FernUniversität in Hagen

Department of Mathematics
and Computer Science

Integration Dualer Interaktionsräume

Die Verknüpfung von textbasierter synchroner Kommunikation mit diskreten Konstruktionswerkzeugen

Martin Mühlpfordt 2009

Research Report 1/2009
ISSN 1865-3944

Editor:	Dean of the Department of Mathematics and Computer Science
Type and Print:	FernUniversität in Hagen
Distribution:	http://deposit.fernuni-hagen.de/view/departments/miresearchreports.html

Integration Dualer Interaktionsräume

Die Verknüpfung von textbasierter synchroner Kommunikation mit diskreten Konstruktionswerkzeugen

Dissertation
zur Erlangung des Grades eines
Doktor-Ingenieurs
der Fakultät für Mathematik und Informatik der
FernUniversität in Hagen

vorgelegt von
Dipl. Inform. Dipl-Psych.
Martin Mühlpfordt

Dezember, 2008

Danksagung

Bei der Erstellung dieser Arbeit haben mich viele Menschen unterstützt. Ihnen gilt mein herzlicher Dank.

An erster Stelle danke ich Prof. Dr.-Ing. Jörg M. Haake für die immer zielorientierte und sehr konstruktive Betreuung meiner Arbeit. Er schärfte meinen Blick für das Wesentliche.

Besonders möchte ich mich bei meinen Kollegen Prof. Hans-Rüdiger Pfister und Dr. Martin Wessner am Fraunhofer-IPSI und insbesondere den Mitstreitern bei den legendären *Writers Workshops* bedanken – Dr. Andrea Kienle, Friederike Jödick, Axel Guicking und Torsten Holmer. Ihr regelmäßiges (durchaus schonungsloses aber immer motivierendes) Urteilen und Antreiben waren die steten Tropfen ...

Wichtige Impulse erhielt ich während meines Forschungsaufenthalts an der Drexel University, Philadelphia. Dem Team um Prof. Gerry Stahl, insbesondere Nan Zhou, Murat Çakir, Johann Sarmiento, Ramon Toledo und Dr. Alan Zemel verdanke ich viele Anregungen und Erkenntnisse.

Ohne die Unterstützung meiner Familie wäre diese Arbeit nicht entstanden. Ihr bin ich zu tiefstem Dank verpflichtet. Meine Frau Cornelia Mühlpfordt hielt mir oft genug den Rücken frei, der gestrenge Blick der Deutschlehrerin Renate Mühlpfordt verhinderte so manche sprachliche Ungereimtheit und erst die mehrwöchigen Klausuraufenthalte bei meinen Schwiegereltern Elisabeth und Hubert Cluse ließen die Arbeit Wirklichkeit werden.

Martin Mühlpfordt
Bielefeld, November 2008

Inhaltsverzeichnis

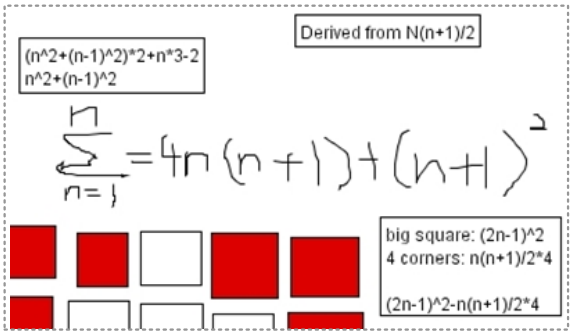
Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Kollaborationstranskripte

1 Einleitung: Duale Interaktionsräume und ihre Probleme

Aznx, bwang8 und Quicksilver – drei durch mehrere Hundert Kilometer voneinander getrennte Jugendliche – sitzen vor ihren miteinander vernetzten Computern und versuchen gemeinsam, ein mathematisches Problem zu lösen. Hierzu tauschen sie in einem Chat einfache Textnachrichten aus und skizzieren ihre Lösung mit Hilfe einer Software, die ihnen einen gemeinsamen virtuellen Arbeitsbereich zum Zeichnen bereitstellt. Im Laufe der Diskussion kommt es zu folgendem kurzen Wortwechsel zwischen Aznx und Quicksilver, wobei links ein Ausschnitt aus dem gemeinsamen Arbeitsbereich zu sehen ist:



Aznx: Suppose their second formula is our third.
Quicksilver: That was taem c's tho
Aznx: No.
Aznx: They didn't do.
Aznx: The nuumber of squares
Quicksilver: ohj!
Aznx: or the find the big square
Quicksilver: that formula
Quicksilver: i thot u meant the other one
Quicksilver: yeah that is ours

Nicht nur Unbeteiligten fällt es schwer zu verstehen, auf welche der links abgebildeten Formeln sich die Äußerungen von Aznx und Quicksilver beziehen. Auch von den Teilnehmern wird einige Anstrengung verlangt. Dies ist nur eine der Schwierigkeiten, die sich aus dem speziellen Lernarrangement ergibt: Die Lernenden befinden sich an verschiedenen Orten, sie kommunizieren über Computer und nutzen spezielle Werkzeuge für die Konstruktion der für die Aufgabenbearbeitung hilfreichen Darstellungen.

Diese Arbeit geht der Frage nach, welche Funktionen eine Software bereitstellen muss, um die Teilnehmer bei ihrer gemeinsamen Lernaktivität zu unterstützen, und wie diese Funktionen in einer Kollaborationsumgebung implementiert werden können.

1.1 Problemfeld: Kollaborationsumgebungen für netzbasiertes Lernen

Innerhalb der letzten Jahre wurden unter dem Oberbegriff *E-Learning* wieder Lehr-Lernmethoden populär, „die sich in der Vergangenheit nur schwer gegen die dominierenden monologischen Unterrichtsformen durchsetzen konnten“ (Schulmeister, 1999), insbesondere auch kooperative und kollaborative[1] Lernmethoden wie Rollenspiele, Teampräsentationen und Projektarbeiten. Ein Grund für die zunehmende Popularität ist, dass gerade solche Lernsituationen, in denen die Teilnehmer über vernetzte Computer miteinander agieren, die engagierte Wissenskonstruktion ermöglichen, denn hierbei sind der Zugang zu verschiedenen Lernressourcen sowie die gemeinsame, auf den Austausch untereinander gerichtete Aktivität wesentliche Bestandteile der Lernarrangements (Romiszowski & Mason, 2004, S. 400f).

In der Forschung hat sich für das Forschungsfeld, welches sich mit der Nutzung von Computern für das kooperative und kollaborative Lernen beschäftigt, der Begriff *Computer Supported Collaborative Learning* (CSCL) eingebürgert. Die Geburtsstunde von CSCL wird meist auf das Jahr 1989 datiert, denn in diesem Jahr fand in Maratea, Italien, mit einem Workshop die erste öffentliche und internationale Veranstaltung statt, die den Begriff CSCL in ihrem Titel führte (Stahl et al., 2006a).

Die Forschung zum Einsatz von Computern für das kollaborative Lernen ist so vielfältig wie die verwendeten didaktischen Ansätze, die technischen Möglichkeiten sowie die Anwendungsfelder. Das Forschungsfeld CSCL zeichnet sich deshalb durch ein hohes Maß an Interdisziplinarität aus, es verbindet Pädagogik mit Psychologie und Informatik. Aus der Blickrichtung der Pädagogik steht die Frage im Vordergrund, wie durch den Einsatz von CSCL-Systemen in geeigneten Lehr-Lernarrangements das kollaborative Lernen unterstützt werden kann. Die psychologische Forschung innerhalb der CSCL zielt eher auf die Erklärung, wie kollaborativ gelernt wird (Wessner, 2005).

Vordringliche Aufgabe der Informatik ist hierbei, im Sinne einer Gestaltungswissenschaft (Rolf, 1992) Softwaresysteme und -funktionalitäten zu entwickeln, die das kooperative und kollaborative Lernen ermöglichen und den Lernenden dabei helfen. So bietet die Informatik „hier sehr weitgehende Möglichkeiten, auf geeignete Weise Inhalte bereitzustellen, einen virtuellen elektronischen Lern- und Arbeitskontext zu schaffen und die Gruppenprozesse zu steuern und zu unterstützen“ (Haake et al., 2004).

Abhängig vom Funktionsumfang der CSCL-Softwaresysteme spricht man allgemein von computer*vermittelter* Kollaboration, wenn sie die zur Kollaboration

[1] Die Verwendung der Begriffe „kollaborativ“ und „kooperativ“ im Zusammenhang mit CSCL ist nicht einheitlich. In dieser Arbeit wird der erstere verwendet. Auf die Gründe hierfür und die Unterscheidung von kooperativem und kollaborativem Lernen wird in Kapitel 2.1.1 eingegangen.

Tabelle 1.1: Raum-Zeit-Matrix für CSCL mit jeweils einem Beispiel (nach Haake et al., 2004, S. 2)

	Ort	
Zeit	**Gleicher Ort**	**Verschiedene Orte (verteilt)**
Gleiche Zeit (synchron)	Synchron und am selben Ort z. B. Computerunterstütztes Klassenzimmer	Synchron und verteilt z. B. Chatsitzung
Verschiedene Zeit (asynchron)	Asynchron und am selben Ort z. B. Schwarzes Brett	Asynchron und verteilt z. B. Diskussionsforum

benötigten Medien bereitstellen und somit das gemeinsame Lernen erst ermöglichen, und von computer*unterstützter* Kollaboration, sofern das System darüber hinaus aktiv in das Kollaborationsgeschehen eingreift, indem es der Gruppe zum Beispiel Rückmeldung über den Kollaborationsverlauf gibt (Jermann, 2004) oder sie etwa mit Hilfe von Kooperationsskripten (Pfister et al., 2003) durch den Lernprozess leitet.

Die verschiedenen Formen computervermittelten kollaborativen Lernens können in Anlehnung an die Typologie von *Groupware*[2]-Systemen (Grudin, 1994) anhand der Raum-Zeit-Matrix klassifiziert werden (siehe Tab. 1.1). Zum einen können sich die Gruppenmitglieder wie in der klassischen Klassenzimmersituation am selben Ort oder wie in virtuellen Lernszenarien an verschiedenen Orten befinden. Zum anderen kann das Lernen synchron (die Mitglieder der Gruppe nehmen gleichzeitig daran teil) oder asynchron erfolgen (die Teilnehmer lernen zu unterschiedlichen Zeiten). Softwaresysteme für das computervermittelte bzw.-unterstützte kollaborative Lernen fokussieren üblicherweise auf eine Zelle der Matrix, so ist das primäre Anwendungsszenario für *CoolModes* (Hoppe et al., 2002) das kollaborative Lernen im Klassenzimmer, während zum Beispiel *Kolumbus* (Kienle, 2003) auf das asynchrone und räumlich verteilte Lernen zielt.

1.2 Ziel der Arbeit: Kollaborationsunterstützung für Duale Interaktionsräume

Diese Arbeit richtet ihr Augenmerk auf die Variante synchrones und örtlich verteiltes CSCL. In dieser befinden sich die miteinander zur selben Zeit kollaborierenden Teilnehmer an verschiedenen Orten, wie es im einführenden Beispiel auf Seite 1 der Fall ist.

In dieser Art Lernarrangement erfolgt jeglicher Austausch zwischen den Lernenden über eine Kollaborationsumgebung. Diese stellt üblicherweise virtuelle

[2] In dieser Arbeit wird auf die englischsprachlichen Termini zurückgegriffen, sofern sie sich auch in der deutschen Literatur durchgesetzt haben.

Kollaborationsräume bereit. Die Metapher des Raumes[3] dient der Organisation der Kollaboration. Im engeren Sinne impliziert sie eine „Kopräsenz“ der Gruppenmitglieder: es können nur diejenigen miteinander kommunizieren, die sich im selben virtuellen Raum befinden, und andersherum, Mitglieder außerhalb des Raumes können im Allgemeinen die Kommunikation nicht verfolgen und an ihr teilnehmen. Im weiteren Sinne bezieht sich die Raummetapher auch auf die Verdeutlichung der gemeinsam nutzbaren Ressourcen. Sehen Teilnehmer in dem Raum ein Dokument, so können sie davon ausgehen, dass auch die anderen Teilnehmer auf dieses Dokument Zugriff haben.

Da sich die Teilnehmer an verschiedenen Orten befinden, muss die Kollaborationsumgebung es den Teilnehmern ermöglichen, miteinander synchron zu kommunizieren. Hierfür können unterschiedlichste synchrone Kommunikationsmedien zum Einsatz kommen, wie z. B. Chat-, Audio- oder Videosysteme. Diese Arbeit konzentriert sich jedoch auf textbasierte, synchrone Kommunikationsmedien, d. h. die Kollaborationsumgebung beinhaltet einen Chat für den synchronen Austausch. Diese Fokussierung auf die textuelle synchrone Kommunikation erfolgt aus drei Gründen:

1. Die bei der textuellen Kommunikation notwendige Verschriftlichung der Äußerungen hat für sich genommen positive Effekte für das (kollaborative) Lernen (siehe Kap. 2.3).

2. Aufgrund ihrer vergleichsweise geringen technischen Anforderungen an die Leistungsfähigkeit der Arbeitsplatzrechner und der Netzwerkanbindung sind textuelle Kommunikationswerkzeuge insbesondere beim *E-Learning* weit verbreitet.

3. Desgleichen erlauben die geringen technischen Anforderungen an das synchrone, textuelle Medium eine leichte Anwendbarkeit der Ergebnisse dieser Arbeit.

Für viele Lernaufgaben ist es hilfreich, wenn die Teilnehmer nicht nur miteinander kommunizieren, sondern darüber hinaus auch digitale Artefakte wie Skizzen oder Mindmaps gemeinsam erzeugen und bearbeiten können. Hierfür existiert bereits eine Reihe von Konstruktionswerkzeugen, welche die kollaborative Erstellung und Manipulation solcher Artefakte in synchronen und örtlich verteilten Szenarien erlauben.

Abbildung 1.1 zeigt schematisch den Aufbau einer beispielhaften Kollaborationsumgebung, welche sowohl die Kommunikation wie auch die gemeinsame Konstruktion ermöglicht. Sie stellt hierfür den Teilnehmern zwei „Werkzeuge“ bereit: Zum einen ein textuelles Kommunikationsmedium und zum anderen ein Werkzeug zur gemeinsamen Bearbeitung der aufgabenrelevanten Artefakte. Den

[3] „Raum“ ist hier im Sinne eines Zimmers gemeint. Leider hat sich in diesem Zusammenhang in der deutschsprachigen Literatur die Übersetzung „Raum“ für das englische „room“ durchgesetzt, was zu Verwechslung mit der Bedeutung im Sinne des englischen „space“ führen kann.

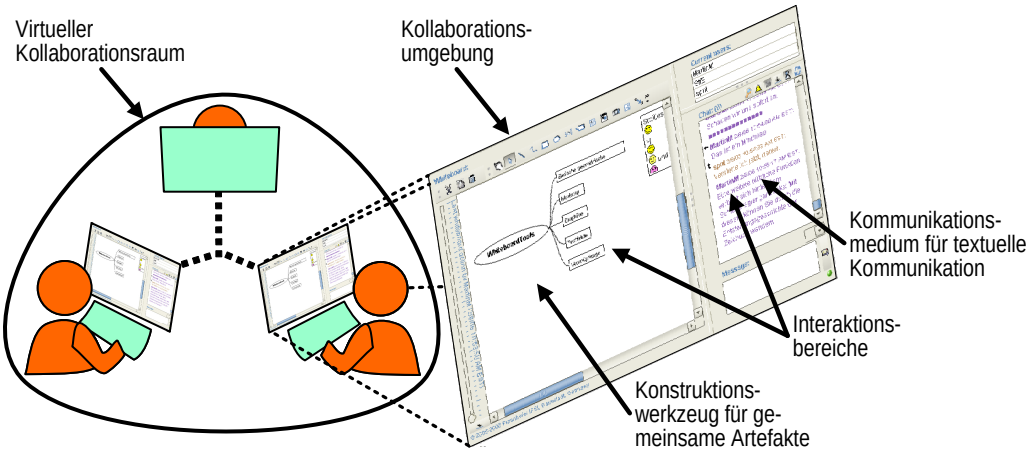


Abbildung 1.1: Virtueller Kollaborationsraum, Kollaborationsumgebung und Interaktionsbereiche

Teilnehmern stehen damit zwei Bereiche für die Interaktion miteinander zur Verfügung, die man nach Dillenbourg & CSCL SIG of Kaleidoscope (2005) als *Communication* bzw. *Discourse Space* für die sprachliche Auseinandersetzung und als *Task* oder besser *Artefact Space* für das Hantieren mit den aufgabenrelevanten Artefakten konzeptualisieren kann. Die Unterscheidung zwischen Diskurs- und Artefaktraum ist zwar ziemlich unscharf, da sowohl im sprachlichen Diskurs die Konzepte der Aufgabe behandelt werden als auch im Artefaktraum miteinander kommuniziert wird. Sie hat aber ihre pragmatische Berechtigung, da sich in virtuellen Lernumgebungen zur synchronen Kollaboration entsprechende separate Werkzeuge für die textuelle Kommunikation und die Artefakterstellung und -bearbeitung finden.

Solche Kollaborationsumgebungen, die zwei Bereiche zur Interaktion bereitstellen, werden im Folgenden „duale Interaktionsräume“ genannt. Im Kapitel 3 werden einige duale Interaktionsräume vorgestellt. Je nach Fokus des Systems bzw. des intendierten Einsatzszenarios ist der Aufgabenraum eine Ergänzung zum Diskursraum, z. B. zur Erstellung von Gesprächsnotizen. Oder aber der Diskursraum ist eine Ergänzung zum – dann meist sehr ausgefeilten – aufgabenspezifischen Artefaktraum, wie z. B. in Belvédère ein Werkzeug zur Erstellung von Konzeptgraphen (Suthers et al., 2003).

Das Nebeneinander der beiden Interaktionsbereiche bringt jedoch eine Reihe von Schwierigkeiten mit sich. Diese resultieren daraus, dass aus Sicht des Akteurs zu einem Zeitpunkt immer nur in einem der beiden Interaktionsbereiche (im textuellen Kommunikationsmedium oder im gemeinsamen Arbeitsbereich) agiert werden kann. Das gleichzeitige Deuten auf Objekte während des Sprechens oder das gleichzeitige Skizzieren einer Lösungsidee an einer Tafel während des Erklärens sind nicht möglich. Die Aktionen werden somit zeitlich und räumlich aufgeteilt: Die intentional zusammenhängenden Aktionen (Skizzieren im Artefaktraum

und Erklären im Diskursraum) müssen in eine Abfolge von Aktionen in den beiden Interaktionsbereichen „übersetzt“ werden.

Aus Sicht der Kollaborationspartner sind die Aktionen darüber hinaus „disembodied acts“ (Zemel et al., 2007): Das eigentliche Tun ist nicht beobachtbar, allein die Resultate (ein neuer Chatbeitrag oder eine Änderung an den Artefakten) werden sichtbar. Diese räumlich und zeitlich segmentierten Aktionen müssen identifiziert und in ihren Bedeutungszusammenhang „zurück übersetzt“ werden.

Die räumliche und zeitliche Segmentierung der Aktionen in dualen Interaktionsräumen führt damit unter anderem[4]

- zu einer Erhöhung der kognitiven Beanspruchung, denn die sich über die beiden Interaktionsbereiche erstreckenden Aktivitäten der anderen müssen mental integriert werden (van Bruggen et al., 2002);
- zu einem erhöhten Koordinationsaufwand, denn die Teilnehmer müssen ihre Aktivitäten in den beiden Bereichen miteinander abstimmen (Pata & Sarapuu, 2003);
- zu Schwierigkeiten in der Kommunikation, denn die Bezugnahme auf die gemeinsamen Artefakte ist im textuellen Kommunikationsmedium nur schwer zu realisieren (Suthers et al., 2003).

Ziel dieser Arbeit ist es nun, softwaretechnische Maßnahmen zu entwickeln, mit denen diesen Schwierigkeiten begegnet wird. Die zentrale Frage lautet: Wie muss das aus den beiden Interaktionsbereichen gebildete Kollaborationsmedium gestaltet sein, d. h. welcher zusätzlicher Funktionalitäten bedarf es, wenn zwei Interaktionsbereiche zu *einem* Medium für die Kollaboration kombiniert werden?

1.3 Interdisziplinärer Charakter der Arbeit

Wie bereits eingangs erwähnt, zeichnet sich das Forschungsfeld CSCL durch interdisziplinäre Fragestellungen aus. Auch für die Entwicklung von Maßnahmen zur Unterstützung der Kollaboration in dualen Interaktionsräumen müssen Forschungsergebnisse verschiedener Fachrichtungen einbezogen werden:

- Welche Basisaktivitäten des kollaborativen Lernens müssen in synchronen, örtlich verteilten Lernarrangements unterstützt werden?

  Zur Beantwortung dieser Frage werden Forschungsergebnisse der Pädagogik und der Pädagogischen Psychologie aufgegriffen.

- Wie lassen sich die Defizite, die beim Einsatz existierender Kollaborationssoftware beobachtbar sind, erklären?

[4] Auf diese und weitere Probleme dualer Interaktionsräume wird detailliert in Kapitel 2.5 eingegangen.

Hierzu werden Erkenntnisse der Kommunikationswissenschaft und Psychologie herangezogen, um Anforderungen an die Unterstützungsmaßnahmen abzuleiten.

- Wie können Unterstützungsmaßnahmen technisch umgesetzt werden?

  Kollaborationssysteme für das synchrone Lernen zeichnen sich gegenüber Einzelbenutzeranwendungen durch eine erhöhte Komplexität aus, da hier die vernetzten Computer möglichst effizient miteinander synchronisiert werden müssen. Die Informatik, insbesondere die Forschung zum computerunterstützten Arbeiten in Gruppen (CSCW), stellt hierfür eine Reihe von Lösungen bereit.

1.4 Lösungsidee: Integrationskonzept für Duale Interaktionsräume

Die zentrale Idee dieser Arbeit ist es, die zwei bislang unverbundenen Interaktionsbereiche Kommunikations- und Konstruktionswerkzeug softwaretechnisch dahingehend zu integrieren, dass sie als *ein* Medium nutzbar sind: die über die beiden Interaktionsbereiche sich verteilenden, jedoch aufeinander bezogenen (Inter-) Aktionen sollen für die Teilnehmer leichter ausdrückbar, koordinierbar, erfassbar und nachvollziehbar werden.

Hierzu werden Integrationsmaßnahmen entwickelt, die auf drei Problemfelder dualer Interaktionsräume abzielen:

Deiktische Bezugnahme: Wie kann die Bezugnahme vom textuellen Kommunikationsmedium auf den gemeinsamen Arbeitsbereich unterstützt werden (wie z. B. „that formula“ auf Seite 1 – im direkten Gespräch hätte Quicksilver auf die entsprechende Formel gezeigt)?

Abhilfe schafft die Möglichkeit zur expliziten Referenzierung, mit der Chatbeiträge visuell mit Objekten im Arbeitsbereich verknüpft werden können.

Koordination: Wie können die Teilnehmer ihre Aktionen im gemeinsamen Arbeitsbereich und ihre Beiträge im Chat miteinander koordinieren? Wie kann z. B. Aznx erkennen, dass Quicksilver gerade eine Idee im unter Umständen nicht sichtbaren Ausschnitt des Arbeitsbereichs skizziert?

Zur Unterstützung bei der Koordination werden *Awareness*-Anzeigen entwickelt, die über die Grenzen der Interaktionsbereiche hinweg über die momentanen Aktionen der Teilnehmer informieren.

Reflexion: Wie können die aufeinander bezogenen, jedoch auf die beiden Interaktionsbereiche verteilten Aktionen und Mitteilungen der Teilnehmer in ihren Beziehungen zueinander zur Reflexion über die Kollaboration nachvollzogen werden? Wie können in späteren Sitzungen zum Beispiel der Chatdis-

kurs zu einem bestimmten Konstruktionsschritt oder die temporär als Verdeutlichung erstellte Skizze zu einer Erklärung im Chat identifiziert werden?

Hierzu werden Maßnahmen aufgezeigt, mit der die Aktionskontexte im jeweils anderen Interaktionsbereich rekonstruiert werden können.

Um dies zu erreichen werden aus informatischer Sicht im Wesentlichen zwei interaktionsbereichsübergreifende Relationen eingeführt:

Relationen zwischen Objekten: Indem einzelne Chatnachrichten und die je nach Konstruktionswerkzeug gegebenen Artefakte als Objekte aufgefasst werden, können deiktische Bezugnahmen über Relationen zwischen diesen Objekten beschrieben werden.

