde ich mein persönliches Fazit wiedergeben.

8.1 Die Entstehung der Aufgabenstellung dieser Diplomarbeit

Das Interesse zu dieser Diplomarbeit entstand während eines Treffens, wo Diplomarbeitsthemen vorgestellt wurden u.a auch zu LAssi.
In einem zweiten Treffen, mit den Mitglieder von dem Projekt „Reinventing Education - Werkzeuge für das Lernen“, ist das Interesse weiter angewachsen und die Entscheidung bei dem Projekt mitzuwirken ist schnell gefallen.
Die aus dem Treffen entstandene Aufgabenstellung für diese Diplomarbeit war, eine Gruppenfunktionalität für LAssi zu entwerfen und zu implementieren.
Die Aufgabenstellung beinhaltete zu einem Anwendungsprofil eine Lösung zu entwerfen und diese zu implementieren.

8.2 Die Entwicklungsphase

Nach der gefundenen Aufgabenstellung galt es sich als erstes in die Thematik einzuarbeiten. Das hieß sich mit der vorhandenen LAssi Software und den Anwendungsprofilen eingehend zu beschäftigen. Hinzu kam noch, die Einarbeitung in das CommSy, dem BSCW und dem ECF. Um dann mit voller Energie und Tatendrang in die Aufgabenstellung einzusteigen.
Bei der Einarbeitungsphase, haben sich zwei Lösungen heraus kristallisiert, zum einen die CommSy-Lösung und zum anderen die ECF-Lösung. Während dieser Phase hat sich herausgestellt, dass das BSCW zum einen kostenpflichtig und des weiteren dem CommSy sehr ähnlich wäre und deswegen eine Entscheidung gegen das BSCW gefallen ist.
Mit der Implementierung der ECF-Lösung wurde zu erst begonnen. Um die CommSy-Lösung implementieren zu können, fehlten zu dem Zeitpunkt noch einige wichtige Funktionen an den Web Services.
Während der Implementierungsphase der ECF-Lösung wurden mehrere Entwicklungszustände den Mitgliedern des Projektes vorgestellt. Die haben ihrerseits wertvolle und brauchbar Kritik zu den Lösungen beigesteuert. Dieses Vorgehen wurde bis zur erfolgreichen ECF-Lösung beibehalten. Genauso verhielt es sich bei der Implementierung der CommSy-Lösung, welche starten konnte, als die Entwicklung der Web Services fertig gestellt war. Ab diesem Zeitpunkt wurde an beiden Lösungen parallel erfolgreich gearbeitet.
Bei der Implementierung der CommSy-Lösung, war es notwendig mit dem Entwickler des Web Services zusammen zu arbeiten. Der Grund der Zusammenarbeit war, dass die Web Services auf die Bedürfnisse von LAssi angepasst werden mussten, ohne die die CommSy-Lösung nicht realisierbar gewesen wäre.

8.3 Anwendbare Ergebnisse der Diplomarbeit

Als anwendbares Ergebnis dieser Diplomarbeit hat sich herausgestellt, dass nicht nur eine Lösung, wie in der Aufgabenstellung gefordert, praktisch sinnvoll ist, sondern zwei PlugIns. Die beiden PlugIns der Lernumgebung LAssi stellen jeweils eine Lösung für das Anwendungsprofil „Lehrer-Schüler-Verhältnis“ dar.
In dem ersten PlugIn, wird das Anwendungsprofil realisiert, indem die auf die Web Services des CommSy zugreifen, um Materialien auszutauschen.
In der zweiten Lösung wird das Anwendungsprofil umgesetzt, indem die LAssi-Clients über einen ECF-Server Nachrichten versenden, um Materialien untereinander auszutauschen.
Die beiden PlugIns wurden den Mitgliedern des Projektes vorgestellt. Die PlugIns sind soweit fertiggestellt, dass sie nach einer Anpassung an die aktuellste LAssi-Version in den nächsten Prototyp übernommen werden können.