Aktionen: Indem die Interaktionsbereiche als diskrete Medien aufgefasst werden, in welchen der aktuelle Zustand das Ergebnis diskreter (zumeist nutzerinitiierter) Aktionen ist, können zum einen die Kontexte zu einzelnen Aktionen rekonstruiert und zum anderen die Aktionen auch in den jeweils anderen Interaktionsbereichen zum Beispiel für *Awareness*-Darstellungen ausgewertet werden.

1.5 Vorgehen und Aufbau

Die Unterstützung der Kollaboration in dualen Interaktionsräumen ist ein bisher wenig behandeltes Forschungsthema. In Kapitel 2 wird deshalb die Problemstellung analysiert. Anhand von 1. Theorien zum kollaborativen Lernen und zur medial vermittelten Kommunikation und 2. Erkenntnissen aus dem Forschungsbereich computergestütztes kooperatives Arbeiten (CSCW) über gemeinsame Arbeitsbereiche sowie 3. empirisch gefundener Defizite dualer unverbundener Interaktionsräume werden Anforderungen an Maßnahmen zur Unterstützung der Kollaboration in dualen Interaktionsräumen abgeleitet.

In Kapitel 3 wird der Stand der Technik bezüglich der Integration von Diskurs- und Aufgabenbereich erörtert. Hierzu werden Ansätze und Systeme vorgestellt, die Alternativen zum unverbundenen Nebeneinander der beiden Interaktionsbereiche umsetzen, und es wird geprüft, inwieweit sie den Anforderungen aus Kapitel 2 gerecht werden.

Kapitel 4 stellt die Lösungskonzepte für die Integration dualer Interaktionsräume dar. Dies umfasst die softwaretechnische Konzeption eines Integrationssystems, welches einerseits den Anforderungen genügt und mit dem andererseits nahezu beliebige Chat- und Konstruktionswerkzeuge verknüpft werden können. Hierzu werden die Anforderungen in sechs Funktionsbereiche eingeteilt:

1. Sicherstellen der Konsistenz der Aktionsabfolgen,
2. Kombination der Interaktionshistorien beider Interaktionsbereiche,
3. Mechanismen zur Kontextrekonstruktion,

4. bereichsübergreifende Dokumentation der Aktionen,
5. deiktische Bezugnahme und
6. integrierte Anzeige von *Awareness*-Informationen.

Für jeden dieser Funktionsbereiche wird gezeigt, wie das Kommunikations- mit dem Konstruktionswerkzeug verknüpft werden kann.

Dieses konzipierte Integrationssystem wurde beispielhaft in dem System *ConcertChat* umgesetzt. In Kapitel 5 wird das System vorgestellt und auf einzelne Aspekte der Realisierung eingegangen.

Diese Arbeit zielt auf die Unterstützung der Kollaboration in dualen Interaktionsräumen. Kapitel 6 zeigt, auf welche Art und Weise die entwickelten Integrationsmaßnahmen von den Lernenden genutzt werden. Hierzu werden zum einen einzelne Kollaborationssequenzen analysiert und zum anderen die Ergebnisse von Interviews mit erfahrenen Nutzern des Systems dargestellt. Des Weiteren wird geprüft, inwieweit die softwaretechnische Konzeption auf verschiedene Kollaborationswerkzeuge anwendbar ist.

Abschließend fasst Kapitel 7 die Ergebnisse der Arbeit zusammen und zeigt die sich aus ihr ergebenden neuen Fragestellungen auf.

2 Problemanalyse: Anforderungen an duale Interaktionsräume

Diese Arbeit fokussiert auf eine spezielle Art von Lernumgebungen für das synchrone, örtlich verteilte kollaborative Lernen: auf duale Interaktionsräume. In diesen gibt es neben einem textuellen Kommunikationskanal (Chat) eine gemeinsame Arbeitsfläche zur kollaborativen Bearbeitung der für die Aufgabe benötigten digitalen Artefakte (z. B. geometrische Zeichnungen oder Skizzen). Die Erfahrungen mit dem kollaborativen Lernen in dualen Interaktionsräumen zeigen jedoch eine Reihe von Schwierigkeiten, die sich aus dem unverbundenen Nebeneinander der beiden Interaktionsbereiche ergeben. Ziel dieser Arbeit ist die Konzeption softwaretechnischer Integrationsmaßnahmen für duale Interaktionräume.

In diesem Kapitel werden hierfür die Anforderungen analysiert: Basierend auf den in Abschnitt 2.1 aufgeführten drei Basiselementen kollaborativen Lernens – dem Diskurs, der kollaborativen Konstruktion gemeinsamer Artefakte und der Reflexion – werden in den darauf folgenden Abschnitten die Merkmale und Besonderheiten der computervermittelten textuellen Kommunikation mit einem Chatwerkzeug (Abschnitt 2.3) sowie der computervermittelten kollaborativen Konstruktion mit einem Konstruktionswerkzeug (Abschnitt 2.4) dargestellt. Anschließend werden in Abschnitt 2.5 die Anforderungen an Maßnahmen zur Integration beider Werkzeuge abgeleitet.

2.1 Lernen in Gruppen

In diesem Kapitel sollen die Grundlagen des Lernens in Gruppen beleuchtet werden. Im Gegensatz zum individuellen Lernen wie auch zur traditionellen monologischen Lehre erfordert das Lernen in Gruppen die Interaktion mit Mitlernern. In der Auseinandersetzung mit den Überlegungen und Konzepten anderer werden gemeinsam Bedeutungen und Wissen konstruiert. Auf diese gemeinsame Wissenskonstruktion zielen kollaborative Lernmethoden. Nachdem im folgenden Abschnitt auf den Begriff des *kollaborativen* Lernens in Abgrenzung zum *kooperativen* Lernen eingegangen wurde, wird anschließend der Frage nachgegangen, warum das Lernen in Gruppen lernförderlich ist. Hierzu werden die wichtigsten theoretischen Perspektiven kurz vorgestellt und anschließend die drei Basiselemente kollaborativen Lernens beleuchtet: der sprachliche Diskurs, die gemeinsame Konstruktion von aufgabenrelevanten Artefakten und die Reflexion über den Lern- und Kollaborationsprozess.

2.1.1 Begriffsbestimmung

Im weitesten Sinne kann unter kooperativem Lernen jegliche Interaktion zwischen zwei oder mehr Teilnehmern mit dem gemeinsamen Ziel des Lernens verstanden werden (Dillenbourg, 1999). Unter diese Definition fallen alle Lehr-Lernarrangements vom klassischen Frontalunterricht bis hin zu Gruppenprojekten unter Nutzung virtueller Lernumgebungen.

In den letzten Jahren gab es vor allem in der angelsächsischen Literatur eine Debatte um die Abgrenzung zwischen *cooperative learning* und *collaborative learning* als jeweils besonderen Formen des Lernens in Gruppen (vgl. Stahl et al., 2006a). Ein Ergebnis dieser Debatte ist, dass zwischen kooperativem und kollaborativem Lernen aufgrund des Ausmaßes an Arbeitsteilung innerhalb der Lerngruppe unterschieden wird (Dillenbourg, 1999). Während bei der *Kooperation* die Lernenden die zu erledigende Arbeit in Teilaufgaben jeweils individuell lösen und anschließend zu einem gemeinsamen Ergebnis integrieren, arbeiten die Lernenden bei der *Kollaboration* meist gemeinsam. Die Unterscheidung zwischen Kooperation und Kollaboration ist aufgrund des graduellen Kriteriums nicht dichotom. Das Ausmaß der Arbeitsteilung spannt eine Dimension zwischen vollständig arbeitsteilig und nicht arbeitsteilig auf.

Hintergrund der Debatte war jedoch weniger die Suche nach einer guten Definition als vielmehr eine Auseinandersetzung über die wesentlichen Aspekte des Lernens in Gruppen. Vertreter des kollaborativen Lernens gehen davon aus, dass durch die Interaktion in der Gruppe die Lernenden gemeinsam Wissen und Bedeutungen konstruieren, und zielen in ihren Arbeiten darauf, wie diese Ko-Konstruktion von Wissen erfolgt und gefördert werden kann (Thalemann, 2004). Demgegenüber betonen die Vertreter des kooperativen Lernens eher die individuellen Lernmotivationen und -erfolge und versuchen diese durch Belohnungs- und Wettbewerbsmechanismen in der Gruppe zu steigern (Slavin, 1997).

Der Begriff der „Kollaboration“ setzt sich aufgrund seiner negativen Konnotation in der deutschsprachigen Literatur nur zögerlich für die Lernmethoden zur gemeinsamen Wissenskonstruktion durch; zumeist wird „kooperatives Lernen“ als Oberbegriff für die verschiedenen Formen des Lernens in Gruppen verwendet. Diese Arbeit fokussiert auf Lernszenarien, in denen die gemeinsame Wissenskonstruktion im Sinne des kollaborativen Lernens im Vordergrund steht. Zur Betonung dieser Art des Lernens in Gruppen wird deshalb im Weiteren der Begriff des „kollaborativen“ Lernens verwendet.

2.1.2 Theoretische Erklärungsperspektiven

So breit gefächert wie die unterschiedlichen Methoden für das kollaborative Lernen in Gruppen sind auch die jeweiligen theoretischen Ansätze zur pädagogischen Gestaltung des Lernens wie auch der Erklärung der Effekte. Diese Ansätze lassen sich hinsichtlich ihrer theoretischen Perspektiven klassifizieren (Slavin, 1996; Fischer et al., 2000). Die wichtigsten Perspektiven, die das Wechselspiel zwischen der Interaktion mit anderen und den individuellen kognitiven Prozessen zu erklären versuchen, sind die soziogenetische Perspektive, die Perspektive der kognitiven Elaboration, die sozio-kulturelle Perspektive sowie die Perspektive der kollektiven Informationsverarbeitung.

Die *soziogenetische Perspektive* geht auf die Arbeiten von Piaget zurück. Grundannahme dieses konstruktivistischen Ansatzes ist, dass der Mensch prinzipiell fähig ist, kognitive Systeme zu konstruieren, mit denen die Erfahrungen mit Personen und Objekten der Umwelt interpretiert werden können (Fischer et al., 2000). Diese Konstruktion wird durch soziale Interaktionen gefördert. In sozialen Interaktionen können kognitive Konflikte hervorgerufen werden, welche zur Auflösung inadäquater und zur Bildung neuer kognitiver Konzepte führen.

Basierend auf kognitiven Modellen der Informationsverarbeitung wird aus Perspektive der *kognitiven Elaboration* (Cohen, 1994) davon ausgegangen, dass der Austausch mit anderen beim Lernen die individuellen kognitiven Prozesse der Wissensbildung anregen und unterstützen. Das individuelle Wissen wird verändert, indem neue Information in die Vorwissensstruktur integriert wird. Durch die Interaktion mit anderen wird die Elaboration neuer Information gefördert, so dass diese eher und besser in die vorhandenen individuellen Wissensstrukturen eingefügt wird. Die Elaboration des zu Lernenden erfolgt sehr effektiv, wenn das zu Lernende einem anderen erklärt wird. Dieser Befund ist Grundlage des Konzepts vom *Reciprocal Teaching* (Palincsar & Brown, 1984; O'Donnell & Dansereau, 1992), bei dem Lernende abwechselnd die Rolle des Erklärenden bzw. des Nachfragenden übernehmen.

Auf den Arbeiten der von den sowjetischen Psychologen Wygotsky[1], Lurija und Leontjew begründeten kulturhistorischen Schule aufbauend wird aus der *sozio-*

[1] Die Umschrift des weißrussischen Namens Выгоцкі ist uneinheitlich. In deutschsprachiger Literatur findet sich „Wygotsky“ wie auch „Wygotski“, im englischen Sprachraum hat sich „Vygotsky“ durchgesetzt.

kulturellen und situierten Perspektive die sozio-kulturelle Einbettung der Entwicklung individueller Kognitionen betont. Im Fokus der Theoriebildung liegt die aktive Tätigkeit des Menschen. In der Tätigkeit eignet sich der Mensch spezifische Erfahrungen vermittels Werkzeugen an. Werkzeuge können hierbei wie Hammer und Stift materieller oder wie Zeichen und Symbole psychologischer Natur sein (Barab et al., 2004). Beim gemeinsamen Lernen geht die Gruppe einer gemeinsamen Handlung bzw. Tätigkeit nach. Eine Handlung wird als zweiseitig gerichteter Prozess verstanden. Durch die Handlung wird in der Umwelt ein Gegenstand aufgebaut, verändert oder manipuliert und gleichzeitig im Individuum der Begriff, das Wissen und das Verständnis über den Gegenstand aufgebaut und modifiziert (Oerter & Montada, 1987, S. 112). „Individuelle Kognitionen – also auch Wissen – und soziale Prozesse sind bei Wygotsky über das so genannte genetische Entwicklungsgesetz verknüpft, wonach höhere psychische Funktionen zunächst auf der interindividuellen Ebene – etwa als Argumentationsfigur in einer Diskussion – auftreten und erst dann auf der intraindividuellen Ebene. Diskursprozesse und -strukturen werden in einem komplexen Zusammenspiel von Externalisierung und Internalisierung zu einem Element der intraindividuellen Regulationsprozesse." (Fischer, 2001, S. 12).

Eine größere Betonung der Prozesse auf Gruppenebene zeichnet die Ansätze der Perspektive der *kollektiven Informationsverarbeitung* aus. Die Informationsverarbeitung auf Ebene der Gruppen wird dabei definiert „as the degree to which information, ideas, or cognitive processes are shared, and are being shared, among the group members and how this sharing of information affects both individual- and group-level outcomes" (Hinsz et al., 1997, S. 53). Annahme ist hierbei, dass die Gruppe als informationsverarbeitendes System aufzufassen ist, welches Merkmale aufweist, die bei den beteiligten Individuen allein nicht zu finden sind. Im Fokus stehen deshalb Fragen wie etwa, ob die Teilnehmer gleichermaßen von einer Kooperation profitieren, ob und wie individuelles Wissen eingebracht wird und ob dieses tatsächlich zu gemeinsamem Wissen wird (Fischer et al., 2000).

2.1.3 Die Basiselemente: Diskurs, Konstruktion und Reflexion

Im Folgenden werden die drei Grundbausteine kollaborativen Lernens vorgestellt: Der *Diskurs*, also die sprachliche Auseinandersetzung der Teilnehmer miteinander, die *Konstruktion gemeinsamer Artefakte*, also die gemeinsame Erstellung von Artefakten zur Repräsentation von Problemstellung, Aufgabenaspekten und Lösungen, sowie die *Reflexion*, also die Betrachtung und Bewertung des Verhaltens und Verstehens im Lernverlauf.

Die Rolle des Diskurses

Die skizzierten kognitiv orientierten Erklärungsansätze postulieren einen engen Zusammenhang zwischen dem Diskurs der Kooperierenden und den individuellen kognitiven Prozessen, wobei während und durch den Diskurs gemeinsames

Wissen konstruiert wird. Je nach theoretischer Perspektive werden unterschiedliche Aspekte des Diskurses betont: Das Aufeinandertreffen unterschiedlicher Auffassungen der Lernpartner, die erhöhte Elaboration des Lerngegenstandes, das Übernehmen der Perspektiven der anderen, die Hilfe durch kompetentere Lernpartner, das sich ergänzende Wissen. Ausschlaggebend für diese Arbeit ist die allen Perspektiven gemeinsame Betonung der Kommunikation als integraler Bestandteil kooperativen Lernens.

Die sprachliche Umsetzung erfordert, die eigenen Gedanken klar und kohärent zu strukturieren. Dabei können Verständnisprobleme und Wissenslücken leichter entdeckt werden (Reimann & Zumbach, 2001). Für eine stabile und ernsthafte Gruppenarbeit ist es jedoch notwendig, dass die Kommunikation nicht wie in den meisten Lernsituationen auf eine pure Weitergabe von Wissen von einem Lehrenden zu den Lernenden zielt, sondern transformativ ist, indem alle Beteiligten bereit zur Änderung ihrer Überzeugungen und Annahmen sind und im Laufe der Kommunikation ihr jeweiliges Wissen erweitern: „Each participant potentially provides creative resources for transforming existing practice.“ (Pea, 1994, S. 288).

Die Rolle der Konstruktion gemeinsamer Artefakte

Neben dem Diskurs ist ein weiteres bedeutsames Element kollaborativer Lehr-Lernarrangements die gemeinsame Konstruktion von Artefakten. Je nach Inhalt und Lernmethode unterscheiden sich die konstruierten Artefakte: Es können zum Beispiel Zusammenfassungen und Mindmaps von zu lernenden Textinhalten (O'Donnell & Dansereau, 1992), Konzeptgraphen zur Darstellung von Argumentationen komplexer Sachverhalte (Suthers, 1999b), strukturierte Sammlungen von Ideen (Haake et al., 2003) oder auch Veranschaulichungen geometrischer Sachverhalte (Stahl, 2006) sein.

Das Konstruieren bietet den Lernenden die Gelegenheit, für einen Bereich ein vertieftes Verständnis zu entwickeln, und einen Kontext, innerhalb dessen sie ihr Verständnis den Mitlernenden oder auch Lehrenden demonstrieren können (Reimann & Zumbach, 2001). Die Artefakte erfüllen beim kollaborativen Lernen mehrere Funktionen.

Erstens helfen die Artefakte als Teil der gemeinsamen Umgebung, die Kommunikation zu vereinfachen und Äußerungen zu desambiguieren. Als konversationale Requisite (Brinck & Gomez, 1992) werden sie dazu herangezogen, sprachlich nur schwer zu fassende Sachverhalte zu verdeutlichen und die in der Kommunikation bezeichneten Gegenstände zu identifizieren. So fanden Bekker et al. (1995) in einer Untersuchung, dass während der gemeinsamen Bearbeitung einer Designaufgabe in einer *face-to-face*-Situation die Zeigegeste (üblicherweise mit einem Finger) zur Identifikation einer anderen Person, eines Gegenstands oder Ortes am häufigsten verwendet wird im Vergleich zu Bewegungsgesten (also der gestischen Darstellung eines zu erklärenden Bewegungsablaufes) bzw. dem gestischen Ausdrücken räumlicher Sachverhalte.

Zweitens dienen die in der Gruppe erstellten Artefakte als gemeinsam genutzte externe Repräsentationen. Externe Repräsentationen sind nach Zhang (1997, S. 180) ganz allgemein „the knowledge and structure in the environments, as physical symbols, objects, or dimensions (e.g. written symbols, beads of abacuses, and dimensions of a graph) and as external rules, constrains, or relations embedded in physical configurations (e.g. spatial relations of written digits, visual and spatial layouts of diagrams, and physical constraints in abacuses)“. Sie dienen als Gedächtnisstütze und bieten Informationen, die direkt wahrnehmbar sind und nicht interpretiert und formuliert werden müssen; sie verankern und strukturieren das kognitive Verhalten, denn Repräsentationen mit eingebauten Regeln beschränken die möglichen Aktionen; sie erleichtern die Aufgabenbearbeitung, da kognitiv weniger Regeln verarbeitet werden müssen (Zhang & Norman, 1994). Die perzeptuelle Salienz von Merkmalen, Relationen und Regeln in der Repräsentation kann die kognitive Verarbeitung beeinflussen[2] (Zhang, 1997).

Die Konstruktion externer Repräsentationen kann mithilfe mehr oder weniger formaler Notationen erfolgen. So werden Mengenbeziehungen durch Venn-Diagramme, Argumentationen in Konzeptgraphen und Prozesse mittels Ablaufplänen veranschaulicht. Die Notationen können in einem Repräsentationswerkzeug implementiert sein, mit dem wiederum Repräsentationen als Artefakte erzeugt werden (Suthers, 1999a). Repräsentationen unterscheiden sich unter anderem hinsichtlich der Modalität der Darstellung (z. B. Text, Animation, Graphik), der Präzision, mit der die Elemente, Strukturen und Regeln des Gegenstandsbereichs repräsentiert sind, und der Komplexität (De Jong et al., 1998; van Bruggen et al., 2002).

Aus tätigkeitspsychologischer Perspektive haben die Artefakte somit sowohl eine Zeichen- wie auch eine Werkzeugfunktion. Die Zeichenfunktion ergibt sich aus der Verwendung als externe Repräsentationen, indem die Repräsentationen beeinflussen, *wie* die Kollaborierenden denken. Die Artefakte dienen als Werkzeug, indem sie die gemeinsam ausgeführten Aktivitäten der Gruppe organisieren und koordinieren (Alterman et al., 2001, S. 7).

Die Rolle der Reflexion

Sowohl konstruktivistische als auch kognitivistische Lehr-Lernansätze betonen das methodische Element beim Lernen: Für den nachhaltigen Lernerfolg ist der Aufbau von Metakompetenzen („*Wie* kann ich lernen?“) wichtig (Kerres & de Witt, 2004). Diese strategischen Kompetenzen können durch Selbstbeobachtung, Beobachtung anderer und Reflexion über den Lernprozess aufgebaut werden (Palincsar & Brown, 1984). Unter Reflexion versteht man metakognitive Prozesse zur Analyse des eigenen Verhaltens und des eigenen Verständnisses eines Sachverhalts (Reimann & Zumbach, 2001). Eine Lerngruppe denkt zum Beispiel nach der Lösung

[2] Der Einfluss auf die kognitive Verarbeitung ist jedoch nicht immer positiv. Durch ungeeignete Eigenschaften der Repräsentation kann die Problemlösung auch erschwert werden.

eines mathematischen Problems darüber nach, wie sie die Lösung unter Ausnutzung welcher Gesetzmäßigkeiten gefunden und ob jeder den Weg verstanden hat oder ob alternative Ansätze möglich wären.

Diese Prozesse der Abstraktion, Perspektivübernahme und Reflexion können den Lernprozess gegenüber dem Lernen durch Versuch und Irrtum beschleunigen und ihn qualitativ verändern, indem das Lernen weniger performanzorientiert (auf das reine Aufgabenerledigen ausgerichtet) und mehr verständnisorientiert wird (Ching, 1999).

Die Reflexion kann während der Kollaboration in reflektiven Interaktionen erfolgen, die Erklärungen, Begründungen und Bewertungen beinhalten (Baker & Lund, 1997). Sie kann aber auch jeweils individuell von den Teilnehmern im Nachhinein bzw. zwischen den Kollaborationen erfolgen, um das Gelernte aufzubereiten oder die nächste Lernsitzung vorzubereiten (Pfister et al., 2004).

2.1.4 Fazit

Das kollaborative Lernen zielt auf die gemeinsame Konstruktion von Wissen und Bedeutungen. In den gemeinsam ausgeführten Aktivitäten – dem gemeinsamen Diskurs, der Konstruktion aufgabenspezifischer Repräsentationen oder dem Hantieren mit aufgabenrelevanten Artefakten sowie reflektiven Interaktionen – verhandelt, erweitert und teilt die Gruppe die für sie relevanten Bedeutungen: angefangen vom gemeinsamen Verständnis der Aufgabe, über die Art und Weise, wie die Aufgabe zu bewältigen ist bis hin zur Erreichung der für die Gruppe relevanten Ziele.

Im nächsten Abschnitt wird nun darauf eingegangen, wie das kollaborative Lernen in Gruppen durch den Einsatz von Computern unterstützt werden kann.

2.2 Minimalanforderungen an Kollaborationsumgebungen für synchrones, örtlich verteiltes Lernen

Wie in Abschnitt 2.1.3 ausgeführt sind die Basiselemente kollaborativen Lernens der sprachliche Diskurs, die gemeinsame Konstruktion von Artefakten und die Reflexion über den Kollaborationsverlauf. Soll das kollaborative Lernen synchron in örtlich verteilten Lernszenarien erfolgen, bedarf es einer Kollaborationsumgebung, die diese Grundelemente berücksichtigt. Zum einen muss es eine Möglichkeit für den sprachlichen Diskurs geben und die Kollaborationsumgebung somit folgende allgemeine Minimalanforderung erfüllen:

A 1. Die Kollaborationsumgebung muss ein Kommunikationsmedium für den sprachlichen Diskurs bereitstellen.

Des Weiteren bedarf es für die gemeinsame Konstruktion aufgabenspezifischer Artefakte auch Werkzeuge, mit denen diese Artefakte betrachtet und kollaborativ bearbeitet werden können, so dass als allgemeine Minimalanforderung an die Lernumgebung gilt:

A 2. Die Kollaborationsumgebung muss Werkzeuge zur Anzeige und kollaborativen Bearbeitung aufgabenspezifischer Artefakte bereitstellen.

Und als drittes muss die Lernumgebung reflektive Interaktionen sowie die Reflexion im Sinne einer individuellen oder kollektiven Analyse des Kollaborationsverlaufs unterstützen:

A 3. Die Kollaborationsumgebung muss die Reflexion in den Interaktionen und als Analyse des Kollaborationsverlaufs unterstützen.

Die ersten beiden Minimalanforderungen entsprechen den unterschiedlichen Funktionen der Interaktionsbereiche dualer Interaktionsräume (siehe Kap. 1.2 und insbesondere Abb. 1.1), also dem *Discourse Space* und dem *Artefact Space*.

Ziel dieser Arbeit ist die Entwicklung softwaretechnischer Maßnahmen, mit denen die Kollaboration in solchen dualen Interaktionsräumen unterstützt wird. Hierzu bedarf es einer genaueren Analyse, wie Gruppen über textuelle Medien kommunizieren und computervermittelt gemeinsam die aufgabenrelevanten Artefakte konstruieren. Diese Analyse erfolgt in den nächsten beiden Abschnitten: In Kapitel 2.3 werden die Merkmale und Besonderheiten der computervermittelten (textuellen) Kommunikation vorgestellt und Minimalanforderungen an Kommunikationswerkzeuge abgeleitet; in Kapitel 2.4 werden analog die Mechanismen und Besonderheiten des computervermittelten Konstruierens in Gruppen vorgestellt und entsprechend Minimalanforderungen an das Konstruktionswerkzeug abgeleitet.

Basierend auf den Analysen des computervermittelten Kommunizierens und Konstruierens erfolgt in Kapitel 2.5 die Analyse der Probleme dualer Interaktionsräume: Woraus ergeben sich die Schwierigkeiten für das kollaborative Lernen, wenn Kommunikations- und Konstruktionswerkzeug unverbunden nebeneinander stehen? Und welche Anforderungen lassen sich daraus an eine Integration ableiten?

2.3 Computervermitteltes Kommunizieren per Chat

Im Folgenden werden die Anforderungen an das Kommunikationswerkzeug analysiert: Welche Eigenschaften muss ein Kommunikationsmedium zur synchronen textuellen Kommunikation aufweisen? Hierbei kann auf die ausgiebige Forschung zur Chatkommunikation zurückgegriffen werden. Zu Beginn soll jedoch grundsätzlich auf Kommunikation eingegangen werden: Was ist Kommunikation, wie

kommunizieren Menschen, wie verstehen sie das Gesagte und wie stellen sie sicher, dass ihre Äußerungen auch verstanden werden?

2.3.1 Mechanismen medial vermittelter Kommunikation

Kommunikation ist ein in vielfältigen Disziplinen wie auch in der Alltagssprache verwendeter Begriff und somit mit einer Reihe von Konnotationen und unterschiedlichen Bedeutungen versehen. Im engeren Sinne versteht man unter Kommunikation einen Vorgang der Verständigung und Bedeutungsvermittlung. „Kommunikation *zwischen Menschen* schließlich stellt – soziologisch betrachtet – eine Form sozialen Handelns dar, das mit subjektivem Sinn verbunden sowie auf das Denken, Fühlen und Handeln anderer Menschen bezogen ist." (Pürer, 2003, S. 4, Hervorhebung im Original). Dieses koordinierte Handeln zwischen den Kommunizierenden ist darauf ausgerichtet, ein gegenseitiges Verstehen zu erreichen und aufrechtzuerhalten. Es zeigt sich in der Sprache und anderen Formen symbolischer Interaktion (Ngwenyama & Lyytinen, 1997).

Eine für die Erklärung der Kommunikation wichtige Theorie ist die von Herbert H. Clark und Kollegen entwickelte psycho-linguistische Kommunikationstheorie, die *Grounding Theory* (siehe z. B. Clark, 1978, 1996)[3]. Die zentrale Frage der *Grounding Theory* ist, wie Menschen kommunizieren, wie sie das Gesagte verstehen und wie sie sicher stellen, dass ihre Äußerungen auch verstanden werden.

Betrachten wir den folgenden Wortwechsel zwischen Anna und Bernd:

Anna: „Aber nach dem vorhin Gesagten könnte es auch genau andersherum sein!"
Bernd: „Stimmt."
Anna: „Und wie machen wir es dann?"

Um die Äußerung von Anna – genauer gesagt, die intendierte Bedeutung der Äußerung – zu verstehen, bedarf es eines gemeinsamen Hintergrundwissens von Sprecher und Zuhörer. Anna geht offenbar davon aus, dass Bernd ein ähnliches Wissen über das Gespräch hat und verstehen wird, worauf sie mit „dem vorhin Gesagten" und „es" verweist. Dies trifft zu, denn Bernd antwortet mit einer akzeptierenden Äußerung. An dieser Stelle geht Anna davon aus, dass Bernd nun weiß, sie denke, aus dem vorher Gesagten könne auch das Gegenteil folgen. Und Bernd wird davon ausgehen, dass Anna dies auch weiß. Als Ergebnis wird die Bedeutung von Annas Äußerung Bestandteil des *Common Ground* von Anna und Bernd.

Das Verstehen ist somit ein aktiver Prozess der Kommunizierenden, durch den sie versuchen, ihre individuellen kognitiven Bezugsrahmen „so weit zur Deckung zu bringen, dass deren Schnittmenge – der sogenannte *common ground* – gerade ausreicht, um das jeweils spezifische Ziel der Kommunikation [...] zu erreichen"

[3] Einen guten Überblick über die *Grounding Theory* geben z. B. Monk (2003) und McCarthy & Monk (1994)

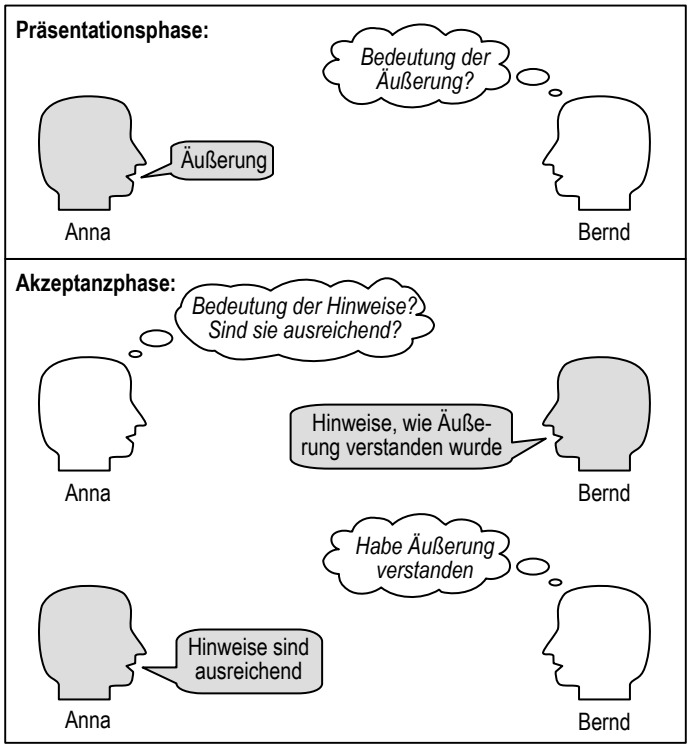


Abbildung 2.1: Schematische Darstellung des *Grounding*-Prozesses (nach Monk, 2003)

(Bromme et al., 2004, Hervorhebung im Original). Dieser Prozess wird *Grounding* genannt. Er besteht minimal aus zwei Phasen (Clark & Schaefer, 1989, S. 265), der Präsentations- und der Akzeptanzphase (siehe auch die schematische Darstellung in Abb. 2.1):

Präsentationsphase: A präsentiert B eine Äußerung. Sie geht dabei davon aus, dass B ihr Hinweise geben wird, aus denen sie ausreichend darauf schließen kann, dass B die Äußerung verstanden hat.

Akzeptanzphase: B gibt A Hinweise, dass er glaubt, die Äußerung ausreichend verstanden zu haben. Er geht dabei davon aus, dass A – nachdem sie diese Hinweise registriert und verstanden hat – auch glaubt, dass er die Äußerung ausreichend verstanden habe. Hierzu kann A wiederum Hinweise geben, dass sie die von B gegebenen Hinweise in der entsprechenden Art und Weise verstanden hat.

Nun muss aber nicht jedes Detail einer Interaktion vollständig durch diesen Prozess des *Groundings* gehen, die Teilnehmer streben vielmehr an, ein für den jeweiligen Zweck der Konversation ausreichendes wechselseitiges Verständnis zu erlangen (Clark & Schaefer, 1989). Dies erstreckt sich über vier Ebenen (Dillen-

bourg & Traum, 2006). Möchte A an B die Information X mitteilen, dann kann A aufgrund der verfügbaren Hinweise schließen,

1. **Verfügbarkeit:** dass die Information X für B (nicht) verfügbar ist,
2. **Wahrnehmbarkeit:** dass B die Information X (nicht) wahrgenommen hat,
3. **Verstehen:** dass B die Information X (nicht) verstanden hat,
4. **Zustimmung:** dass B der Information X (nicht) zustimmt.

Clark & Brennan (1991) erweiterten ihre Theorie, um den Einfluss medialer Eigenschaften auf das Kommunikationsverhalten erklären zu können. Grundannahme ist hierbei, dass die Eigenschaften eines Mediums die Kommunikationsmöglichkeiten beschränken oder auch erweitern können. Tabelle 2.1 gibt einen Überblick über eine Reihe medialer Eigenschaften, welche Einfluss auf die kommunikativen Strategien haben. Diese Eigenschaften beeinflussen den Aufwand, um einen Beitrag zu formulieren, ihn zu produzieren und zu verstehen, Missverständnisse auszuräumen und auf Gegenstände in der gemeinsamen Umgebung zu deuten. Da die Teilnehmer versuchen, den gemeinsamen Aufwand bei der Kommunikation zu minimieren, führen unterschiedliche mediale Eigenschaften zu unterschiedlichen Kommunikationsstrategien.

2.3.2 Eigenschaften der Chatkommunikation

Chat ist ein computervermitteltes Medium zur synchronen Kommunikation, über das sich zwei oder mehr Teilnehmer in nahezu Echtzeit textuelle Nachrichten zusenden können. Mittlerweile ist es ein etabliertes Kommunikationsmedium, welches sich großer Beliebtheit im Freizeitbereich erfreut, aber auch zum Standardumfang von Lernplattformen gehört.

Eine kurze Geschichte des Chats

Kurz nachdem die ersten Computer 1970 im Rahmen des ARPAnet vernetzt waren, begannen die Nutzer dieses Systems sich über synchrone textuelle Kommunikation auszutauschen (Hamman, 1997). 1973 wurde das erste Konferenzsystem PLANET (Planning Network) von Jacques Vallee, Roy Amara und Robert Johansen entwickelt. Die eigentliche Geburtsstunde öffentlich verfügbarer Chatsysteme liegt im Jahr 1988, als der finnische Student Jarkko Oikarinen an der Universität von Oulu zusammen mit Jyrki Kuoppala und Jukka Pihl den Internet Relay Chat (IRC) entwickelte und auf einem Server (tolsun.oulu.fi) verfügbar machte. Dieses IRC-System basierte auf dem Konzept der Chaträume. Um Nachrichten mit anderen Teilnehmern auszutauschen, musste man sich in einem Chatraum „treffen“. Alle in diesem Raum verschickten Nachrichten wurden dann von den aktuell im Raum befindlichen Teilnehmern empfangen. Das System fand sehr schnell Verbreitung und die Nutzerzahl steigt seither stetig an (Oikarinen, 1993).

Tabelle 2.1: Mediale Eigenschaften, welche die Kommunikationsmöglichkeiten beeinflussen (nach Clark & Brennan, 1991)

Kopräsenz: A und B teilen dieselbe physikalische Umgebung. Im direkten Gespräch können die Kommunizierenden einander sehen und hören, erkennen, worauf die anderen sich gerade konzentrieren und was sie gerade machen. In anderen Medien ist dies nur eingeschränkt der Fall.[a]
Sichtbarkeit: A und B können sich sehen. Dies ist im direkten Gespräch der Fall. Ihnen stehen vielfältige gestische und mimische Ausdrucksmöglichkeiten zur Verfügung. In Videokonferenzsystemen kann man zwar einander sehen, aber nur eingeschränkt, was die anderen gerade hantieren oder worauf sie deuten.[b]
Hörbarkeit: A und B können sich hören. Im Gespräch wie auch beim Hinterlassen von Nachrichten auf einem Anrufbeantworter können durch die verbale Artikulation Ironie, Unsicherheit oder auch Eindringlichkeit ausgedrückt werden, in nichtverbalen Medien wie Brief oder Chat ist dies nicht möglich.
Gleichzeitigkeit von Produktion und Präsentation: B empfängt ungefähr gleichzeitig zur Produktion durch A. Dies ist im Gespräch gegeben; bei Briefen, Emails oder aber auch Chat ist dies nicht der Fall.
Simultanes Produzieren: A und B können simultan senden und empfangen. Im Gespräch können die Zuhörer ohne zu unterbrechen über die *Back-Channels* (z. B. mit einem Lächeln) „senden“. Analog ist ein simultanes Senden und Empfangen auch bei *Talk*-Programmen (bei denen die Tastatureingaben Zeichen für Zeichen übertragen werden) möglich. Simultanität ist nicht bei Briefen und Emails gegeben.
Beitragsabfolge: Die Beiträge von A und B bilden eine Sequenz. Ein Gespräch besteht üblicherweise aus einer kohärenten Abfolge von Redebeiträgen, ein Beitrag aus einer anderen Konversation würde als Unterbrechung wahrgenommen. In anderen Medien wie Email oder Anrufbeantworter können Beitrag und Erwiderung durch eine Vielzahl anderer (für diese Konversation irrelevante) Beiträge getrennt sein.
Wiederholbarkeit: B kann die Beiträge von A wiederholt ansehen bzw. abspielen. Im Gespräch ist ein Redebeitrag flüchtig, in anderen Medien sind die Beiträge gespeichert (auf der Anrufbeantworterkassette, auf dem Papier oder als digitales Dokument) und können somit wiederholt angesehen bzw. angehört werden.[c]
Überarbeitbarkeit: A kann einen Beitrag an B (vor dem Absenden) überarbeiten. Im Gespräch und bei der Verwendung audio-visueller Kommunikationsmedien ist dies nicht möglich. Erfolgt die Produktion einer Äußerung jedoch vor einem dedizierten Absenden (wie bei Briefen, Emails oder auch im Chat), so kann der Beitrag meist beliebig überarbeitet werden.

[a] Während Clark & Brennan (1991) diese Dimension auf die tatsächliche *physikalische* Kopräsenz, also das Miteinander am selben Ort, beschränken, wird die Kopräsenz von anderen Autoren (z. B. Convertino et al., 2005) weiter gefasst: Kopräsenz ist das Erleben des „Zusammenseins“, welches mehr oder weniger gegeben sein kann, eben auch in virtuellen Räumen (siehe zur Diskussion Nova, 2005; IJsselsteijn & Riva, 2003).

[b] Die Bereitstellung eines Videokanals reicht allein nicht aus, um die Vorteile der Sichtbarkeit in der Kopräsenz zu erreichen (Whittaker, 2003).

[c] Dillenbourg & Traum (2006) argumentieren, dass die Möglichkeit zur Wiederholung nur ein Aspekt der zeitlich dauerhaften Verfügbarkeit der Beiträge sei. Sie bevorzugen deshalb die Bezeichnung *Persistence* für diese Medieneigenschaft.

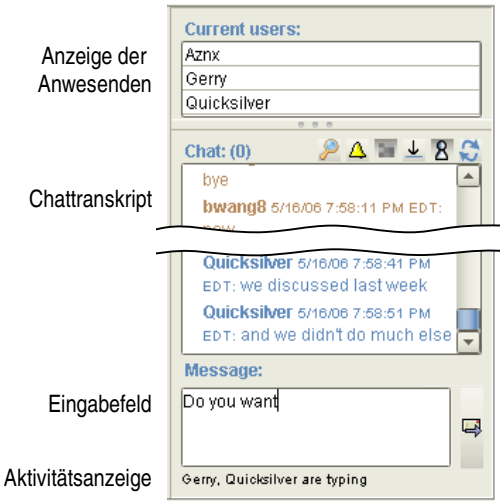


Abbildung 2.2: Benutzungsschnittstelle eines typischen Chatwerkzeugs mit Anzeige der sich gerade im virtuellen Raum befindlichen Personen, dem Chattranskript und dem Eingabefeld für die Erstellung von Nachrichten. Unter diesem befindet sich die Anzeige, wer gerade an einem Beitrag arbeitet

Die medialen Eigenschaften ...

Die Funktionsweise von Chat ist recht einfach: In einem Eingabefenster können Texte erstellt werden, die anschließend durch Tastendruck an die Teilnehmer verschickt werden. Dort werden sie in der Reihenfolge des Eintreffens – üblicherweise als fortlaufender Text, dem sogenannten Chattranskript – angezeigt. Abbildung 2.2 zeigt den typischen Aufbau eines Chatwerkzeugs.

Die Eingabe des Textes selbst ist jedoch durch die anderen Teilnehmer nicht beobachtbar, die Produktion einer Äußerung und deren Rezeption durch die anderen erfolgt also prinzipiell zeitlich versetzt, weshalb es auch als „quasi-synchrones“ Medium[4] bezeichnet wird (Garcia & Jacobs, 1999).

Betrachtet man das Medium Chat im Lichte der in Tabelle 2.1 genannten medialen Eigenschaften, so ergibt sich in erster Näherung folgendes Bild: *Kopräsenz*, *Sicht-* und *Hörbarkeit* sind offensichtlich nicht gegeben, gleiches gilt für die *Gleichzeitigkeit von Produktion und Präsentation*. Chat erlaubt zwar das *simultane Produzieren von Beiträgen*, diese können jedoch nicht – wie das zustimmende simultane „Mhm“ im Gespräch – aufeinander bezogen sein, denn der jeweilige Inhalt ist noch nicht bekannt. Auf die Besonderheiten der *Beitragsabfolge* wird später im De-

[4] Dies ist eine technische Sicht auf die Synchronizität eines Mediums, denn Synchronizität ist „weniger ein physikalischer oder technischer Parameter als vielmehr eine soziale Regel bzw. ein meta-kommunikativer Kontrakt: Der Sprecher setzt voraus, dass sein Zuhörer auf die Botschaft wartet und auf diese so schnell wie möglich reagiert; entscheidend ist dabei die Aufrechterhaltung des subjektiven Gefühls für die Synchronizität der Auseinandersetzung.“ (Reinmann-Rothmeier & Mandl, 1999, S. 14)

tail eingegangen. Auf globaler Ebene hängt diese jedoch auch von der Nutzung des Raums ab. Öffentliche Freizeitchats ähneln eher der Konversation bei einer Party mit einer Vielzahl simultan ablaufender Gespräche, die sich denselben Kanal teilen.

Aufgrund der Textualität und der damit einhergehenden Persistenz zeichnet sich Chat aber durch *Wiederholbarkeit* aus: Ältere Beiträge können wiederholt gelesen werden. Da die Beiträge darüber hinaus erst nach vollständiger Eingabe explizit durch den Nutzer abgeschickt werden, können die Beiträge beliebig *überarbeitet* werden.

Die Wiederholbarkeit und die Überarbeitbarkeit einer Nachricht vor dem Senden erweitern entsprechend der *Grounding Theory* die Kommunikationsmöglichkeiten, da sie die *Grounding*-Kosten reduzieren. So erlaubt die Möglichkeit zur Überarbeitung den Verfassern, den Beitrag so lange umzuformulieren, bis er ihren jeweiligen Ansprüchen genügt. Aber auch die Beschränkungen des Mediums können zu unerwarteten positiven Effekten für das kollaborative Lernen führen (Jermann, 2004). Die Verschriftlichung bedingt eine sorgfältigere Planung der eigenen kommunikativen Beiträge. Indem das Medium nur beschränkte Ausdrucksmöglichkeiten erlaubt, müssen die Lernenden geeignete Repräsentationen finden, die auf das Wesentliche reduziert sind. Dies kann zu einer tieferen und kritischeren Auseinandersetzung mit dem Kommunikationsgegenstand führen.

. . . und ihre Folgen

Aus linguistischer Perspektive teilen Chatdiskurse eine Reihe von Merkmalen mit gesprochenen Diskursen (Koch & Oesterreicher, 1994), wie z. B. die erhöhte Akzeptanz syntaktischer und grammatikalischer Oberflächenfehler und die Verwendung informeller Redewendungen. Dies wird auf die Ähnlichkeit der Kommunikationssituation (Murray, 2000) oder mehr noch auf die kommunikative Grundhaltung (Beißwenger, 2002) zurückgeführt. Es zeigt sich ein Zusammenhang zwischen Austauschgeschwindigkeit und Mündlichkeit: Je schneller die Teilnehmer aufeinander reagieren, desto mehr tragen die Beiträge Merkmale der mündlichen Sprache.

Wie wirken sich die medialen Eigenschaften nun auf die Kommunikation aus? Prominentes Merkmal von Chat-Diskursen ist ihre inkohärente Struktur, aufeinander folgende Beitrage sind – im Gegensatz zu mündlichen Gesprächen – oft nicht aufeinander bezogen (Herring, 1999).

Dies ist einerseits Folge der medial bedingten zeitlichen Trennung von (privater) Produktion und Präsentation eines Beitrags (Garcia & Jacobs, 1999). Hat A einen Beitrag verschickt und antwortet B auf diesen, so können während der Zeit, die B zum Eingeben benötigt, Beiträge anderer Teilnehmer eintreffen. Der Beitrag von B wird dann nach dem „Mühlenprinzip“[5] nicht an der von B intendierten

[5] „Wer zuerst kommt, mahlt zuerst!“

Stelle (nämlich adjazent zum Beitrag von A), sondern hinter all den mittlerweile verschickten Nachrichten der anderen angezeigt.[6]

Dasselbe Muster von durch Nachrichten anderer Teilnehmer unterbrochenen Adjazenzpaaren findet sich jedoch auch in Chatapplikationen, die anzeigen, wenn andere Teilnehmer mit dem Tippen einer neuen Nachricht beginnen. Dies würde zumindest theoretisch eine Koordination beim *Turn Taking* erlauben. Aber auch in solchen Systemen neigen die Chatnutzer dazu, simultan zu tippen (Isaacs et al., 2002). Insofern kann man davon ausgehen, dass im Chat die in Gesprächen übliche Konvention – „es redet immer nur einer" – nicht angewandt wird. Hierfür spricht auch die Beobachtung, dass in Chatdiskursen mehrere Themen simultan besprochen, mehrere „Gespräche" gleichzeitig stattfinden (Beißwenger, 2003).

Die fehlende lokale Kohärenz der Beiträge führt dazu, dass der Aufwand zum Verstehen eintreffender Nachrichten erhöht ist. Im ungünstigsten Falle kann der intendierte Kontext[7] einer Nachricht nicht mehr identifiziert und somit auch ihre Bedeutung nicht mehr verstanden werden (Pimentel et al., 2003).

2.3.3 Minimalanforderung an Chatwerkzeuge

Auf technischer Seite ergibt sich für das Medium Chat somit als Minimalanforderung, dass durch die Übertragung zumindest die Reihenfolge der Beiträge für alle Teilnehmenden gleichermaßen bewahrt (alle Teilnehmer empfangen die Beiträge in derselben Reihenfolge) und die Vollständigkeit sichergestellt wird (es geht kein Beitrag verloren), welches in folgender Anforderung zum Ausdruck kommt:

C 1. Das Chatwerkzeug muss konsistente und vollständige Chattranskripte bereitstellen.

Um mit den Eigenarten des Mediums umzugehen, haben sich in der Chatkommunikation eine Reihe spezifischer Kommunikationsstrategien etabliert (Murray, 2000). So tendieren die Chatnutzer zur häufigen Verwendung konventionalisierter Abkürzungen und Akronyme[8] wie z. B. „TTIUWP" für „This Thread Is Useless Without Pictures" oder auch „KISS" für „Keep It Short and Simple". Des Weiteren werden Oberflächenfehler während der Beitragsproduktion wie Tippfehler und falscher Satzbau akzeptiert. Um emotionale Konnotationen auszudrücken und zu desambiguieren, werden konventionalisierte Symbole (Emoticons, z. B. stellt „:-)" ein seitwärts zu betrachtendes lachendes Gesicht dar) verwendet (Rivera et al.,

[6] Eine ausführliche Diskussion zum Zusammenhang technischer/medialer Eigenschaften und Beitragssequenzialität findet sich in Beißwenger (2003).