8.4 Probleme während der Diplomarbeit

In der Erarbeitung der zwei Lösungen haben sich einige wenige Probleme ergeben, die erfolgreich gelöst wurden. Ein Problem war die Einarbeitung in LAssi, die eine fertige und komplexe Software ist und somit wenig Flexibilität in einigen Bereichen der Implementierung bietet. Am schwierigsten war der Weg auf die Materialien von LAssi zugreifen zu können. Hierbei konnten die Entwickler von LAssi weiterhelfen und somit das Problem gelöst werden. Durch diese Hilfestellung war es ein leichtes an die Materialien zu kommen.
Des weiteren, war es schwierig sich in die Eclipse-RCP einzuarbeiten, da es ein komplett neues Gebiet der Implementierung für mich war. Die Eclipes-RCP zu verstehen hat viel Zeit in Anspruch genommen.
Des letzteren sind kleinere Komplikationen während der Implementierungsphase aufgetreten, die durch die Bücher [Dau01] und [Dau02] aufgehoben werden konnten. Das Buch [Dau02] diente allein der Problemlösungen während des Implementierens.
Somit waren alle kleineren und größeren Probleme erfasst und wurden gelöst.

8.5 Persönliches Fazit

Als erstes möchte ich erwähnen, das mir die Arbeit und Entwicklung an LAssi sehr viel Spaß und Freude gemacht hat. Durch die Weiterentwicklung habe ich zudem an Erfahrung gewonnen und kann diese hoffentlich in meinen späteren beruflichen Werdegang einbringen.
Es würde mich sehr freuen, wenn ich weiterhin für das Projekt tätig sein könnte und somit meine Erfahrungen beisteuern kann, da mir LAssi sehr ans Herz gewachsen ist.
Darauf bezogen war es mir sehr wichtig eine funktionierende Lösung zu implementieren, die ich dem Projekt zur Verfügung stellen kann. Was mir zur Freude aller Mitglieder des Projektes gelungen ist. Das ist auch der Grund warum sich meine Diplomarbeit eher technisch entwickelt hat, als konzeptionell.
Wie schon erwähnt war mir sowie den Mitgliedern die Implementierung für das Vorankommen von LAssi sehr wichtig. Deshalb habe ich den größten Teil meiner Planung auf die Entwicklung verwendet, wodurch leider die Zeit zur detaillierten Ausarbeitung der konzeptionellen Möglichkeiten zu kurz kam. Nichts desto trotz bin ich der Meinung, eine alle zufriedenstellende Diplomarbeit angefertigt zu haben.

Literaturverzeichnis

[Axis01] Apache Axis 1.4 Homepage, *http://ws.apache.org*

[BSCW01] BSCW Homepage, *http://www.bscw.de*

[BSCW02] BSCW 4.3 Handbuch, *http://www.bscw.de/download/bscw_help_43_de.pdf*

[BSCW03] Admin Manual BSCW V4.3, http://www.bscw.de/download/AdminManual43.pdf

[Ben01] Bengel, G., *Grundkurs Verteilte Systeme*, Vieweg Verlag 2004

[Com01] CommSy Homepage, *http://www.commsy.de*

[Com02] CommSy BenutzerInnenhandbuch, *http://www.commsy.de/downloads/commsy_nutzungshandbuch.pdf*

[Com03] CommSy Moderationshandbuch, *http://www.commsy.de/downloads/commsy_moderationshandbuch.pdf*

[Dau01] Daum, B., *Rich-Client-Entwicklung mit Eclipse 3.1*, dpunkt.verlag 2005

[Dau02] Daum, B., *Das Eclipse-Codebuch*, dpunkt.verlag 2006

[ECF01] Eclipse Communication Framework Project Homepage, *http://www.eclipse.org/ecf*

[ECF02] ECF Documentation, *http://www.eclipse.org/ecf/documentation.php*

[ECF03] eclipseCon 2006 ECF Tutorial, *http://www.eclipecon.org/2006/Sub.do?id=46*

[ECF04] Getting started with the Eclipse Communication Framework, *http://www-128.ibm.com/developerworks/opensource/library/os-ecl-commfwk/*

[LAssi01] Homepage des Projektes „Reinventing Education - Werkzeuge für das Lernen“, *http.//www.lassitools.org*

[Sch01] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J., *Design Patterns*, Addison-Wesley, 1995

[SFT01] Schoder, D. ; Fischbach, K.; Teichmann, R., *Peer-to-Peer,* Springer Verlag, 2002

[SOAP01] Soap Version 1.2 Spezifikation, *http://www.w3.org/TR/soap12*

[Tay01] Taylor, I. J., *From P2P to Web Services and Grids*, Springer Verlag, 2005