[7] Der linguistisch korrekte Begriff ist „Kotext": „‚Co-text' designates surrounding text that has been written before or after an enunciation and that provides elements for understanding it. This term is used in Linguistics as an effort to solve the ambiguity of the word context, which has a wider meaning." (Pimentel et al., 2003)

[8] Im WWW finden sich viele Listen der gebräuchlichen (und auch außergewöhnlichen) Akronyme, z. B. listet `http://de.wikipedia.org/wiki/Liste_der_Abkürzungen_(Netzjargon)` über 370 Abkürzungen.

1996). Viele Chatapplikationen stellen hierfür Icons bereit, die in den Chatbeitrag eingefügt werden können.

Wie dem Problem des drohenden Kontextverlustes begegnet werden kann, zeigen die Kommunikationsmethoden erfahrener Chatter. Diese verwenden *cohesive Devices* (Nash, 2005) als Hinweise, worauf sie sich mit ihrem Beitrag beziehen. Solche Hinweise sind z. B. das Aufgreifen von in anderen Beiträgen verwendeten Wendungen oder die direkte Adressierung. Ein Chatwerkzeug sollte dieses Explizitmachen von Bezügen unterstützen, indem ein Beitragsautor seinen Chatbeitrag direkt mit anderen Beiträgen bzw. Beitragsteilen verknüpfen kann (Pfister & Mühlpfordt, 2002). Diese Verknüpfung muss dann den anderen Teilnehmern zusammen mit dem verweisenden Beitrag zugänglich sein bzw. angezeigt werden. Ein Chatwerkzeug muss also folgender Anforderung genügen:

C 2. Das Chatwerkzeug muss die explizite Bezugnahme auf einen oder mehrere andere Beiträge bzw. Beitragsbestandteile ermöglichen.

Die zeitliche Trennung zwischen Produktion und Rezeption einer Äußerung wird nicht nur durch die technisch bedingte zeitlich versetzte Präsentation hervorgerufen. Werry (1997) weist darauf hin, dass die Eingabe einer textuellen Nachricht die volle Konzentration verlangt und damit dem Strom der eintreffenden Nachrichten kaum Aufmerksamkeit gewidmet werden kann. Hierfür ist jedoch mit dem Chattranskript, also der fortlaufenden Anzeige der eintreffenden Nachrichten, bereits eine Lösung „eingebaut“. Ältere Nachrichten sind persistent und können auch später noch nachgelesen werden. Einige Befunde sprechen dafür, dass diese Persistenz älterer Nachrichten im Chattranskript eine wichtige Funktion für die Kommunikation hat. Gergle et al. (2004b) fanden, dass Paare mittels Chat eine Problemlöseaufgabe effizienter bearbeiteten, wenn ihr Chattranskript 12 Zeilen statt nur die letzten zwei Zeilen umfasste.

Die Persistenz des Chattranskripts kann verschiedene Ausprägungen annehmen. Im einfachsten Falle stehen nur die Nachrichten zum nachträglichen Lesen zur Verfügung, die während der Anwesenheit des Teilnehmers empfangen wurden. Im erweiterten Falle werden alle im Chatraum verschickten Nachrichten vom System gespeichert. Diese Variante der Persistenz erlaubt die nachträgliche Reflexion über den Diskurs (O'Malley, 1995; Reimann & Zumbach, 2001), der Diskurs „may be searched, browsed, replayed, annotated, visualized, restructured, and recontextualized“ (Erickson, 1999). Sie bietet darüber hinaus nachträglich eintreffenden Teilnehmern die Möglichkeit, den bisherigen Diskurs zu betrachten und nachzuvollziehen. Hieraus folgt als Minimalanforderung an Chatwerkzeuge:

C 3. Das Chattranskript muss systemweit persistent verfügbar sein.

Wie koordinieren die Teilnehmer im Chat nun ihr kommunikatives Handeln? Garcia & Jacobs (1999) fanden bei der konversationsanalytischen Auswertung von

Chatsitzungen, dass Teilnehmer – im Widerspruch zur oben angeführten Argumentation von Werry (1997) – während der Beitragserstellung durchaus auf eintreffende Nachrichten reagieren und ihren Beitrag gegebenenfalls an die angekommene Nachricht anpassen. Erleichtert wird die Koordination auch durch die in modernen Chatsystemen anzutreffende Tipp-*Awareness* (Tran et al., 2005). Hierbei wird im einfachsten Falle signalisiert, dass ein anderer Teilnehmer anfängt, einen Beitrag zu tippen. Ausgefeiltere Systeme gehen darüber hinaus und erkennen, wann diese Teilnehmer beim Tippen pausieren, und zeigen dies entsprechend bei den anderen Teilnehmern an (Ogura et al., 2003). Diese Information hilft auch der Missinterpretation von Schweigen (Garcia & Jacobs, 1998) vorzubeugen. Hierunter versteht man das Problem, dass ein vermeintliches Ausbleiben einer Antwort (weil der Reagierende einige Zeit zum Verfassen einer Antwort benötigt) als Nichtreagieren aufgefasst wird. Welche Informationen für die Teilnehmer notwendig und hilfreich sind, ist Gegenstand der Forschung (Holmer & Wessner, 2005) und geht über den Rahmen dieser Arbeit hinaus. Wichtig ist jedoch, dass solche Informationen erfasst, an die Computer der anderen Teilnehmer übertragen und dort angezeigt werden. Vor dem Hintergrund dieser Betrachtungen ergibt sich als Minimalanforderung:

C 4. Den Teilnehmern müssen *Awareness*-Informationen über simultane Beitragserstellungen anderer Teilnehmer zur Verfügung gestellt werden.

2.4 Computervermitteltes kollaboratives Konstruieren

Im vorangegangenen Kapitel wurden die Anforderungen an das Kommunikationsmedium für den sprachlichen Diskurs identifiziert. Nun soll der andere Aspekt kollaborativen Lernens – die Konstruktion gemeinsamer Artefakte – betrachtet werden. Hierfür bedarf es für das Lernen in örtlich verteilten Lernarrangements geeigneter Konstruktionswerkzeuge, die es der Lerngruppe erlauben, aufgabenrelevante Artefakte zu erstellen, zu betrachten und zu bearbeiten. Der Fokus dieser Arbeit liegt dabei weniger auf dem Funktionsumfang der aufgabenspezifischen Werkzeuge, als vielmehr auf der Frage, wie diese mit dem textuellen Kommunikationswerkzeug integriert werden müssen, um die kollaborativen Aktivitäten, bestehend aus dem Diskurs und der gemeinschaftlichen Konstruktion, zu unterstützen. Aus diesem Grunde zielt dieses Kapitel auf allgemeine Anforderungen an kollaborative Konstruktionswerkzeuge, unabhängig von den je nach Artefakttypus unterschiedlichen Berarbeitungsfunktionen.

Zuerst werden kollaborative Konstruktionswerkzeuge und deren Funktionen für die Kollaboration allgemein charakterisiert. Anschließend wird darauf eingegangen, wie die Kollaborierenden ihr gemeinsames Tun koordinieren und ein *Grounding* ihrer Aktionen sicherstellen. Zum Abschluss dieses Kapitels werden dann die allgemeinen Anforderungen an kollaborative Konstruktionswerkzeuge formuliert.

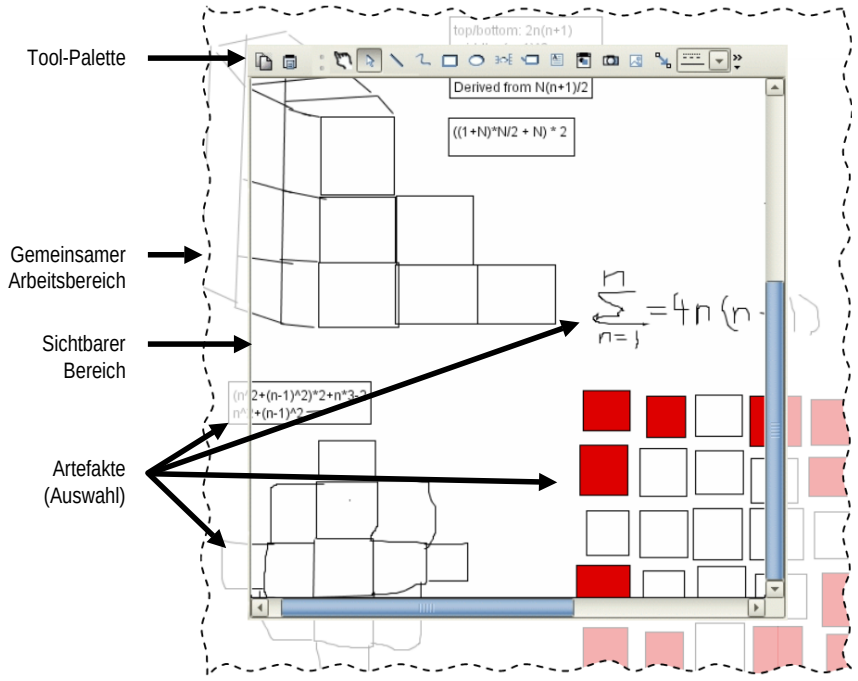


Abbildung 2.3: Benutzungsschnittstelle eines beispielhaften kollaborativen Konstruktionswerkzeugs. Der gemeinsame Arbeitsbereich erstreckt sich meist über den (individuell) sichtbaren Bereich hinaus. Mit den in der Palette angebotenen Tools können Artefakte erzeugt, bearbeitet und arrangiert werden

2.4.1 Charakterisierung kollaborativer Konstruktionswerkzeuge

Unter einem kollaborativen Konstruktionswerkzeug wird in dieser Arbeit eine Software verstanden, die den Nutzern einen gemeinsamen Arbeitsbereich zur Verfügung stellt, in dem Objekte erstellt, angezeigt und manipuliert werden können, wobei die Objekte und Effekte der Manipulationen gleichzeitig[9] auf den korrespondierenden Bildschirmbereichen der Kooperationspartner sichtbar werden (Whittaker, 2003). Diese Systeme werden auch WYSIWIS-Systeme *(What You See Is What I See)* genannt.[10]

In Abbildung 2.3 ist als ein Beispiel ein *Shared Whiteboard* zu sehen. Geeignete Werkzeuge erlauben es, Artefakte im gemeinsamen Arbeitsbereich zu erzeugen,

[9] Die Gleichzeitigkeit ist eine Idealisierung, denn damit die anderen Teilnehmer die neu erstellten Objekte bzw. die Effekte der Manipulationen sehen können, müssen diese über das Netzwerk übertragen werden. Die Übertragungszeit variiert mit der Leistungsfähigkeit der beteiligten Computer und der Netzwerkverbindungen.

[10] Man unterscheidet zwischen „*strict*“ und „*relaxed*“ WYSIWIS (Stefik et al., 1987a). In Systemen, die einem strikten WYSIWIS folgen, sehen die Nutzer tatsächlich immer dasselbe. Verschiebt z. B. ein Teilnehmer den sichtbaren Ausschnitt eines Arbeitsbereichs, so wird dieser auch bei den anderen Teilnehmern verschoben. In *relaxed*-WYSIWIS-Systemen können sich die Sichten der Teilnehmer dagegen je nach Benutzereinstellung unterscheiden.

zu bearbeiten und gegebenenfalls in ihrer räumlichen Anordnung zueinander zu verändern. Die Ausdehnung des gemeinsamen Arbeitsbereichs ist meist größer als der sichtbare Bereich, da die verfügbare Bildschirmfläche heutiger Monitore z. B. gegenüber einer Schultafel immer noch recht beschränkt ist.

Ein kollaboratives Konstruktionswerkzeug stellt somit die Funktionalitäten einer vergleichbaren Einzelbenutzeranwendung bereit, wobei die Teilnehmer von ihren jeweiligen, miteinander vernetzten, Computern aus gemeinsam die Artefakte sehen und bearbeiten können. Aus technischer Perspektive muss das kollaborative Konstruktionswerkzeug sicherstellen, dass die Zustände der gemeinsam berarbeiteten Artefakte auf den verschiedenen Rechnern der Teilnehmer konsistent sind. Damit lässt sich ein kollaboratives Konstruktionswerkzeug allgemein als verteiltes interaktives Medium auffassen (Mauve, 2000), wobei zwischen diskreten und kontinuierlichen Medien unterschieden werden kann.

Der Zustand eines diskreten Mediums wie dem erwähnten *Shared Whiteboard* verändert sich allein aufgrund von diskreten (meist durch Nutzeraktionen hervorgerufenen) Ereignissen. Verschiebt ein Teilnehmer ein Objekt an eine andere Position, so ändert sich der Zustand des Mediums. Solange diese aus den Aktionen resultierenden Operationen auf allen Rechnern in der korrekten Reihenfolge ausgeführt werden, ist die Konsistenz der jeweiligen lokalen Zustände auf den verschiedenen Rechnern gewahrt.

In kontinuierlichen Medien hängt dagegen der Zustand auch von der Zeit ab. Bei einer Operationsausführung muss hierbei zusätzlich sichergestellt werden, dass sie zum korrekten Zeitpunkt erfolgt. Beispiele hierfür sind das gemeinsame Betrachten und Annotieren von Videos oder von Animationen. Hierfür bedarf es komplexerer Konsistenzmechanismen, wie sie z. B. von Mauve (2000) oder Cristian et al. (1985) entwickelt wurden. Diese Arbeit beschränkt sich jedoch auf die Betrachtung diskreter Medien.

Bei der Entwicklung der ersten synchronen *Groupware*-Applikationen standen die technischen Probleme verteilter Systeme im Vordergrund: Wie erfolgt die Synchronisation der jeweils lokalen Datenbestände, wie erfolgt die Kommunikation zwischen den Prozessen, wie können konfligierende Aktionen, die zu unerwarteten oder inkonsistenten lokalen Zuständen führen, vermieden bzw. aufgelöst werden?

Hierbei wurden zwei verschiedene Wege beschritten. Zum einen wurde der Ansatz verfolgt, die als Einzelbenutzeranwendungen vorhandenen Systeme für mehrere Nutzer durch sogenanntes *Application Sharing* verfügbar zu machen. In diesem Falle läuft eine Anwendung auf genau einem der beteiligten Computer, während die Rechner der anderen Teilnehmer allein als Ein- und Ausgabegerät für diese Anwendung dienen. Der Nachteil hierbei ist, dass nicht mehrere Benutzer gleichzeitig mit der Anwendung interagieren können, sondern immer nur einer (Schümmer & Schuckmann, 2001). Für das *Application Sharing* sind deshalb *Turn-Taking*-Mechanismen notwendig.

Der andere Weg bestand in der Entwicklung eigenständiger *Groupware*-Anwendungen, die das gleichzeitige Agieren mehrerer Teilnehmer erlauben. Dies

ging meist einher mit der Entwicklung von *Groupware-Frameworks*, wie z. B. GROVE (Ellis et al., 1991), COAST (Schuckmann et al., 1996) oder DreamTeam (Roth & Unger, 1998). Anfangs lag der Fokus auf Systemen für den Einsatz in kopräsenten Meetings (z. B. Cognoter, Stefik et al. 1987b), später auch für räumlich verteilte synchrone Kooperationen.

2.4.2 Mechanismen computervermittelten Ko-Konstruierens

Die primäre Funktion eines kollaborativen Konstruktionswerkzeuges besteht zunächst darin, einer örtlich verteilten Lerngruppe die gemeinsame Erstellung und Bearbeitung von aufgabenrelevanten Artefakten (siehe Kap. 2.1.3) zu ermöglichen. Diese Artefakte werden von der Gruppe gemeinsam konstruiert. Die eigentliche Konstruktionsarbeit können Gruppen ganz verschiedenartig organisieren. Nach Jermann (2002) lassen sich drei Organisationsformen unterscheiden. Zum einen können die Teilnehmer sich die Arbeit aufteilen; jeder Teilnehmer übernimmt hierbei eine eigenständige Teilaufgabe, wobei die Aktivitäten der einzelnen Teilnehmer auf die Konstruktion des Teilergebnisses ausgerichtet sind. Zum anderen können die Teilnehmer verschiedene Rollen bei der Konstruktion übernehmen. So finden sich Kollaborationsmuster, bei denen ein Teilnehmer die eigentliche Manipulation durchführt, während ein anderer nur beobachtet und kommentiert. Drittens können die Teilnehmer die Konstruktion aber im eigentlichen Sinne kollaborativ erledigen: Es werden keine spezialisierten Rollen oder Teilaufgaben übernommen, sondern alle Teilnehmer arbeiten gleichermaßen und gleichberechtigt an allen Teilaufgaben. Natürlich werden auch hier die einzelnen Artefaktmanipulationen jeweils nur durch einen Teilnehmer ausgeführt, die anderen sind jedoch an der Planung und Bewertung beteiligt, müssen diese – im Sinne des *Groundings* – verstehen und akzeptieren. Diese letzte Variante der Organisation ist für das kollaborative Lernen zu favorisieren (Pata & Sarapuu, 2003). Die Teilnehmer müssen hierbei ihre jeweiligen individuellen Aktionen aufeinander abstimmen, miteinander koordinieren und wechselseitig verstehen. Ein kollaboratives Konstruktionswerkzeug ist somit auch ein Medium für verschiedenste Arten sozialer Aktivität (Ngwenyama & Lyytinen, 1997).

Welche Merkmale und Besonderheiten zeigen nun aber Kollaborationen in örtlich verteilten Szenarien mittels eines Konstruktionswerkzeuges? In den meisten Untersuchungen hierzu werden die Konstruktionswerkzeuge zusammen mit einem Kommunikationskanal verwendet, oft ist dies ein Audio-, manchmal auch ein Videokonferenzsystem.[11] In einer Studie analysierten Whittaker et al. (1991) jedoch, wie räumlich verteilte Triaden allein mittels eines *Shared Whiteboard* kollaborieren. Hierzu sollten die Gruppen zwei Aufgaben bearbeiten. Die erste bestand darin, eine priorisierte Liste von Kriterien für den Kauf eines Hauses zu erstellen, und die zweite im Finden von Terminen für gemeinsame Treffen. Das verwendete *Shared Whiteboard* erlaubte es, Freihandzeichnungen zu erstellen, kurze Texte

[11] Es deutet einiges darauf hin, dass ein zusätzlicher Videokanal gegenüber der rein auditiven Kommunikation im allgemeinen keinen Vorteil hat (Whittaker, 2003).

einzugeben und Objekte wieder zu entfernen. Für die Analyse wurden die Eingaben der Teilnehmer in drei Kategorien eingeteilt: *Artefakte* (mittels Freihandzeichnungen und kurzer Texte erstellte Listen, Tabellen, Kalender und ähnliches), *Prosa* (Texte, die nicht Bestandteile von Artefakten waren) und *Deixis* (Zeichnungen, die einen Bezug zu vorherigen Eingaben hatten, wie z. B. Fragezeichen). Die Elemente der drei Kategorien unterschieden sich auffällig hinsichtlich ihrer durchschnittlichen Verweildauer, also wie lange die Elemente im gemeinsamen Arbeitsbereich sichtbar waren, bevor sie wieder gelöscht wurden. Die Autoren schlossen daraus, dass die Gruppen das Medium flexibel für die unterschiedlichen Aspekte der gemeinsamen Problemlösung nutzten. Die *Artefakte* wurden eher zur Repräsentation der Probleminhalte und *Prosa* eher für die Kommunikation über den Inhalt sowie die Koordination genutzt. *Artefakte* zeichneten sich dadurch aus, dass sie relativ lange im gemeinsamen Arbeitsbereich verfügbar blieben, erst nach Rücksprache mit den anderen Teilnehmern gelöscht und in hohem Maße simultan von verschiedenen Teilnehmern erstellt wurden. Im Gegensatz hierzu wurde *Prosa* wesentlich eher und mit weniger Rücksprache wieder gelöscht, während deren Eingabe weniger häufig simultan zu Eingaben anderer Teilnehmer erfolgte. Für die als *Deixis* kategorisierten Elemente ergab die Analyse, dass diese sich wesentlich häufiger auf *Artefakte* als auf *Prosa* bezogen, wobei die Bezugnahme vor allem auf Elemente erfolgte, die schon länger verfügbar waren.

Die Teilnehmer in der zitierten Studie waren also in der Lage, die beschränkten Möglichkeiten, die ihnen das *Shared Whiteboard* zur Verfügung stellt, flexibel für die verschiedenen Aspekte der Kommunikation und Koordination zu nutzen. Nach Endsley (1995) setzt dies seitens der Teilnehmer ein wechselseitiges Wissen – die sogenannte *Situational Awareness* – über das Tun der anderen, den Zustand der Bearbeitung und die gemeinsame Umgebung voraus. Dieses Wissen wird mittels der in der Umgebung verfügbaren Hinweise aufgebaut und aufrechterhalten. Die drei wichtigsten Informationsquellen hierfür sind nach Gutwin & Greenberg (2002):

Kommunikation als Konsequenz des Agierens: Teilen die Lernenden eine gemeinsame Umgebung, so können sie einander beim Manipulieren mit den Objekten der Arbeitsumgebung beobachten. Indem zum Beispiel ein Schachspieler einen Zug macht, zu einer Figur greift, diese anhebt und auf ein anderes Feld stellt, wird gleichzeitig auch dem Gegenspieler vermittelt, welche Figur von welcher Position auf eine neue bewegt wird. Beide wissen nun, dass jetzt der Gegenspieler am Zug ist.

Veränderungen an den Artefakten: Unabhängig von der Beobachtbarkeit der Akteure bieten die Artefakte selbst Informationen über aktuelle Aktivitäten. Die beobachtbaren Veränderungen an den Artefakten (zum Beispiel ihre Bewegung, die Geräusche des Verschiebens etc.) erlauben Rückschlüsse darauf, was mit ihnen gemacht wurde.

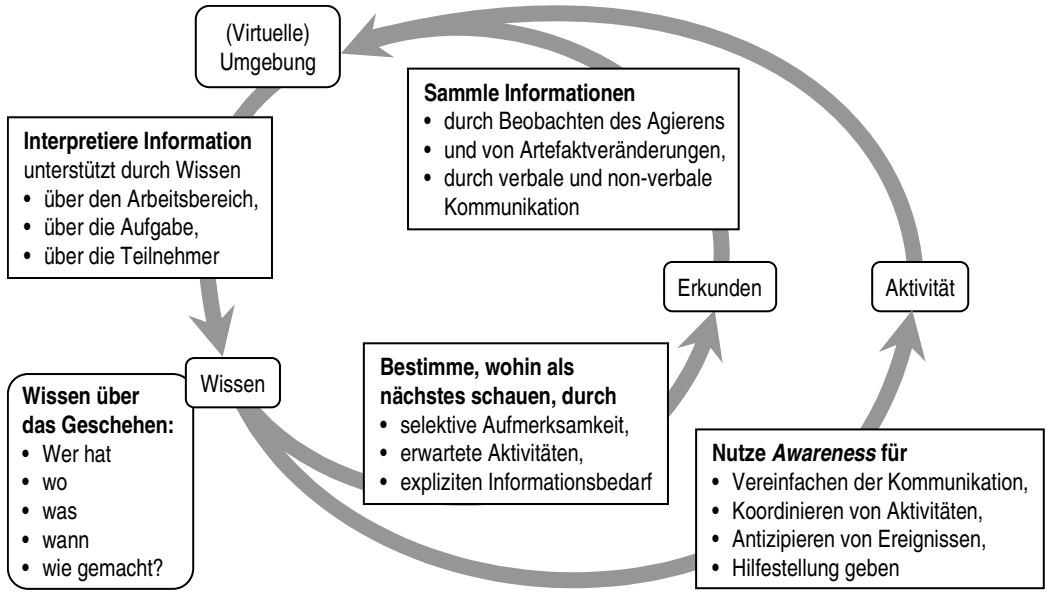


Abbildung 2.4: Das *Awareness*-Konzept für gemeinsame Arbeitsbereiche (nach Gutwin & Greenberg, 2002, S. 439)

Intentionale Kommunikation und Gesten: Natürlich geben sich die Teilnehmer gegenseitig auch intentional Hinweise, indem sie die anderen über ihr Tun informieren. Dies kann einerseits sprachlich erfolgen, aber die Teilnehmer können auch „die Aktionen sprechen lassen“ (Gergle et al., 2004a): ein Schachspieler kann seinem Mitspieler mittels Geste oder durch Demonstration zeigen, dass ein anderer Zug seines Erachtens besser ist.

Gutwin & Greenberg (2002) erstellten ein umfangreiches Modell darüber, wie *Awareness* aufgebaut und aufrechterhalten wird. Eine schematische Darstellung findet sich in Abbildung 2.4. Das Wissen darüber, was im gemeinsamen Arbeitsbereich gerade passiert, wer etwas wie gemacht hat oder gerade tut, wird in einem Wahrnehmungs-Aktions-Zyklus aufgebaut und aufrechterhalten. Die Kollaborierenden sammeln Informationen, die die Umgebung und die Kollaborationspartner bereitstellen, und integrieren diese in ihr Wissen über das aktuelle, vergangene und erwartete Geschehen. Hieraus können sie wiederum ableiten, an welchen Stellen weitere Informationen zu erlangen sind. Das Wissen über das Geschehen, über das Tun der anderen, hilft ihnen bei ihren Aktivitäten. Es erleichtert die Kommunikation, hilft bei der Koordination der Aktivitäten und unterstützt die Teilnehmer bei der gegenseitigen Hilfestellung.

In örtlich verteilten Szenarien, in denen sich die Teilnehmer nicht sehen und hören können, stehen diese Informationen nur eingeschränkt oder gar nicht zur Verfügung. Eine wichtige Funktionalität von kollaborativen Konstruktionswerkzeugen ist deshalb, geeignete *Awareness*-Informationen über das momentane und

vergangene Tun der Anderen bereitzustellen. Die Informationen über die aktuellen Aktivitäten der anderen sollten hierbei am Ort des Geschehens präsentiert werden: Modifiziert zum Beispiel ein Teilnehmer ein Objekt, so sollte die entsprechende *Awareness*-Information an diesem Objekt dargeboten werden (Gutwin & Greenberg, 2002).

Wie können nun aber Informationen über das vergangene Geschehen verfügbar gemacht werden? Oben wurde darauf eingegangen, dass in einem diskreten Medium jede Modifikation der Artefakte durch die Teilnehmer einen neuen Zustand definiert. Der gemeinsame Konstruktionsverlauf entspricht somit der Abfolge aller Modifikationen, er kann durch die Sequenz aller Artefaktmodifikationen beschrieben werden.

Die Bereitstellung des Konstruktionsverlaufs als Ressource für die Lernenden entspricht auch der allgemeinen Anforderung **A 3**, dass die Kollaborationsumgebung die Reflexion über den gemeinsamen Kollaborationsverlauf unterstützen muss, denn „eine Voraussetzung für reflektierendes Lernen ist, dass ‚Spuren' des eigenen (individuellen oder kollektiven) Problemlösehandelns und/oder Sachverständnisses verfügbar sind, die zum Gegenstand von Analysen gemacht werden können" (Reimann & Zumbach, 2001).

2.4.3 Anforderungen an kollaborative Konstruktionswerkzeuge

Welche allgemeinen Anforderungen stellen sich nun an kollaborative Konstruktionswerkzeuge? Zum ersten muss sichergestellt sein, dass den synchron lernenden Teilnehmern konsistente Sichten auf die gemeinsamen Artefakte bereitstehen, so dass als Anforderung gilt:

K 1. Konstruktionswerkzeuge müssen den Zustand der Artefakte für alle Teilnehmer konsistent halten.

Des Weiteren muss die Aufrechterhaltung der *Situational Awareness* unterstützt werden, indem geeignete *Awareness*-Informationen über die momentanen Aktivitäten der anderen erfasst, an die Computer der Teilnehmer übertragen und dort dargeboten werden. Hieraus folgt als Anforderung:

K 2. Konstruktionswerkzeuge müssen *Awareness*-Informationen bereitstellen.

Für die Unterstützung der Reflexion über den gemeinsamen Kollaborationsprozess muss den Lernenden der vergangene Konstruktionsverlauf als Ressource verfügbar gemacht werden. Diese Ressource muss während einer Sitzung – auch für später eintreffende Teilnehmer – und in nachfolgenden Sitzungen bereitstehen, so dass in Anlehnung an Anforderung **C 3** formuliert werden kann:

K 3. Der Konstruktionsverlauf muss systemweit persistent verfügbar sein.

2.5 Integrationsbedarf für duale Interaktionsräume

Wie in Abschnitt 2.2 ausgeführt, fokussiert diese Arbeit auf Kollaborationsumgebungen für die örtlich verteilte synchrone Kollaboration, die als duale Interaktionsräume zwei Interaktionsbereiche bereitstellen: Ein Medium für die sprachliche Kommunikation und ein Konstruktionswerkzeug für die gemeinsame Erstellung und Bearbeitung der aufgabenrelevanten Artefakte. In den letzten beiden Abschnitten wurde auf die Besonderheiten der computervermittelten Kommunikation und kollaborativen Konstruktion eingegangen. Es sollen nun die Folgen des einfachen Nebeneinanders von Kommunikations- und Konstruktionswerkzeug mit dem Ziel betrachtet werden, die Probleme des Nebeneinanders zu identifizieren und entsprechend Anforderungen an die Integration der beiden Werkzeuge abzuleiten.

2.5.1 Kollaboration in dualen Interaktionsräumen

In Kapitel 2.2 wurde darauf eingegangen, dass die beiden Interaktionsbereiche einer Kollaborationsumgebung zwei Funktionen übernehmen: Der Diskursbereich dient dem sprachlichen Austausch, während im Aufgabenbereich die aufgabenrelevanten Artefakte erzeugt und manipuliert werden können. Dieser Sicht entsprechend dienen die Artefakte als externe Repräsentation von Aufgabenkonzepten und als externes Gedächtnis für die Gruppe (Hutchins, 1990). Wie im vorangegangenen Abschnitt 2.4 jedoch deutlich wurde, sind die Aktionen im Arbeitsbereich auch kommunikative Akte zum Aufbau und zur Aufrechterhaltung der *Situational Awareness*. Die Bedeutung des gemeinsamen Arbeitsbereichs für die Kommunikation geht aber darüber hinaus, indem er das *Grounding* in der Kommunikation erleichtert (Kraut et al., 2002, S. 32f):

Erleichtern der Beitragsproduktion: Durch die gemeinsam verfügbare Repräsentation des Aufgabenzustands sowie der Aktionen in der Repräsentation werden die Kosten der Produktion von Beiträgen, die sich auf den Aufgabenzustand und die Aktionen beziehen, reduziert.

Beobachten des Verstehens: Indem die Aktionen der Kooperationspartner im gemeinsamen Arbeitsbereich beobachtet werden können, stehen diese als Hinweise des Aufgabenverständnisses bereit. Die visuelle Rückmeldung ist weniger relevant, wenn die Aufgabe eine geringe Komplexität aufweist bzw. wenn die Teilnehmer über ein gemeinsames eindeutiges Vokabular zur Beschreibung der Aufgabe und der Aktionen verfügen (z. B. Beschreibung der Züge in einer Fernschachpartie).

Es ist also zu erwarten, dass die Interaktionen der Teilnehmer im Chat und im Konstruktionswerkzeug aufeinander bezogen und miteinander verwoben sind. Ein Beispiel hierfür geben Suthers et al. (2003). Sie untersuchten die Kollaboration von Dyaden, die eine gemeinsame Hypothese über eine ungeklärte neurolo-

gische Erkrankung aufstellen sollten. Den örtlich verteilt Lernenden stand hierfür die Software Belvédère zur Verfügung, die in dieser Studie einen gemeinsamen Arbeitsbereich zur Konstruktion von Konzeptgraphen sowie einen Chat bereitstellte. Bei einer der Dyaden war nun gut zu beobachten, wie sich ihre Interaktion miteinander über die beiden Interaktionsbereiche erstreckte. Nachdem die beiden Lernenden alle Informationen als Knoten und Kanten des Konzeptgraphen gesammelt hatten, experimentierte einer der beiden mit verschiedenen räumlichen Anordnungen der Knoten, um anschließend im Chat zu äußern: „boy we got something", worauf sein Gegenüber erwiderte: „hehe ALUMINUM!!!!". Hierbei wird deutlich, dass ersterer offenbar davon ausgeht, dass die für ihn nun aus dem Graphen ableitbare Hypothese sich auch bereits dem anderen erschlossen hat, was dieser auch bestätigt. Die gemeinsame Konstruktion des Konzeptgraphen und das räumliche Organisieren durch einen der Teilnehmer führte also dazu, dass die beiden bereits Vermutungen über eine gemeinsame Hypothese hatten. Im anschließenden kurzen sprachlichen Austausch wurden diese Vermutungen wechselseitig abgesichert, es erfolgte das *Grounding*.

Ähnliches fanden Dillenbourg & Traum (2006). Sie untersuchten, wie zwei verschiedene Interaktionswerkzeuge, nämlich eine chatähnliche MOO-Umgebung[12] und ein *Shared Whiteboard*, von Dyaden insbesondere für das *Grounding* genutzt werden. Das Chat- bzw. MOO-Fenster und das *Shared Whiteboard* waren auf dem Bildschirm nebeneinander angeordnet. Die Paare hatten zur Aufgabe, die in der MOO-Umgebung verfügbaren Hinweise zur Lösung eines fiktiven Kriminalfalles zusammenzutragen und gemeinsam den Täter zu überführen. Hierzu konnten sie im *Shared Whiteboard* Skizzen anlegen, Texte in Textboxen notieren und diese frei arrangieren. Es zeigte sich, dass das *Whiteboard* vor allem für die Sammlung und Organisation von Fakten – den Problemzustand – genutzt wurde. Das *Grounding* dieser Informationen erfolgte dann vorrangig im Chat.[13] Den Grund für diese unterschiedliche Nutzung der beiden Interaktionsbereiche sehen Dillenbourg & Traum weniger in den graphischen Möglichkeiten des *Whiteboards* als vielmehr in den unterschiedlichen Persistenzgraden von Chat und *Whiteboard*. Während im gemeinsamen Arbeitsbereich die Informationen so lange verfügbar bleiben, bis sie durch die Teilnehmer gelöscht werden, ist der Chat nur semi-persistent, denn neu eintreffende Nachrichten schieben ältere aus dem sichtbaren Bereich. Dies beeinflusst zwei Aspekte der Kollaboration. Der erste ist die Wahl des Ortes, an dem eine Information präsentiert wird. Informationen, die für längere Zeit von Interesse sind und zur Repräsentation des Problemzustandes beitragen, werden eher im

[12] MOO steht für ***M**ulti User Dungeon **O**bject **O**riented*. Dies sind textbasierte Computersysteme für Rollenspiele, in denen viele Spieler gleichzeitig mit- und gegeneinander spielen können. Die virtuelle Spielwelt ist in Räume aufgeteilt, in denen man andere Spieler wie auch virtuelle Figuren treffen kann. Die Interaktion mit den Spielern und den Figuren wie auch die Navigation durch die Räume erfolgt rein textuell. Die Oberfläche eines MOOs ist ganz analog zu Chatwerkzeugen aufgebaut, sie besteht aus einem Transkript- und einem Eingabefenster.

[13] Interessant hierbei ist, dass zwischen der Präsentation einer Information im *Shared Whiteboard* und der Reaktion darauf durch den anderen Teilnehmer im Durchschnitt eine erstaunlich lange Zeitspanne von 70 Sekunden lag.

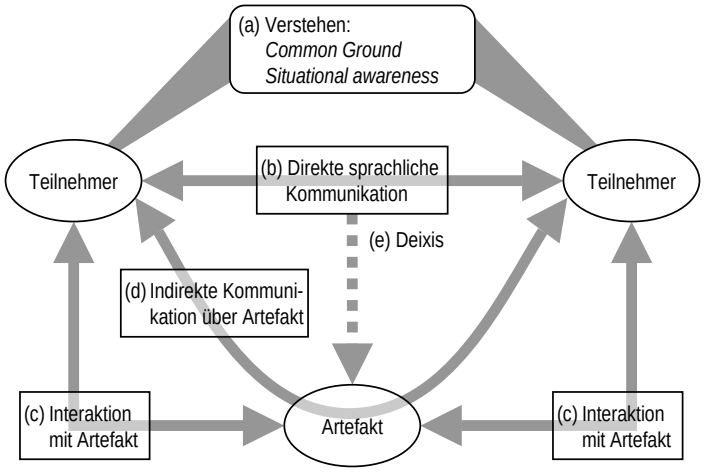


Abbildung 2.5: Kommunikation und Interaktion (nach Miles et al., 1993)

Shared Whiteboard notiert, während andere Informationen (zum Beispiel zur Koordination des weiteren Vorgehens) eher über den Chat kommuniziert werden. Der andere Aspekt betrifft den Aufwand, den die Teilnehmer zum *Grounding* einer Information treiben. Entsprechend der in Abschnitt 2.3 genannten vier *Grounding*-Ebenen beeinflusst der Grad an Persistenz einer Information den Bedarf an gegenseitiger Zusicherung: Kann ein Teilnehmer davon ausgehen, dass dem anderen die Information verfügbar ist (1. Ebene) und von diesem wahrgenommen wird (2. Ebene) und dass diese – wie z. B. bei einfachen Fakten – wohl auch verstanden (3. Ebene) und akzeptiert (4. Ebene) wird, dann bedarf es kaum eines weiteren Austauschs. Dies ist der Fall, wenn einfache Fakten im *Whiteboard* präsentiert werden. Es ist jedoch mit mehr *Grounding*-Aktivität zu rechnen, wenn Schlussfolgerungen im Chat präsentiert werden, denn dort sind diese nur zeitlich begrenzt sicht- und somit wahrnehmbar: ob die Schlussfolgerung akzeptiert, verstanden oder überhaupt wahrgenommen wurde, bleibt solange unklar, bis das Gegenüber entsprechende Hinweise gibt. Die von Dillenbourg & Traum angeführten Befunde entsprechen diesem Muster.[14]

Für die Kollaboration in dualen Interaktionsräumen ergibt sich somit folgendes Bild (siehe Abb. 2.5): Die Kommunikation und die gemeinsame Konstruktion erfolgen auf Grundlage des gemeinsamen Wissens (in der Abbildung mit (a) gekennzeichnet). Dieses ist einerseits der *Common Ground* darüber, was die Aufgabe ist, wie diese zu bearbeiten ist etc., und andererseits die *Situational Awareness* über das aktuelle Tun der anderen. Basierend auf diesem Wissen erfolgt die Kom-

[14] Der Wahl des Interaktionsortes aufgrund der unterschiedlichen Persistenz geht konform mit den in Kapitel 2.4 angeführten Befunden der Studie von Whittaker et al. (1991). Während sich hier die Interaktionsbereiche durch unterschiedliche Persistenz auszeichneten und die Teilnehmer den adäquaten Ort wählten, kontrollierten dort die Teilnehmer selbst die Persistenz, indem sie nicht mehr gebrauchte Informationen aus dem Arbeitsbereich löschten, um Platz für neue zu schaffen.

munikation, wobei diese wiederum das gemeinsame Wissen erweitert. Die Kommunikation kann sowohl direkt durch den sprachlichen Austausch erfolgen (Pfeil (b)), wie auch indirekt mittels der Artefakte (Pfeil (c)), indem die Interaktionen der anderen mit den Artefakten oder zumindest deren Effekte beobachtet werden können. Die Artefakte haben dabei eine duale Funktion (Tatar et al., 1991): Sie sind einerseits Elemente *in* der Konversation und andererseits sind sie Objekte, *über* die gesprochen wird. Dies wird durch den Pfeil (d) in der Abbildung hervorgehoben. Hierfür ist es notwendig, dass diese Artefakte in der Kommunikation schnell und richtig identifiziert werden (Clark & Brennan, 1991).

Die Benutzungsschnittstellen dualer Interaktionsräume, in denen die Interaktionsbereiche in getrennten Bildschirmbereichen angesiedelt sind, zeichnen sich durch ein prinzipielles Problem aus: van Bruggen et al. (2002) weisen darauf hin, dass die Konstruktion einer externen Repräsentation bereits einen hohen *cognitive Load* verursacht. Entsprechend der *Cognitive Load Theory* erhöht sich die kognitive Beanspruchung im dualen Interaktionsraum, da in diesem die von den Mitlernenden gesendeten Informationen an verschiedenen Orten – dem Chat und dem Konstruktionswerkzeug – erscheinen und somit mental integriert werden müssen. Folge ist der *split-attention Effect*, d. h. die Aufmerksamkeit muss permanent mit einem hohen kognitiven Aufwand zwischen den räumlich getrennten Bereichen wechseln. Einen Ausweg aus diesem Problem sehen van Bruggen et al. in der Entwicklung von Schnittstellen, die die zu integrierenden Elemente auch visuell integrieren.

2.5.2 Anforderungen an Integrationsmaßnahmen

Welcher Bedarf besteht nun für eine verbesserte softwaretechnische Integration? Wobei und wie müssen Kollaborierende in dualen Interaktionsräumen besonders unterstützt werden? Hier lassen sich drei Problemfelder unterscheiden:

- Das prominenteste Problemfeld im Zusammenhang mit der Chatkommunikation ist die gerade erwähnte Bezugnahme auf Artefakte im gemeinsamen Arbeitsbereich.

- Zum Zweiten bedarf es einer weitergehenden Unterstützung bei der Koordination der sich über die beiden Interaktionsbereiche erstreckenden Aktivitäten.

- Drittens muss für die Erleichterung der Reflexion über den Kollaborationsverlauf die Trennung der Interaktionshistorien pro Interaktionsbereich überwunden werden.

Diese drei Problemfelder sollen nun im Einzelnen beleuchtet werden.

Deixis in dualen Interaktionsräumen

Im normalen Gespräch werden für die Identifizierung von Objekten der gemeinsamen Umgebung oftmals Zeigegesten eingesetzt (Bekker et al., 1995). Eine Geste mag zwar scheinbar redundant zum Gesagten sein (z. B. kann bereits „Sieh mal, die schöne Kirche dort!“ in der Situation alle relevanten Informationen enthalten, um die Kirche zu identifizieren), sie vereinfacht jedoch das Kommunizieren. Die Präsentation der Informationen über „mehrere Kanäle“, also sowohl verbal als auch visuell durch Gesten, macht ein korrektes Verstehen wahrscheinlicher (McCarthy & Monk, 1994).

Visuelle Ausdrucksmöglichkeiten wie Gesten stehen bei der Verwendung eines textuellen Kommunikationsmediums jedoch nicht zur Verfügung. Auch die zum Teil in kollaborativen Konstruktionswerkzeugen verfügbaren Telepointer (Stefik et al., 1987a) – dies sind spezielle, für alle sichtbare, nutzerspezifische Markierungen, deren Verschieben in nahezu Echtzeit übertragen wird – bieten in Verbindung mit dem Kommunikationsmedium Chat kaum Abhilfe (Dillenbourg & Traum, 2006). Der Grund liegt in der quasi-synchronen Natur des Mediums. Wie in Kapitel 2.3 ausgeführt, erfolgt das Lesen eines Chatbeitrags zeitlich versetzt zur Produktion, der Autor einer Nachricht hat kaum Einfluss darauf, wann diese von den anderen gelesen wird. Die erläuternde Geste mittels eines Telepointers ist somit kaum mit dem Lesen der Nachricht zu koordinieren.

Die Schwierigkeiten bei der Bezugnahme auf die Objekte im gemeinsamen Arbeitsbereich zeigten sich auch in den oben beschriebenen Studien. So fordern Suthers et al. im Fazit ihrer Untersuchung mit dem Belvédère-System:[15]

> Designers of online learning environments are advised to seek more natural means of referencing the contents of shared representations, particularly in conjunction with verbal communication. For example, chat or discussion tools might be designed to enable easy insertion of visual references to elements of other representations being discussed.
>
> (Suthers et al., 2003)

Analog zur Anforderung **C 2**, nach der eine explizite Bezugnahme aus dem Chat auf andere Chatbeiträge möglich sein muss, ergibt sich hier die Forderung, dass bei der Kombination von Chat mit einem Konstruktionswerkzeug die explizite Referenzierung von Elementen des gemeinsamen Arbeitsbereichs unterstützt werden muss:

I 1. Für die Integration muss die explizite Bezugnahme aus dem Chat auf den gemeinsamen Arbeitsbereich bereitgestellt werden.

Hierbei lassen sich zwei verschiedene „Gesten“ unterscheiden: Zum einen das Zeigen auf einzelne Objekte, um deren referentielle Identität herzustellen, wobei

[15] Auch van Bruggen (2003) bemängelt in Auswertung seiner Studien mit dem Belvédère-System die fehlende Verknüpfbarkeit von Chat und Konzeptgraphen.

die Objekte durch die Spezifika des Konstruktionswerkzeuges gegeben sind (z. B. sind dies in einem *Shared Whiteboard* die geometrischen Objekte, in Belvédère die Knoten und Kanten des Konzeptgraphen). Zum anderen bedarf es aber auch einer Möglichkeit, auf bestimmte Details bzw. Ausschnitte der gemeinsamen Arbeitsumgebung zu verweisen, z. B. um räumliche Relationen der Objekte zueinander oder Einzelheiten der Objekte zu identifizieren. Hieraus resultiert als Anforderung an die Integration:

I 2. Die explizite Referenzierung muss auf werkzeugspezifische Objekte wie auch auf Ausschnitte des Arbeitsbereichs möglich sein.

Dabei muss jedoch der dynamische Charakter der Artefakte beachtet werden. Zwischen dem Zeitpunkt, an dem ein Teilnehmer in seinem Beitrag explizit auf den gemeinsamen Arbeitsbereich Bezug nimmt, und dem Zeitpunkt, an dem dieser Beitrag gelesen wird, kann sich der Zustand des gemeinsamen Arbeitsbereichs durch Manipulationen der Teilnehmer verändern und somit der eigentliche Kontext des Beitrags verloren gehen. Deutlich wird dies zum Beispiel im Falle des Löschens von Objekten. Verweist ein Teilnehmer in seinem Beitrag auf ein Objekt und wird dieses simultan oder anschließend durch konkurrierende Aktionen anderer Teilnehmer entfernt, so ist dieses beim späteren Lesen der Nachricht nicht mehr verfügbar. Bei der Anzeige einer expliziten Referenz muss demzufolge der ursprüngliche Kontext – der Zustand des gemeinsamen Arbeitsbereichs bei der Beitragserstellung – zugänglich gemacht werden. Dies kann zum Beispiel durch eine *translucent History* (Genau & Kramer, 1995) erfolgen, bei der die ältere Version des Arbeitsbereichs halbtransparent „durchscheint“. Als Anforderung folgt somit:

I 3. Der Mechanismus zur expliziten Referenzierung auf den gemeinsamen Arbeitsbereich muss den Artefaktzustand bei der Bezugnahme erfassen und beim Lesen verfügbar machen.

Koordination der verteilten Aktivitäten

Bei der obigen Diskussion des Telepointer-Ansatzes wurde bereits kurz diskutiert, welche Schwierigkeiten sich aus der Verbindung des quasi-sychronen Mediums Chat mit einem WYSIWIS-System, bei dem die Manipulationen an den Artefakten in nahezu Echtzeit übertragen werden (siehe Kap. 2.4.1), ergeben. Diese „Inkompatibilität“ macht sich auch bei einem anderen Aspekt der Kollaboration in dualen Interaktionsräumen bemerkbar, nämlich der Koordination von direkter sprachlicher Kommunikation (Pfeil (b) in Abb. 2.5 auf S. 37) und indirekter Kommunikation (Pfeil (d)) über die Artefakte.

Während im Medium Chat das zeitlich versetzte Rezipieren eines Beitrags technisch bedingt ist und durch das persistente Chattranskript unterstützt wird (siehe S. 27), basiert der WYSIWIS-Ansatz auf der Annahme, dass die Manipulationen

von den anderen Teilnehmern auch simultan beobachtet werden. Während im Chattranskript die einzelnen Aktionen (die Beiträge) in ihrer Reihenfolge dokumentiert sind, zeigt der gemeinsame Arbeitsbereich des Konstruktionswerkzeugs nur den aus allen Aktionen resultierenden Artefaktzustand an; die einzelnen Aktionen selbst sind jenseits des Ausführungszeitpunktes nicht direkt ersichtlich.

Im Abschnitt 2.5.1 wurde darauf hingewiesen, dass sich die Kommunikation in dualen Interaktionsräumen über die beiden Interaktionsbereiche erstreckt. Die Aktionen im Konstruktionswerkzeug können intentionale kommunikative Akte sein; sie können Reaktionen auf Chatbeiträge sein. Auf Aktionen im Konstruktionswerkzeug können die Teilnehmer wiederum mit Chatbeiträgen reagieren.

Um nun den Teilnehmern ein verzögertes „Lesen“ der Aktionen im Konstruktionswerkzeug zu ermöglichen, müssen diese in ihrer zeitlichen Relation zu den Chatbeiträgen dokumentiert werden. Hierfür bietet sich das Chattranskript als Dokumentation der kommunikativen Beiträge in ihrer zeitlichen Abfolge an. Daraus folgt als Anforderung an die Integration:

I 4. Aktionen an den gemeinsamen Artefakten im Konstruktionswerkzeug müssen im Chattranskript „nachlesbar“ sein.

Da die Reihenfolge, in welcher die verschiedenen kommunikativen Akte dargeboten werden, für die Interpretation wichtige Informationen tragen, ist es notwendig, dass alle Teilnehmer die Beiträge im Chat und die Aktionen an den gemeinsamen Artefakten im Konstruktionswerkzeug in derselben Abfolge präsentiert bekommen. Somit muss für die Integration von Chat und Konstruktionswerkzeug gefordert werden:

I 5. Für alle Teilnehmer muss die Abfolge aller Chatbeiträge und Aktionen an den gemeinsamen Artefakten im Konstruktionswerkzeug identisch sein.

Die in dualen Interaktionsräumen gegebene Möglichkeit, Beiträge zur intentionalen Kommunikation nicht nur im Chat, sondern auch im gemeinsamen Arbeitsbereich zu erzeugen, hat auch Einfluss auf die Koordination der Beitragsproduktion. In der Diskussion der Anforderungen an das Chatwerkzeug wurde auf die Rolle der Aktivitäts-*Awareness* eingegangen, welche den Teilnehmern hilft, ihre simultanen Beitragsproduktionen zeitlich abzustimmen. In dualen Interaktionsräumen gilt es nun darüber hinaus, die Beitragsproduktionen im Chat mit denen im gemeinsamen Arbeitsbereich zu koordinieren. Hierzu müssen demzufolge den Nutzern *Awareness*-Informationen über aktuelle Eingaben der anderen präsentiert werden. Diese Präsentation sollte nach van Bruggen et al. (2002) visuell integriert[16] erfolgen, woraus als Anforderung folgt:

[16] Die visuelle Integration kann z. B. erreicht werden, indem bei der Darstellung der verschiedenen Informationen die Gestaltgesetze (Wertheimer, 1923) berücksichtigt werden.

I 6. Informationen über aktuelle Aktivitäten in den beiden Interaktionsbereichen müssen visuell integriert präsentiert werden.

Unterstützung der Reflexion

In den Anforderungen **C 3** und **K 3** wurden zur Unterstützung der Reflexion persistente Historien über die Aktivitäten in den jeweiligen Interaktionsbereichen gefordert. Die Historie des Chats ist die zeitlich geordnete Liste der Beiträge, die Historie der gemeinsamen Konstruktion ergibt sich aus der Sequenz der diskreten Operationen an den gemeinsamen Artefakten (siehe S. 34). Indem der Verlauf der Diskussion und die Konstruktionshistorie dauerhaft verfügbar gemacht werden, wird die Analyse des Diskursverlaufs sowie der Artefaktkonstruktion erleichtert.

Wie sich jedoch gezeigt hat, können der sprachliche Diskurs und die Aktionen zur Konstruktion der Artefakte eng aufeinander bezogen sein. Der Diskurs im Chat kann sich auf Objekte im gemeinsamen Arbeitsbereich beziehen, Konstruktionsschritte können in Reaktion auf den Chatdiskurs erfolgen etc. Für das Nachvollziehen des Kollaborationsverlaufs bedarf es demzufolge einer *kombinierten* Historie, die die zeitliche Abfolge der Aktivitäten in beiden Interaktionsbereichen berücksichtigt. Für die Integration dualer Interaktionsräume gilt damit als Anforderung:

I 7. Es muss eine kombinierte Historie der Aktivitäten in beiden Interaktionsbereichen verfügbar gemacht werden.

Wie kann diese kombinierte Historie nun aber für die Reflexion verwendet werden? Vollständige Interaktionshistorien werden genutzt, um die Sitzungen im *Nachhinein* wie einen Film abspielen zu können (siehe z. B. Landsman, 2006; Tang, 2003). Hierbei wird für jeden Zeitpunkt der Gesamtzustand der Kollaborationsumgebung rekonstruiert. Es werden also genau die Chatbeiträge angezeigt, die bis zu diesem Zeitpunkt empfangen wurden, und der Arbeitsbereich wird in den zu diesem Zeitpunkt gültigen Zustand zurück versetzt.

Für die Unterstützung der Reflexion *während* der Kollaboration ist es jedoch nicht notwendig, diese vollständigen älteren Gesamtzustände wiederherzustellen. Dies hieße nämlich zum Beispiel, dass die nach dem angezeigten Zustand empfangenen Chatbeiträge ausgeblendet werden müssten. Wichtiger für die Reflexion *während* der Kollaboration ist es vielmehr, die Kontextbeziehungen zwischen den Aktivitätshistorien der beiden Interaktionsbereiche verfügbar zu machen.

Dies betrifft zum einen den Artefaktkontext zum sprachlichen Diskurs. Im Chattranskript ist der vollständige Diskursverlauf dokumentiert. Für das Verständnis älterer Beiträge kann es jedoch notwendig sein, den zugehörigen Zustand des gemeinsamen Arbeitsbereiches zu rekonstruieren, da dieser als Kontext für die Kommunikation genutzt wurde. Duale Interaktionsräume müssen also folgender Anforderung genügen:

I 8. Die Rekonstruktion des Artefaktkontextes zu beliebigen Chatbeiträgen muss ermöglicht werden.

Dieselbe Argumentation trifft zum anderen auch für jeden entsprechend Anforderung **K 3** verfügbaren älteren Artefaktzustand zu. Der gemeinsame Arbeitsbereich erlaubt es, alle vergangenen Zustände zu rekonstruieren. Um nun aber nachzuvollziehen, welcher sprachliche Austausch im Kontext dieses älteren Zustands erfolgte, ob es z. B. im Chat Erläuterungen oder Diskussionen zu dem diesen Zustand erzeugenden Konstruktionsschritt gab, muss im Chattranskript die zugehörige Sequenz identifiziert werden. Hierzu müssen duale Interaktionsräume geeignete Funktionalitäten bereitstellen und somit die Anforderung erfüllen:

I 9. Die Identifikation des Chatdiskurses zu beliebigen älteren Artefaktzuständen muss erleichtert werden.

2.6 Fazit

Beim computervermittelten kollaborativen Lernen werden die Interaktionen zwischen den Lernenden und das daraus resultierende Kommunikationsverhalten durch die medialen Eigenschaften der Kollaborationsumgebung beeinflusst. Die medialen Eigenschaften resultieren einerseits aus den Merkmalen des zugrunde liegenden Kommunikationskanals, also seiner zeitlichen Charakteristika und der Granularität der übertragenen Aktionen, und andererseits aus den Affordanzen, Visualisierungen und Interaktionsmöglichkeiten der Benutzungsschnittstelle.

Besteht eine Kollaborationsumgebung aus einem Kommunikationsmedium und einem gemeinsamen Arbeitsbereich zur Konstruktion und Bearbeitung gemeinsam genutzter Artefakte, so stehen für die synchrone Kollaboration zwei Interaktionsbereiche zur Verfügung. Die Teilnehmer nutzen diese flexibel für die Bearbeitung der Aufgabe und die Kommunikation miteinander. In der kollaborativen Nutzung werden diese beiden Bereiche durch die Teilnehmer nicht nur simultan, sondern auch integriert genutzt: Eine Aktion in dem einen Bereich kann Reaktionen in dem anderen Bereich nach sich ziehen. Die über die beiden Bereiche verteilten, aber aufeinander bezogenen und abgestimmten Aktivitäten, seien es das Versenden eines Chatbeitrags oder eine Modifikation der Artefakte, sind Grundlage der gemeinsamen Wissenskonstruktion.

Wie die Analyse der empirischen Befunde und theoretischen Ansätze gezeigt hat, können die mediale Gestaltung und der Funktionsumfang der Kollaborationsumgebung jedoch Barrieren aufbauen, die eine integrierte Nutzung in der Kollaboration erschweren. Problematisch an dualen Interaktionsräumen sind insbesondere die deiktische Bezugnahme auf Objekte in der gemeinsamen virtuellen Arbeitsumgebung, die Koordination der simultan möglichen Aktivitäten sowie das Nachvollziehen des sich über beide Interaktionsbereiche erstreckenden Kollaborationsverlaufs. Hieraus wurden neun Anforderungen (I1 bis I9) an Integrati-

onsmaßnahmen für duale Interaktionsräume abgeleitet, die beim Überwinden der Barrieren helfen sollen. Des Weiteren wurden Minimalanforderungen für virtuelle Kollaborationsumgebungen (A1 bis A3), deren Werkzeug zur Chatkommunikation (C1 bis C4) sowie deren Werkzeug zur kollaborativen Konstruktion (K1 bis K3) aufgestellt, die für den sinnvollen Einsatz beim kollaborativen Lernen erfüllt sein müssen. Tabelle 2.2 fasst diese nochmals zusammen.

Im nächsten Kapitel werden verwandte Forschungsansätze vorgestellt und hinsichtlich dieser Anforderungen untersucht.

Tabelle 2.2: Alle Anforderungen im Überblick zusammen mit der Anforderungsnummer (linke Spalte) sowie der Seite, auf der sie aufgestellt wurde

Nr.	Anforderung	S.
Minimalanforderungen an Kollaborationsumgebungen:		
A 1	Die Kollaborationsumgebung muss ein Kommunikationsmedium für den sprachlichen Diskurs bereitstellen.	18
A 2	Die Kollaborationsumgebung muss Werkzeuge zur Anzeige und kollaborativen Bearbeitung aufgabenspezifischer Artefakte bereitstellen.	19
A 3	Die Kollaborationsumgebung muss die Reflexion in den Interaktionen und als Analyse des Kollaborationsverlaufs unterstützen.	19
Minimalanforderungen an Chatwerkzeuge:		
C 1	Das Chatwerkzeug muss konsistente und vollständige Chattranskripte bereitstellen.	26
C 2	Das Chatwerkzeug muss die explizite Bezugnahme auf einen oder mehrere andere Beiträge bzw. Beitragsbestandteile ermöglichen.	27
C 3	Das Chattranskript muss systemweit persistent verfügbar sein.	27
C 4	Den Teilnehmern müssen *Awareness*-Informationen über simultane Beitragserstellungen anderer Teilnehmer zur Verfügung gestellt werden.	28
Minimalanforderungen an kollaborative Konstruktionswerkzeuge:		
K 1	Konstruktionswerkzeuge müssen den Zustand der Artefakte für alle Teilnehmer konsistent halten.	34
K 2	Konstruktionswerkzeuge müssen *Awareness*-Informationen bereitstellen.	34
K 3	Der Konstruktionsverlauf muss systemweit persistent verfügbar sein.	34
Anforderungen an integrierte Interaktionsräume:		
I 1	Für die Integration muss die explizite Bezugnahme aus dem Chat auf den gemeinsamen Arbeitsbereich bereitgestellt werden.	39
I 2	Die explizite Referenzierung muss auf werkzeugspezifische Objekte wie auch auf Ausschnitte des Arbeitsbereichs möglich sein.	40
I 3	Der Mechanismus zur expliziten Referenzierung auf den gemeinsamen Arbeitsbereich muss den Artefaktzustand bei der Bezugnahme erfassen und beim Lesen verfügbar machen.	40
I 4	Aktionen an den gemeinsamen Artefakten im Konstruktionswerkzeug müssen im Chattranskript „nachlesbar“ sein.	41
I 5	Für alle Teilnehmer muss die Abfolge aller Chatbeiträge und Aktionen an den gemeinsamen Artefakten im Konstruktionswerkzeug identisch sein.	41
I 6	Informationen über aktuelle Aktivitäten in den beiden Interaktionsbereichen müssen visuell integriert präsentiert werden.	42
I 7	Es muss eine kombinierte Historie der Aktivitäten in beiden Interaktionsbereichen verfügbar gemacht werden.	42
I 8	Die Rekonstruktion des Artefaktkontextes zu beliebigen Chatbeiträgen muss ermöglicht werden.	43
I 9	Die Identifikation des Chatdiskurses zu beliebigen älteren Artefaktzuständen muss erleichtert werden.	43

3 Verwandte Ansätze

Im Kapitel 2 wurden die Probleme analysiert, die ein reines Nebeneinander von einem Chatwerkzeug zur textuellen Kommunikation und einem Konstruktionswerkzeug zur kollaborativen Bearbeitung gemeinsamer Artefakte in einem dualen Interaktionsraum mit sich bringen, und Anforderungen an eine verbesserte Integration der beiden Werkzeuge formuliert.

In diesem dritten Kapitel werden verwandte Arbeiten und Ansätze vorgestellt. Abschnitt 3.1 stellt die Kriterien vor, anhand derer die Ansätze ausgewählt wurden, und liefert eine Kategorisierung, mit der die relevanten Ansätze gruppiert werden können. Anschließend werden prototypische Vertreter der einzelnen Kategorien vorgestellt.

Dieses Kapitel schließt mit einer Bewertung der verwandten Arbeiten entsprechend der für eine verbesserte Integration dualer Interaktionsräume zu erfüllenden Anforderungen.

3.1 Auswahlkriterien und Kategorisierung

Diese Arbeit beschäftigt sich mit der Frage, wie duale Interaktionsräume für das synchrone, örtlich verteilte kollaborative Lernen besser integriert werden können, um die durch die Separation der Interaktionen hervorgerufenen Probleme bei der Kommunikation, Koordination und Reflexion zu bewältigen. Diese Frage wurde bisher in der CSCL-Forschung – abgesehen vom Aufzeigen der Defizite mangelnder Integration, wie in Kapitel 2.5 ausgeführt – nicht explizit adressiert. Ein wesentlicher Bereich in der Forschung zum computergestützten kollaborativen Lernen ist jedoch, geeignete Softwareumgebungen zu entwickeln, mit denen Gruppen kollaborativ lernen können. Aus der Fülle der Forschungsarbeiten sind für diese Arbeit diejenigen von Interesse, welche

1. auf die synchrone und örtlich verteilte Kollaboration zielen,
2. die textuelle Kommunikation und
3. die Erstellung aufgabenrelevanter Artefakte ermöglichen.

In Kapitel 2.2 wurde der Aufbau dualer Interaktionsräume vorgestellt: Sie bestehen aus einem Werkzeug für den textuellen Diskurs – dem Diskursraum – und einem Werkzeug zur gemeinsamen Konstruktion aufgabenrelevanter Artefakte – dem Artefaktraum. Die erste Kategorie verwandter Systeme umfasst demzufolge jene, die dieses **Nebeneinander von Diskurs- und Artefaktraum** umsetzen. Diese werden in Kapitel 3.2 daraufhin analysiert, inwieweit sie die in Kapitel 2.5.2 identifizierten Problemfelder dualer Interaktionsräume – die *deiktischen Bezugnahmen*, die *Koordination der simultanen Aktivitäten* und die *Reflexion über den Kollaborationsverlauf* – adressieren und die Anforderungen an eine verbesserte Integration erfüllen.

Neben diesem „Standardansatz“ der Kombination von Chat- und Konstruktionswerkzeug existieren prinzipiell zwei weitere Varianten, den Diskurs- mit dem Artefaktraum zu integrieren:

- Die erste besteht in der **Einbettung des Diskurses in den Artefaktraum**. Hierbei wird die Möglichkeit zur textuellen Kommunikation in den Artefaktraum eingebettet.
- Die andere Variante besteht umgekehrt in der **Einbettung der Artefakte in den Diskursraum**: die Artefakte werden in die textuelle Kommunikation integriert, indem die Chatbeiträge nicht nur aus Text bestehen, sondern z. B. auch Skizzen oder mathematischen Formeln eingefügt werden können.

In den Kapiteln 3.3 und 3.4 werden diese beiden Varianten genauer betrachtet und geprüft, inwieweit sie Alternativen zum integrierten Nebeneinander von Diskurs- und Artefaktraum darstellen und Lösungen für die genannten Problemfelder bieten. Das Kapitel schließt mit einer Bewertung der verwandten Arbeiten hinsichtlich der Anforderungen an eine verbesserte Integration.

3.2 Nebeneinander von Diskurs- und Artefaktraum

Das Nebeneinander von Diskurs- und Artefaktraum in Chat- und Konstruktionswerkzeug ist die naheliegende Lösung, um sowohl den Diskurs wie auch die Erstellung und Bearbeitung von Artefakten in synchronen und örtlich verteilten Lernszenarien zu ermöglichen.

Abbildung 3.1 zeigt die Kollaborationsumgebung *GrewpEdit* (Langton et al., 2004; Granville & Hickey, 2005) für die kollaborative Softwareentwicklung. *GrewpEdit* wurde für den Einsatz in der Informatiklehre konzipiert. Die Benutzungsschnittstelle teilt sich im Wesentlichen in vier Bereiche auf: ein Chatfenster für die textuelle Kommunikation, ein Texteditor für die gemeinsame Bearbeitung des Programmcodes, eine Anwesendenliste und ein Bereich, in welchem die Ausgaben der jeweils lokalen Compiler aller Teilnehmer eingesehen werden können. Im gemeinsamen Texteditor sind die Cursorpositionen aller Teilnehmer durch farbliche Hervorhebungen markiert. Dieselbe Zuordnung von Farben zu Teilnehmern findet sich auch in der Chathistorie, in der Anwesendenliste und in den Fenstern für die Compilerausgaben.

GrewpEdit ist ein typisches Beispiel für *nicht*integrierte Interaktionsräume. Bis auf die bereichsübergreifenden Farbmarkierungen gibt es keinerlei Unterstützungsmaßnahmen, um die über die Interaktionsbereiche verteilten Aktionen der

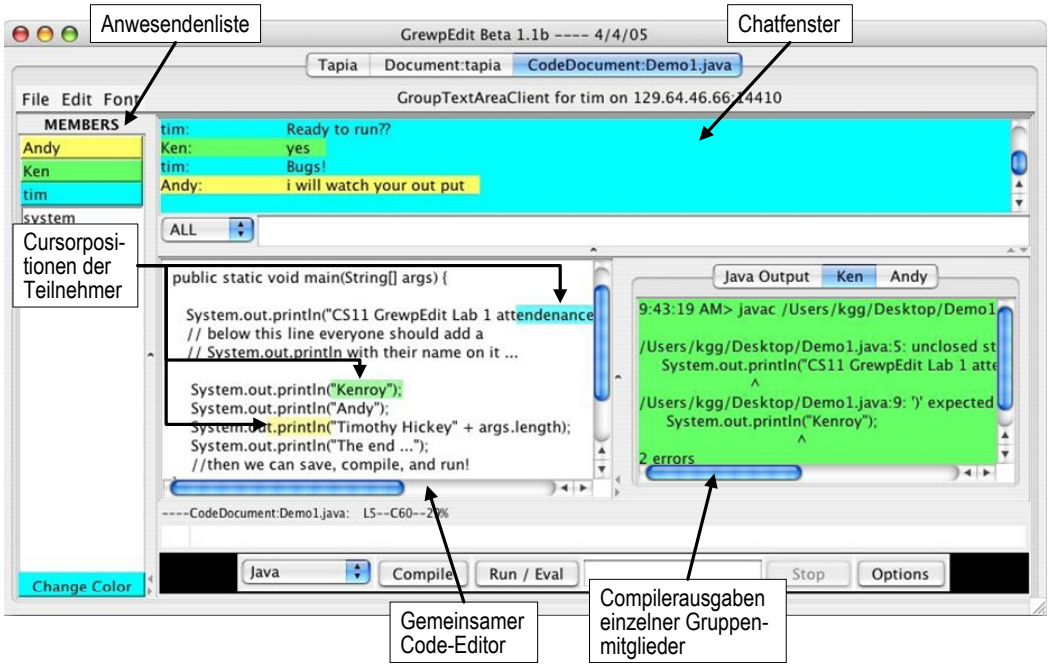


Abbildung 3.1: *GrewpEdit* – ein kollaborativer Java-Editor (Screenshot entnommen Granville & Hickey, 2005)

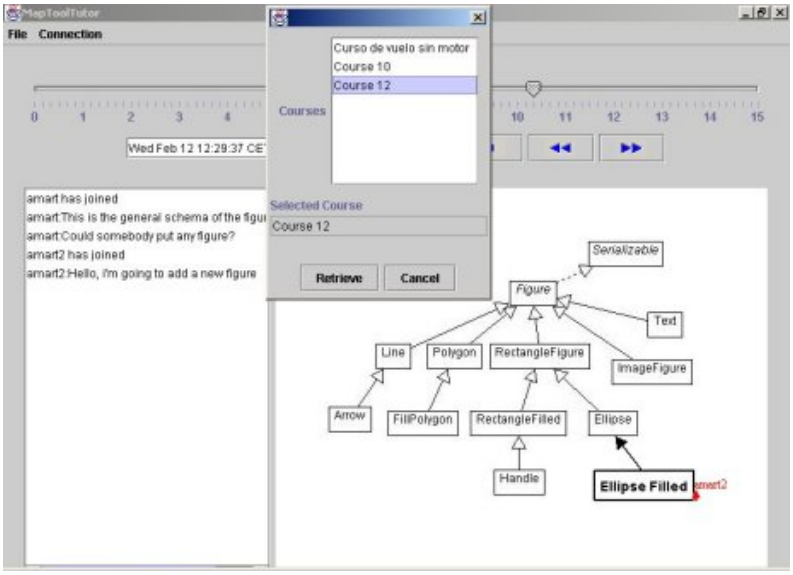


Abbildung 3.2: *MaptoolTutor*, mit dem aufgezeichnete *MapTool*-Sitzungen abgespielt werden können

Teilnehmer zu verknüpfen. Es gibt kaum Unterstützung für die deiktische Bezugnahme. Die farbliche Markierung der Cursor-Positionen im gemeinsamen Texteditor kann zwar im Sinne eines Tele-Pointers genutzt werden. Da die Markierung jedoch flüchtig ist, gehen Verweise in den Chatbeiträgen auf die Position des Cursors nach einem Umsetzen des Cursors ins Leere. Auch wird die Koordination simultanen Agierens nicht durch zusätzliche Funktionalitäten unterstützt.

GrewpEdit gehört wie z. B. auch das *Group Homework Tool* (Langton et al., 2004) und *VesselWorld* (Feinman, 2006) zu einer Familie von *Groupware*-Applikationen, die an der Brandeis-University in Waltham, USA, entwickelt wurde. Für diese existiert ein Protokollierungssystem, welches alle Aktionen während einer Sitzung aufzeichnet (Landsman, 2006). Diese Aufzeichnungen können später genutzt werden, um die Sitzungen wie einen Film abzuspielen. Somit steht zwar prinzipiell für diese Applikationen eine vollständige Interaktionshistorie zur Verfügung, sie kann jedoch nur nachträglich durch ein eigenständiges Werkzeug wiedergegeben werden und ist für die Teilnehmer während der Sitzung nicht verfügbar.

Ein analoger Ansatz wurde für das *MapTool* umgesetzt. Dieses auf dem *Collaborative ANTS System*[1] der Universität von Murcia, Spanien, basierende Werkzeug kombiniert ein *Shared Whiteboard* mit einem Chat. Jegliche Aktion in einer Sitzung wird durch das System aufgezeichnet und kann anschließend mit einem eigenen Werkzeug, dem *MaptoolTutor* (siehe Abb. 3.2) abgespielt werden. Wie der Name jedoch nahelegt, ist auch dies eher auf die nachträgliche Analyse durch Tutoren oder Forscher ausgerichtet und weniger für die Unterstützung der Reflexion in-

[1] Siehe `http://ants.dif.um.es/cscl/`.

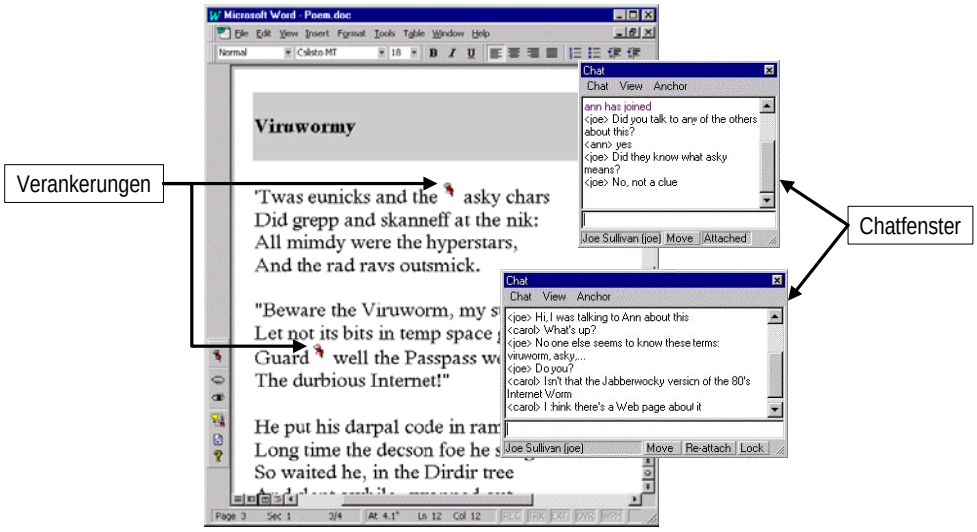


Abbildung 3.3: *Anchored Conversations*: Zu jeder beliebigen Stelle eines Word-Dokumentes (markiert durch ein Stecknadelsymbol) kann ein separater Chat angelegt werden (aus Churchill et al., 2000)

nerhalb der Lerngruppe gedacht, zumal die Historie nicht während der Sitzung bereitsteht.

Die vorgestellten Systeme erfüllen keine der Anforderungen an eine Integration von Chat- und Konstruktionswerkzeug: Die deiktische Bezugnahme, die Koordination der simultanen Aktionen in den kombinierten Interaktionsräumen und die Reflexion über den Kollaborationsverlauf werden nicht durch Integrationsmaßnahmen unterstützt.

In den folgenden beiden Kapiteln sollen nun die anderen beiden Varianten der Integration von Diskurs- und Artefaktraum beleuchtet werden.

3.3 Einbettung des Diskurses in den Artefaktraum

Ein anderer Ansatz, den textuellen Diskurs und die Bearbeitung von Artefakten zu ermöglichen, liegt in der Einbettung der Kommunikation in den Artefaktraum. Eine erste Variante ist durch das in Abbildung 3.3 gezeigte *Anchored Conversations*-Werkzeug (Churchill et al., 2000) realisiert. Mit diesem können Chatdiskurse an jede beliebige Stelle eines Word-Dokuments „angeheftet“ werden. Die Diskurse sind hierbei persistent und über das Dokument wie auch separat abrufbar. Das Werkzeug unterstützt die Verteilung der Dokumente, so dass die Kollaborationspartner jeweils eine lokale Kopie erhalten, wobei diese dann jedoch nicht gekoppelt sind: lokale Änderungen an den Dokumenten werden nicht an die anderen Kollaborationspartner übermittelt.

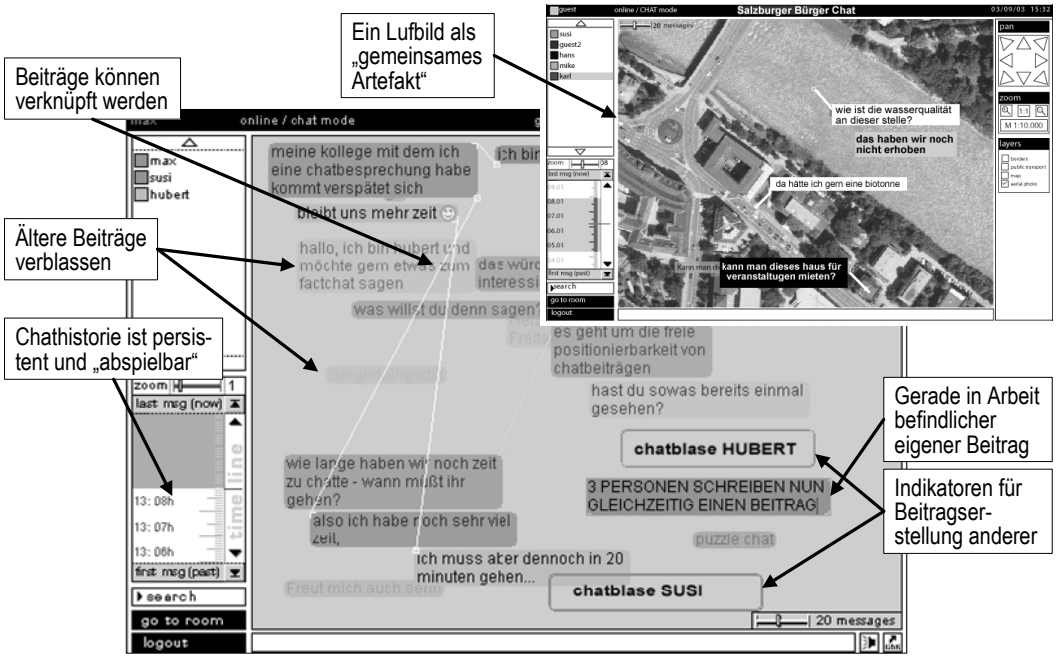


Abbildung 3.4: Benutzungsschnittstelle des *FactChat*-Systems (aus Harnoncourt et al., 2005)

Das *Anchored Conversations*-Werkzeug folgt der Metapher der „Klebezettel“ (*Sticky Notes*): An das Dokument können Zettel geklebt werden, auf denen dann jeweils ein Chat zu einer spezifischen Textstelle möglich ist. Durch die Verankerung der Chats im Dokument wird deren Kontext bewahrt. Jenseits der Verankerung der Chat-„Zettel“ werden jedoch deiktische Bezüge nicht unterstützt.

Die Word-Dokumente selbst sind nicht kollaborativ editierbar. Da aber innerhalb des Dokuments verschiedene Chats möglich sind, bilden diese einen „multiplen“ Interaktionsraum. Die verschiedenen Chats im Dokument sind allerdings völlig unabhängig voneinander; ob auf einem anderen Chat-„Zettel“ als dem gerade fokussierten kommuniziert wird, ist nicht erkennbar. Die einzelnen Diskurse sind somit fragmentiert. Dies trifft auch für die persistenten Historien zu. Obwohl der Inhalt der einzelnen Chat-„Zettel“ dauerhaft gespeichert wird und somit alle Aktionen im „multiplen“ Interaktionsraum persistent verfügbar sind, ist die zeitliche Abfolge über die Chats hinweg kaum nachvollziehbar.

Dies wird jedoch in einer weitergehenden Variante der Einbettung vom Diskurs in den Artefaktraum ermöglicht, dem *FactChat*-System (Harnoncourt et al., 2005, siehe Abb. 3.4). In diesem hängt man seinen Chatbeitrag nicht an das Ende einer zeitlich geordneten Beitragsliste, sondern positioniert ihn frei auf einer Chat-„Fläche“. Damit die Chat-„Fläche“ nicht irgendwann voller Beiträge ist, werden ältere Beiträge langsam ausgeblendet, sie verblassen mit der Zeit. Der gesamte Chatverlauf ist persistent, mit einem Schieberegler (links in der Abbildung) kann

durch die Vergangenheit gefahren werden. Die deiktische Bezugnahme wird im *FactChat*-System auf zweierlei Arten unterstützt. Zum einen können die Beiträge beliebig auf der Fläche positioniert und somit durch räumlich Nähe auch semantische Nähe ausgedrückt werden. Zum anderen erlaubt das System, eine explizite Verknüpfung zu einem anderen Chatbeitrag herzustellen (die weißen Linien in der Abbildung).

Die Chat-„Fläche“ kann nun auch zu einer gemeinsamen Arbeitsfläche erweitert und zur Darstellung gemeinsamer Artefakte genutzt werden. Der kleine Screenshot oben rechts in Abbildung 3.4 zeigt beispielsweise, wie eine Luftaufnahme als gemeinsames Artefakt für den Chat dient. Damit können durch die Positionierung der Chatbeiträge deiktische Bezüge auf das Artefakt ausgedrückt werden.

Im *FactChat*-System sind die Artefakte jedoch durch die Nutzer weder erzeugbar (das Hintergrundbild wird für den Chatraum festgelegt) noch während der Sitzung manipulierbar. Eine Erweiterung des Konzeptes auf erzeugbare Artefakte stellt das *GraffiDis*-System (Leponiemi, 2003) dar. In diesem können die Teilnehmer nicht nur Texte, sondern auch einfache Zeichnungen auf der Chat-„Fläche“ erzeugen. Diese verhalten sich analog zu den Chatbeiträgen: Einmal erzeugt können sie nicht mehr verändert werden und verblassen mit der Zeit. Insofern gestaltet es sich bei diesem Ansatz schwierig, gemeinsame aufgabenrelevante Artefakte zu erzeugen und zu bearbeiten. In Leponiemi (2003) wird denn auch berichtet: „users utilized the graphical elements only for strengthening the reference to the target annotation“. Somit wird die Möglichkeit zur Erzeugung graphischer Objekte von den Teilnehmern allein für die explizite Referenzierung genutzt. Dabei gehen die Ausdrucksmöglichkeiten zur Referenzierung über die Verknüpfung mit einem anderen Chatbeitrag – wie es im *FactChat*-System möglich ist – hinaus, da mit den graphischen Elementen auf beliebige Ausschnitte und Bereiche der gemeinsamen Arbeits- und Chatfläche referenziert werden kann.

Die drei vorgestellten Systeme in dieser Kategorie zeichnen sich alle dadurch aus, dass die Artefakte nicht manipulierbar sind. Inwieweit sind die Konzepte jedoch prinzipiell auf veränderbare Artefakte erweiterbar? Das Konzept des *Anchored Conversations*-Werkzeugs ist es, den Kontext eines Chats durch die Verankerung im Dokument herzustellen und zu bewahren. Wird nun jedoch die Textpassage, an die der Chat geheftet wird, in einem gemeinsam editierbaren Dokument verändert, so geht der ursprüngliche Chatkontext verloren. Die Metapher der Klebezettel stößt damit an ihre Grenzen: Nicht nur, dass die Klebezettel einen veränderlichen Inhalt haben (auf ihnen erfolgt der Chat), sondern auch das Dokument, an das sie geheftet werden, verändert sich. Somit stellen sich auch hier die gleichen Probleme wie beim Nebeneinander von Diskurs- und Artefaktraum: Wie können der Dokumentkontext zum Diskurs und umgekehrt der Diskurskontext zur Textmodifikation erhalten und nachvollziehbar bereitgestellt werden? Wie können die sich über Textdokument und Klebezettel verteilten Aktionen der Teilnehmer koordiniert werden? Wie kann im Diskurs auf Textpassagen (und die Diskurse auf anderen Klebezetteln!) verwiesen werden?

Wie sieht es für die anderen beiden Ansätze aus? Es ist naheliegend, die Werkzeuge um dauerhaft sichtbare und manipulierbare Objekte zu erweitern. Die zeitlichen Beziehungen zwischen Chatbeiträgen und Artefaktänderungen können dann jedoch nicht mehr anhand des Verblassens erkannt werden, denn die Artefakte sollen ja dauerhaft sichtbar bleiben. Somit ergeben sich hier analog zur Variante des Nebeneinanders von Diskurs- und Artefaktraum die Schwierigkeiten, wie diese zeitlichen Beziehungen (Anforderung **I 4**), wie die integrierte Interaktionshistorie (**I 7**) sowie der Artefakt- und Diskurskontext (**I 8** und **I 9**) für die Nutzer verfügbar gemacht werden können.

Damit reduzieren sich die Unterschiede zwischen der eingebetteten Variante zur Variante, Diskurs- und Artefaktraum nebeneinander zu präsentieren, auf zwei Aspekte:

1. Die Chatbeiträge im eingebetteten Falle verblassen, während sie im anderen Falle ausgeblendet werden. Jeweils durch Schieberegler kann durch die Historie gefahren werden.

2. Im eingebetteten Falle sind die Chatbeiträge frei positionierbar, ihre zeitliche Abfolge wird durch ihre Farben visualisiert, während im anderen Falle die Beiträge in einer Liste sortiert sind; die zeitliche Abfolge wird durch die räumliche Position zueinander dargestellt.

Welche der beiden Darstellungsvarianten für den Chatdiskurs die bessere ist, kann in dieser Arbeit nicht beantwortet werden. Die Einbettung von Diskurs- in den Artefaktraum liefert jedoch für sich genommen keine Lösungen für die Probleme dualer Interaktionsräume.

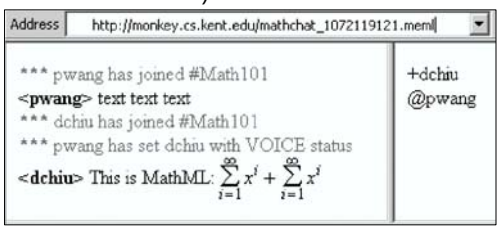
3.4 Einbettung der Artefakte in den Diskursraum

Nachdem im vorangegangenen Abschnitt gezeigt wurde, dass die Einbettung des Diskurses in den Artefaktraum keine Lösung für die Integrationsprobleme darstellt, soll nun die andere Variante – die Einbettung der Artefakte in den Diskursraum – geprüft werden. Die Ansätze in dieser Kategorie zeichnen sich dadurch aus, dass das Medium Chat erweitert wird um nicht-textuelle Beitragsbestandteile: Bei der Erstellung eines Chatbeitrags können die Nutzer je nach System unterschiedliche Objekte in den Beitrag einfügen.

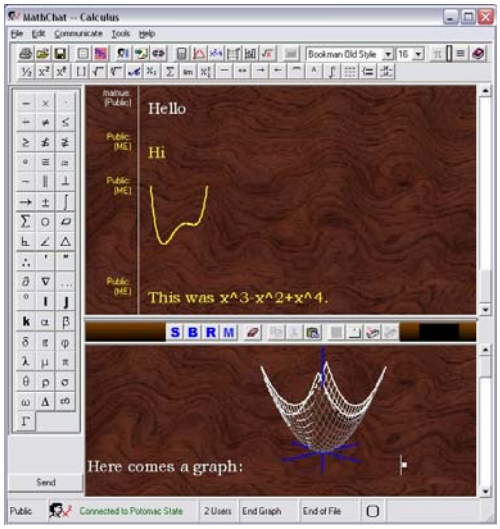
Abbildung 3.5.a zeigt einen Screenshot des *MathChat*-Systems (Chiu, 2004). Dieses zielt darauf, die Kommunikation über mathematische Fragestellungen zu erleichtern. Bei der Erstellung des Beitragstextes können mathematische Formeln mit einer speziellen Notation in den fortlaufenden Text eingegeben werden. Die Formeln werden dann im Chattranskript in der gewohnten Weise dargestellt. Gleiche Möglichkeiten bietet zum Beispiel auch der *CureChat*[2].

[2] Siehe `http://www.pi6.fernuni-hagen.de/DocCURE/manual.html`.

a) MathChat



b) MathChat – Electronic Abacus



c) MSN Messanger

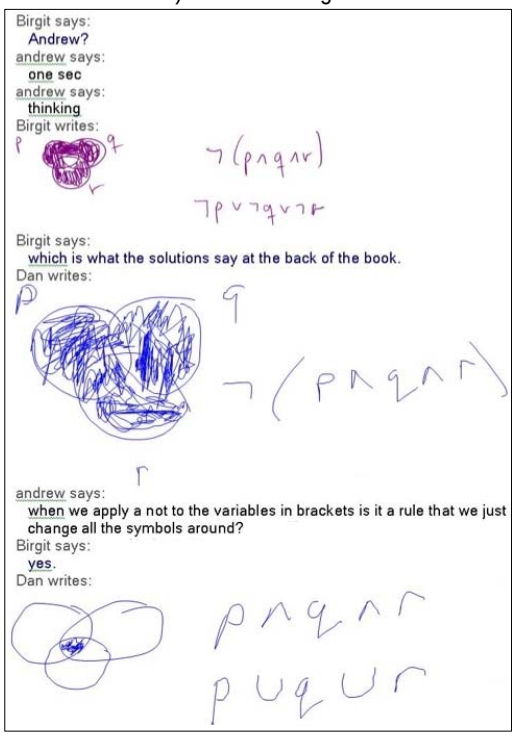


Abbildung 3.5: Einbettung mathematischer Objekte in den Chat: a) mathematische Formeln (MathChat, aus Chiu, 2004), b) komplexe mathematische Objekte, a) Freihandzeichnungen (aus Loch & McDonald, 2007)

In die gleiche Richtung zielt das ebenfalls *MathChat* genannte System von *Electronic Abacus Software*[3] (siehe Abb. 3.5.b). Es geht jedoch weit über die Möglichkeiten des erstgenannten Systems hinaus. Neben der Einbettung von Formeln erlaubt es z. B. auch das Einfügen von Graphen ein- bzw. zweidimensionaler Funktionen und bietet eine Reihe individuell nutzbarer Werkzeuge z. B. zur Symbolmanipulation und für einfache statistische Auswertungen. Die mathematischen Objekte im Chattranskript (z. B. ein von einem anderen Teilnehmer verschickter Graph) können in das eigene Eingabefenster kopiert, dort manipuliert und als Bestandteil einer neuen Nachricht wieder verschickt werden. Vergleichbar zu diesem System ist das in die *Mathyards*[4] integrierte Chat-Werkzeug.

Während die eben genannten Systeme spezifische mathematische Objekte unterstützen, bietet der MSN Messenger[5] die Möglichkeit, anstelle einer Textnach-

[3] Siehe `http://www.electricabacus.com/`.
[4] Siehe `http://www.mathyards.com/cp2/cp.html`.
[5] Siehe `http://get.live.com/messenger/overview`.

richt eine Freihandzeichnung zu verschicken. Abbildung 3.5.c zeigt den Auszug eines Chattranskripts, welches in einer mathematischen Übung unter Einsatz dieses Werkzeugs entstand (Loch & McDonald, 2007).

Das letzte Beispiel zeigt jedoch auch anschaulich den Nachteil dieses Ansatzes: Da die Artefakte jeweils in die Chatbeiträge eingebettet werden, entsteht durch jede Modifikation – die über *Copy & Paste* oder aber gar durch ein Neuzeichnen wie in Abbildung 3.5.c erfolgen muss – eine neue Instanz einer aufgabenrelevanten Repräsentation. Damit erübrigen sich zwar die Probleme dualer Interaktionsräume – eine Koordination simultaner Aktionen ist nicht notwendig; die Artefakthistorie ist bereits Bestandteil der Diskurshistorie. Das geht jedoch einher mit dem Fehlen einer gemeinsamen Repräsentation. Dieser Ansatz trägt auch nur für „kleine“ Artefakte, die im Diskurs eine illustrierende Funktion haben. Erfordert die Aufgabe jedoch umfangreichere Repräsentationen wie z. B. Mindmaps, dann ist dies über in die Chatbeiträge eingebettete Objekte kaum noch möglich.

Zu guter Letzt soll auf ein Defizit aller vorgestellten Systeme hingewiesen werden. Keines der Systeme unterstützt die deiktische Bezugnahme auf eingebettete Artefakte: Möchte ein Teilnehmer auf ein Objekt oder einen Ausschnitt eines Objekts in einem älteren Chatbeitrag verweisen, so muss dies mit mehr oder weniger hohem Aufwand rein sprachlich mit den bekannten Folgen drohender Ambiguität erfolgen.

3.5 Zusammenfassung und Bewertung

In diesem Kapitel wurden drei Varianten der Kombination von Diskurs- und Artefaktraum vorgestellt und exemplarische Vertreter jeder Kategorie daraufhin analysiert, inwieweit sie die in Kapitel 2 ausgeführten Probleme dualer Interaktionsräume adressieren.

Tabelle 3.1 fasst diese Analyse hinsichtlich der grundlegenden Anforderungen zusammen: Erlaubt das System den textuellen Diskurs (Anforderung **A 1**) und die kollaborative Bearbeitung gemeinsamer Artefakte (**A 2**)? Unterstützt das System die Teilnehmer bei der deiktischen Bezugnahme (**I 1**, **I 2**, **I 3**), der Koordination der simultanen Aktivitäten in den beiden Interaktionsbereichen (**I 4**, **I 5**, **I 6**) und der Reflexion über den Kollaborationsverlauf (**I 7**, **I 8**, **I 9**)?

Die einzigen Systeme, welche beide Minimalanforderungen – die Ermöglichung eines textuellen Diskurses und die kollaborative Bearbeitung gemeinsamer Artefakte – erfüllen, realisieren ein Nebeneinander von Diskurs- und Artefaktraum (siehe Kap. 3.2). Allerdings sind diese für die Kollaborierenden unverbunden; es wird keine der Integrationsanforderungen vollständig erfüllt.

Wie in Abschnitt 3.3 gezeigt, führt die Einbettung des Diskurses in den Artefaktraum nicht per se zu einer Lösung der Probleme dualer Interaktionsräume. Das einzige System in dieser Kategerie, welches die Bearbeitung gemeinsamer Artefakte in Ansätzen erlaubt, ist *GraffiDis*. Wird dieser Ansatz jedoch weitergedacht

hin zu gemeinsam manipulierbaren Artefakten, dann ergeben sich hier für die Koordination und Reflexion dieselben Schwierigkeiten wie in der ersten Variante.

Auch der in Abschnitt 3.4 vorgestellte umgekehrte Ansatz – die Einbettung der Artefakte in den Diskursraum – liefert keine befriedigende Lösung. Die größte Schwierigkeit dieser Variante liegt in der für die Nutzer nur sehr aufwendig herzustellenden gemeinsamen Artefakte, da jede Veränderung zu einer neuen Instanz des Artefakts führt, die dann Bestandteil eines Chatbeitrags ist.

Damit erfüllt keines der betrachteten Systeme die für eine verbesserte Interaktion in dualen Interaktionsräumen notwendigen Anforderungen. Im folgenden Kapitel wird für die erste Variante – das Nebeneinander von Diskurs- und Artefaktraum – eine Lösung entwickelt, welche die beiden Interaktionsbereiche auf verschiedene Art und Weisen miteinander integriert und damit die Kollaborierenden bei der deiktischen Bezugnahme, der Koordination ihrer simultanen Aktivitäten und der Reflexion unterstützt.

Tabelle 3.1: Alle vorgestellten Systeme bzw. Ansätze und die jeweilige Erfüllung der Anforderungen im Überblick

Anforderung	GrewpEdit	MapTool	Anchored Conversation	FactChat	GraffiDis	Mit editierbaren Artefakten[a]	MathChat	Electronic Abacus	MSN Messenger
Minimalanforderung:									
A 1: Textueller Diskurs	✓	✓	✓	✓	✓	✓	✓	✓	✓
A 2: Kollaborativ bearbeitbare Artefakte	✓	✓	×	×	(✓)[b]	✓	×	(✓)[c]	×
Deixis:									
I 1: Referenzierung	×	×	(✓)	✓	✓	✓	×	×	×
I 2: auf Objekte & Ausschnitte	×	×	×	✓	✓	✓	×	×	×
I 3: Referenz kodiert Zustand[d]	–	–	×	–	–	×	–	–	–
Koordination:									
I 4: Aktionsdokumentation	×	×	×	–	✓	×	–	–	–
I 5: Identische Reihenfolge[e]	?	?	?	?	?	?	?	?	?
I 6: Integrierte *Awareness*	×	×	×	–	×	×	–	–	–
Reflexion:									
I 7: Kombinierte Historie	(✓)	(✓)	×	–	(✓)	×	✓	✓	✓
I 8: Kontext für Chatbeitrag	×	×	×	–	(✓)	×	✓	✓	✓
I 9: Diskurs zum Artefaktzustand	×	×	×	–	(✓)	×	✓	✓	✓

✓ Anforderung wird erfüllt
(✓) Anforderung wird mit Einschränkung erfüllt
× Anforderung wird nicht erfüllt
– Anforderung nicht anwendbar
? Keine Information verfügbar

[a] Dies ist die hypothetische Erweiterung der Einbettung von Diskurs in Artefaktraum um kollaborativ manipulierbare Artefakte.

[b] Die graphischen Elemente sind – einmal erzeugt – nicht weiter modifizierbar. Dennoch ist es prinzipiell möglich, einfache Zeichnungen gemeinsam zu erstellen.

[c] Die kollaborative Bearbeitung wird in Grenzen über *Copy & Paste* mit anschließender Modifikation des Objektes unterstützt.

[d] Dies ist eine „bedingte“ Anforderung: Wenn das System kollaborativ manipulierbare Artefakte und die Referenzierung unterstützen, wird dann auch der referenzierte Artefaktzustand kodiert und ggf. wieder angezeigt?

[e] Es liegen zu keinem System Informationen darüber vor, ob die Reihenfolge zugesichert wird.

4 Integration dualer Interaktionsräume

In Kapitel 2 wurde aufgezeigt, mit welchen Problemen die Lernenden in dualen Interaktionsräumen konfrontiert sind. Während in der Interaktion die Kollaborierenden ihre Aktivitäten über die Interaktionsbereiche verteilen, sie aufeinander beziehen und miteinander koordinieren, zeigte die Analyse verwandter Forschungsarbeiten und Systeme in Kapitel 3, dass es bisherigen Systemen an integrierenden und überbrückenden Funktionalitäten mangelt.

In diesem Kapitel werden Integrationsmaßnahmen für duale Interaktionsräume entsprechend der in Kapitel 2 abgeleiteten Anforderungen konzipiert. Diese zielen auf die Unterstützung der deiktischen Bezugnahme, der Aktivitätskoordination und der Reflexion und umfassen:

1. die Möglichkeit zur expliziten Referenzierung für die Erleichterung deiktischer Bezugnahmen,
2. die Sicherstellung der Reihenfolge, in der die Teilnehmer die Aktivitäten präsentiert bekommen,
3. die integrierte Darstellung der *Awareness*-Informationen sowie
4. das Dokumentieren der Artefaktmanipulationen in der Chathistorie, um die Koordination der Aktivitäten zu erleichtern, und
5. das Bereitstellen einer kombinierten Interaktionshistorie sowie
6. die Kontextrekonstruktion von Aktionen für die Unterstützung der Reflexion.

4.1 Vorbemerkungen

4.1.1 Grundlegende Designentscheidungen

Gegenstand dieser Arbeit ist die Entwicklung von Integrationsmaßnahmen für duale Interaktionsräume, also für Kollaborationsumgebungen, die aus einem Chat- und einem Konstruktionswerkzeug bestehen. Im Kapitel 2.5 wurden neun Anforderungen an die Integration aufgestellt. In diesem Kapitel wird nun das Konzept eines Integrationssystems vorgestellt, welches einerseits diesen Anforderungen genügt und mit dem andererseits nahezu beliebige Chat- und Konstruktionswerkzeuge integriert werden können. Vorausgesetzt wird hierbei, dass die in den Kapiteln 2.3 und 2.4 formulierten Minimalanforderungen von den Werkzeugen erfüllt werden.[1]

Das Integrationssystem ist dabei eine „Brücke“ zwischen Chat- und Konstruktionswerkzeug. Dies zeigt die folgende Abbildung 4.1:

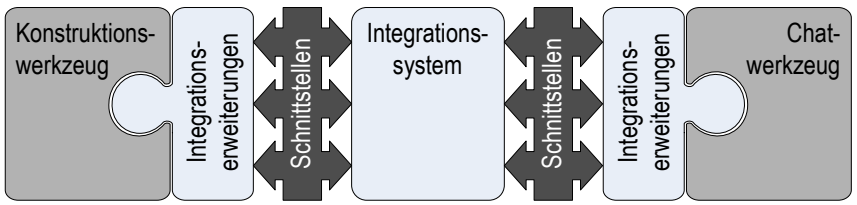


Abbildung 4.1: Schematische Darstellung des Integrationssystems

Das eigentliche Integrationssystem stellt Basisfunktionalitäten für die verschiedenen Integrationsmaßnahmen zur Verfügung. Seitens der Werkzeuge sind je nach Integrationsmaßnahme verschiedene Erweiterungen notwendig. Damit das Integrationssystem mit den Werkzeugen interagieren kann, werden Schnittstellen definiert. Über diese können die werkzeugseitigen Integrationsergänzungen mit dem Integrationssystem und letzteres mit den Werkzeugen in einer standardisierten Form kommunizieren. Um größtmögliche Flexibilität hinsichtlich der technischen Anwendungsmöglichkeiten zu erreichen, wird bei der Modellierung auf folgende Aspekte besonderer Wert gelegt:

Erweiterbarkeit des Integrationssystems: Die für die Integration notwendigen Basisfunktionalitäten sollen leicht an werkzeugspezifische Aspekte angepasst werden können.

Minimale Änderungen an Werkzeugen: Es existiert bereits eine Reihe von Chat- und Konstruktionswerkzeugen. Um diese effizient mit dem Integrationssystem verknüpfen zu können, sollen die für die Integrationsmaßnahmen notwendigen Änderungen minimal gehalten werden.

[1] Weiter unten im Kapitel wird unter Bezugnahme auf die Minimalanforderungen darauf eingegangen, wie diese von den Werkzeugen realisiert werden können bzw. wie mit „Verstößen“ gegen die Voraussetzungen umgegangen werden kann.

Bei Kollaborationsumgebungen, mit denen Teilnehmer von verschiedenen Orten aus synchron miteinander kommunizieren und kollaborieren können, handelt es sich per se um verteilte Systeme (Koch & Schlichter, 2001): Die auf den Rechnern der einzelnen Teilnehmer in jeweils eigenen Laufzeitumgebungen ablaufenden *Groupware*-Anwendungen müssen miteinander kommunizieren, ihre Daten miteinander abgleichen und die Anzeigen aller Benutzer möglichst zeitgleich aktualisieren (Schümmer & Schuckmann, 2001). Das Design und die Implementierung solcher Kollaborationsumgebungen sind im Vergleich zu Einzelbenutzeranwendungen mit einer höheren Komplexität verbunden. Die Komplexität resultiert unter anderem aus folgenden Punkten:

Verteiltes System: Die Systemfunktionalitäten erstrecken sich über mehrere miteinander kommunizierende Rechner. Hieraus folgen als zu behandelnde Probleme die Konsistenzhaltung der jeweiligen lokalen Datenmodelle, die Synchronisation nebenläufiger Prozesse und natürlich die Kommunikation zwischen den beteiligten Computern.

Mehrere Eingabeströme: Im Gegensatz zu Einzelbenutzeranwendungen, die meist nur auf die Eingaben des einzelnen Nutzers reagieren, müssen für Kollaborationsumgebungen die Eingabeströme mehrerer miteinander interagierender Nutzer berücksichtigt werden.

Mittlerweile existiert eine Reihe von *Groupware-Frameworks* zur Entwicklung von Kooperationsanwendungen, die gleichzeitig auch die entsprechende Infrastruktur für die Laufzeit bereitstellen: GroupKit (Roseman & Greenberg, 1996), COAST (Schuckmann et al., 1996), DyCE (Tietze, 2001), DreamTeam (Roth & Unger, 1998) und darauf basierend DreamObjects (Lukosch, 2003), Agilo (Guicking et al., 2005), Sametime (Attardo et al., 2007), um nur einige zu nennen. Ziel der *Frameworks* ist es, die Anwendungsentwickler von einem Großteil der Probleme bei der Programmierung verteilter Systeme zu befreien, indem ihnen fertig implementierte Strukturen bereitgestellt werden, welche zentrale *Groupware*-Funktionalitäten umsetzen (Phillips, 1999). Die in diesem Konzept vorgestellten Integrationsmaßnahmen sind jedoch überwiegend auf Applikationsebene angesiedelt und damit unabhängig von spezifischen *Groupware-Frameworks*.

Der softwaretechnische Entwurf zur Umsetzung der Integrationsmaßnahmen folgt der objektorientierten Modellierung, also dem momentan vorherrschenden Programmierparadigma. Hierbei werden etablierte Entwurfsmuster (Gamma et al., 1995) genutzt. Das Softwaredesign erfolgt entsprechend dem Entwurfsmuster der funktionalen Dekomposition (Avgeriou & Tandler, 2006). Dieses sieht vor, einerseits die Zuständigkeiten unterschiedlicher Komponenten möglichst klar voneinander abzugrenzen und andererseits die Funktionalitäten und die diese Funktionalitäten bereitstellenden Komponenten entsprechend ihrer sachlogischen Zusammenhänge zu gruppieren. Die Komponenten nutzen dann Dienste, die andere Komponenten mittels Schnittstellen bereitstellen. Zur Gruppierung der Funktionalitäten wird Gebrauch gemacht vom Schichtenentwurfsmuster sowie

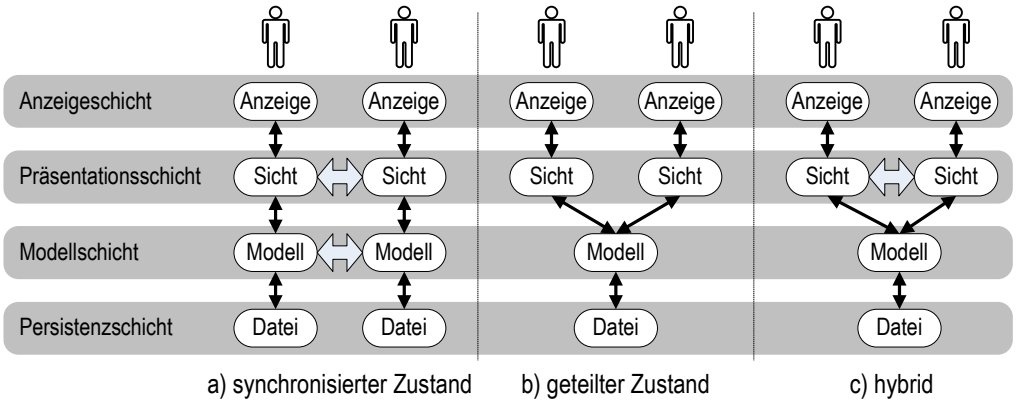


Abbildung 4.2: Pattersons Referenzmodell (nach Patterson, 1995)

dem *Model-View-Controller*-Konzept (Buschmann et al., 1996), auf die nun genauer einzugehen ist.

Nach dem Schichtenentwurfsmuster werden die Systemkomponenten in aufeinander aufbauenden Abstraktionsschichten gruppiert. Jede Schicht stellt über Schnittstellen Dienste zur Verfügung, die sowohl von Komponenten innerhalb der Schicht wie auch der darüber liegenden Schichten genutzt werden können. In welche Schichten das System aufgespalten wird, hängt von dessen Komplexität ab.

Für *Groupware*-Anwendungen wurde auf Basis des Schichtenmodells eine Reihe von Klassifikationsschemata entwickelt, die auf die Beschreibung unterschiedlicher Verteilungsarchitekturen zielen (Patterson, 1995; Dewan, 1999; Phillips, 1999; Roth & Unger, 2000). Abbildung 4.2 zeigt das Referenzmodell von Patterson (1995). In diesem werden vier Schichten unterschieden:

1. In der *Persistenzschicht* werden die Daten der *Groupware*-Anwendung überdauernd (z. B. in einer Datenbank) gespeichert.

2. Die *Modellschicht* enthält die von der Anwendung bearbeiteten Daten.

3. In der *Präsentationsschicht* erfolgt die Berechnung und Beschreibung der Modelldatendarstellung.

4. In der *Anzeigeschicht* sind die für die Anzeige benötigten Betriebssystemteile wie Graphiktreiber gruppiert; sie ist jeweils lokal vorhanden.

Basierend auf dieser Schichtenaufteilung können drei verschiedene *Groupware*-Architekturen unterschieden werden. Bei Applikationen, die der ersten Architektur folgen (siehe Abb. 4.2.a), arbeitet jeder Nutzer mit einer vollständigen lokalen Kopie der Anwendung. Um einen gemeinsamen Anwendungszustand für alle Nutzer herzustellen, müssen auf jeder Schicht Änderungsinformationen zwischen den einzelnen Anwendungen ausgetauscht werden. Hierdurch werden die Zustände der Einzelanwendungen synchronisiert. Im Gegensatz zu diesem Ansatz

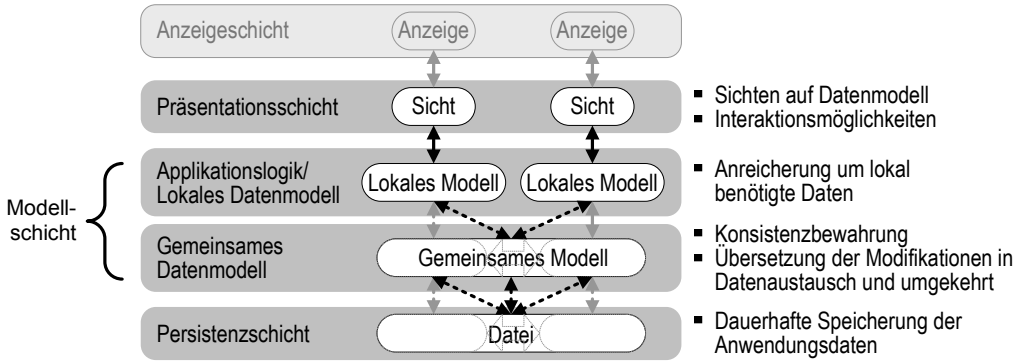


Abbildung 4.3: Schichtenmodell kollaborativer Komponenten

wird bei Architekturen, die auf einem geteilten Zustand basieren, der gemeinsame Anwendungszustand hergestellt, indem untere Schichten von allen Nutzern gemeinsam verwendet werden. Diese Schichten sind im System nur einmal vorhanden, in Abbildung 4.2.b sind es z. B. die Persistenz- und Modellschichten. Der Vollständigkeit halber sei noch die dritte – hybride – Architekturvariante erwähnt, welche eine Mischform der vorangegangenen Varianten ist. Gemeinsame Modellzustände werden erreicht, indem auf ein geteiltes Modell zugegriffen wird. Der gemeinsame Darstellungszustand wird jedoch hergestellt, indem sich die jeweils lokalen Präsentationsschichten synchronisieren.

Von Pattersons Referenzmodell ausgehend werden in dieser Arbeit folgende Annahmen über die Architekturen der Werkzeuge gemacht (siehe Abb. 4.3):

- Die Anzeigeschicht bündelt die nahe am Betriebssystem liegenden Funktionalitäten. Sie wird in modernen Programmiersprachen nicht direkt angesprochen, sondern über GUI-Bibliotheken genutzt. Sie spielt somit für die Modellierung keine Rolle.

- Die Werkzeuge verfügen über nicht-synchronisierte Sichten, um ein *relaxed* WYSIWIS zuzulassen. Dies ist notwendig, um die individuelle Reflexion des bisherigen Interaktionsverlaufs zu ermöglichen, da hierbei unterschiedliche Sichten auf die gemeinsamen Artefakte benötigt werden.

- Die Modellschichten der Werkzeuge lassen sich in zwei Teile aufspalten: Zum einen in das gemeinsame Datenmodell, welches entweder entsprechend der Architektur des geteilten Zustands zentralisiert oder aber entsprechend des synchronisierten Zustands repliziert ist. Und zum anderen in ein lokales Datenmodell, welches basierend auf dem gemeinsamen Datenmodell entsprechend der Applikationslogik gebildet wird und die lokal benötigten Daten vorhält.

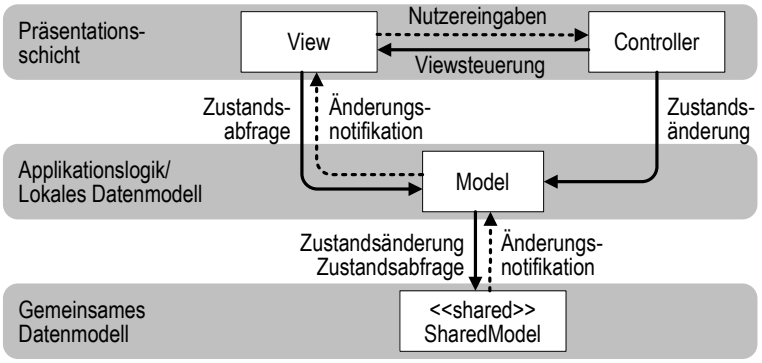


Abbildung 4.4: Kooperatives MVC (nach Rubart, 2005, S. 118)

- Analog zum gemeinsamen Datenmodell wird für jedes Werkzeug eine Persistenzschicht angenommen, deren Verteilungsarchitektur nicht weiter eingeschränkt wird.

Aus der Aufspaltung der Modellschichten in ein gemeinsames und ein lokales Datenmodell folgend wird in der Konzeption davon ausgegangen, dass die Teilnehmer eine Clientapplikation nutzen, um auf die Kollaborationsumgebung, welche die beiden Werkzeuge in einem dualen Interaktionsraum verknüpft, zuzugreifen.

Die Integrationsmaßnahmen zielen auf die Unterstützung der Teilnehmer bei der Interaktion in dualen Interaktionsräumen und sind vorrangig auf der Präsentationsschicht und der Modellschicht angesiedelt. Bei der Strukturierung der Funktionalitäten auf diesen beiden Schichten hat sich das *Model-View-Controller* (MVC)-Konzept etabliert. Dieses dient dazu, die Benutzungsschnittstelle (das *Graphical User Interface*, GUI) mit ihren Anzeigen und Ein- und Ausgabemöglichkeiten vom zugrunde liegenden Datenmodell sowie der Applikationslogik zu trennen. Nutzereingaben werden durch den *Controller* interpretiert und das Datenmodell entsprechend modifiziert. Änderungen im Datenmodell wiederum werden an den *View* propagiert, welcher die Darstellung aktualisiert und dafür das Datenmodell direkt ausliest. Die *Controller*- und *View*-Komponenten sind meist eng miteinander verzahnt. Eine Applikation besteht üblicherweise aus einer Reihe von Modellen mit zugehörigen *View-Controller*-Paaren. Diese *View-Controller*-Paare sind meist in einer *Subview/Superview*-Baumstruktur organisiert, welche die visuelle Teil-Ganzes-Beziehung repräsentiert (Phillips, 1999, S. 14). Somit können die Nutzereingaben (z. B. Mausbewegungen) und das Zeichnen des GUI lokal und damit effizient behandelt werden.

Das MVC-Konzept lässt sich, wie in Abbildung 4.4 dargestellt, für *Groupware*-Applikationen verfeinern (Rubart, 2005; Roth, 2000). Für *Groupware*-Applikationen ist es notwendig, ein allen Teilnehmern gleichermaßen zur Verfügung stehendes Datenmodell zu verwalten. Dieses gemeinsame Datenmodell kapselt die Konsistenzhaltung zwischen den jeweiligen Instanzen pro Nutzer. Es benachrichtigt das

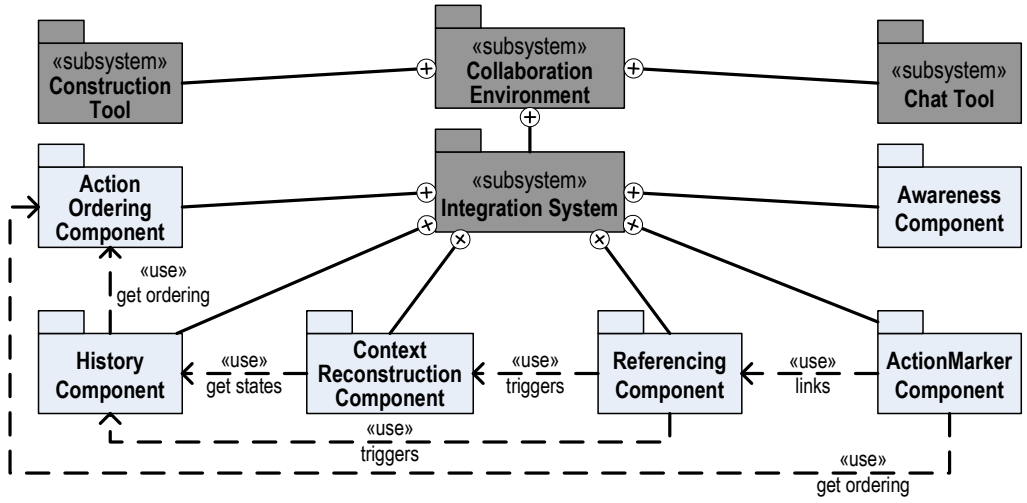


Abbildung 4.5: Abstrakte Sicht des Integrationssystems mit seinen zentralen Elementen und deren Beziehungen untereinander

lokale Datenmodell über Änderungen seiner Daten bzw. seines Zustandes und stellt Methoden zur Zustandsänderung (die entsprechend die Konsistenz des globalen gemeinsamen Datenmodells zusichern) und zur Abfrage bereit.

Die Elemente *Model*, *View*, *Controller* und *SharedModel*[2] lassen sich den drei Schichten des Schichtenmodells zuordnen: Das *SharedModel* entspricht dem gemeinsamen Datenmodell, das (lokale) *Model* befindet sich auf Ebene der Applikationslogik und der *View* zusammen mit dem *Controller* in der Präsentationsschicht.

Das kooperative MVC-Konzept dient im Folgenden als Rahmenkonzept zum Entwurf der Integrationskomponenten. Bevor hiermit begonnen wird, soll im nächsten Abschnitt jedoch zunächst auf die funktionale Dekomposition der Integrationsmaßnahmen eingegangen werden.

4.1.2 Funktionale Dekomposition

Die funktionale Dekomposition dient der Strukturierung eines Softwaresystems, um die Zuständigkeiten der verschiedenen Systemkomponenten möglichst klar voneinander abzugrenzen. Eine erste Dekomposition ergibt sich aus den in der Problemanalyse abgeleiteten Anforderungen. Diese lassen sich in sechs Funktionsbereiche einteilen, welche durch eigenständige Integrationskomponenten realisiert werden (siehe die sechs weiß hinterlegten Komponenten in Abbildung 4.5):

Konsistente Aktionsabfolgen: Für alle Teilnehmer soll die Reihenfolge, in der Chatbeiträge eintreffen (Anforderung **C1**) sowie die Änderungen im gemeinsamen Arbeitsbereich des Konstruktionswerkzeugs sichtbar werden

[2] Entsprechend der in der Informatik vorherrschenden Konvention werden im Folgenden für Systembestandteile englische Bezeichnungen verwendet.

(Anforderung **K 1**), dieselbe sein. Dies gilt auch für die Reihenfolge, in der sich die Chatbeiträge und die Artefaktänderungen abwechseln (Anforderung **I 5**). Dies fällt unter die Zuständigkeiten der *ActionOrdering*-Komponente.

Kombinierte Interaktionshistorien: Die beiden Interaktionsbereiche müssen jeweils persistente Interaktionshistorien bereitstellen (Anforderung **C 3** sowie **K 3**). Diese müssen darüber hinaus integriert, also in ihrer verschränkten Abfolge verfügbar gemacht werden (Anforderung **I 7**). Zuständig hierfür ist die *History*-Komponente.

Kontextrekonstruktion: Zu jedem Chatbeitrag müssen der zugehörige Artefaktkontext (Anforderung **I 8**) und zu jedem Artefaktzustand der entsprechende Chatdiskurs (Anforderung **I 9**) ermittelt werden können. Bereitgestellt werden diese Funktionalitäten durch die *ContextReconstruction*-Komponente.

Aktivitätsdokumentation: Im Chattranskript sollen die durch die Kollaborierenden getätigten Aktionen dokumentiert werden, so dass die zeitliche Abfolge der Chatbeiträge und Aktionen sichtbar wird (Anforderung **I 4**). Dies wird erreicht durch die *ActionMarker*-Komponente.

Deiktische Bezugnahme: Es muss den Teilnehmenden möglich sein, ihre Chatbeiträge mit anderen Chatbeiträgen und Objekten der gemeinsamen Arbeitsumgebung zu verknüpfen. Diese Verknüpfungen sollen mit den Nachrichten übertragen und den Empfängern angezeigt werden (Anforderungen **C 2**, **I 1**, **I 2** sowie **I 3**). Dies wird realisiert durch die *Referencing*-Komponente.

Integrierte *Awareness*-Anzeige: Zur Erleichterung der Koordination simultaner Aktivitäten müssen die *Awareness*-Informationen, die im Chatwerkzeug (Anforderungen **C 4**) und im Konstruktionswerkzeug (**K 2**) verfügbar sind, integriert präsentiert werden (Anforderung **I 6**). Umgesetzt wird dies durch die *Awareness*-Komponente.

Eine andere Dekomposition ergibt sich aus dem grundlegenden Aufbau der Kollaborationsumgebung (siehe Kap. 2.2): Diese besteht aus einem Chat- und einem Konstruktionswerkzeug und wird um die im Integrationssystem zusammengefassten Integrationskomponenten erweitert. Somit ergibt sich aus Systemsicht eine Aufteilung in die drei grundlegenden Teilsysteme Chat- und Konstruktionswerkzeug sowie Integrationssystem (siehe wiederum Abb. 4.5):

Das Chatwerkzeug stellt die Funktionalitäten für die Chatkommunikation bereit.

Das Konstruktionswerkzeug erlaubt die kollaborative Konstruktion gemeinsamer Artefakte.

Im Integrationssystem – dem Fokus dieser Arbeit – sind all die Komponenten gebündelt, die zur Umsetzung der Integrationsmaßnahmen notwendig sind.

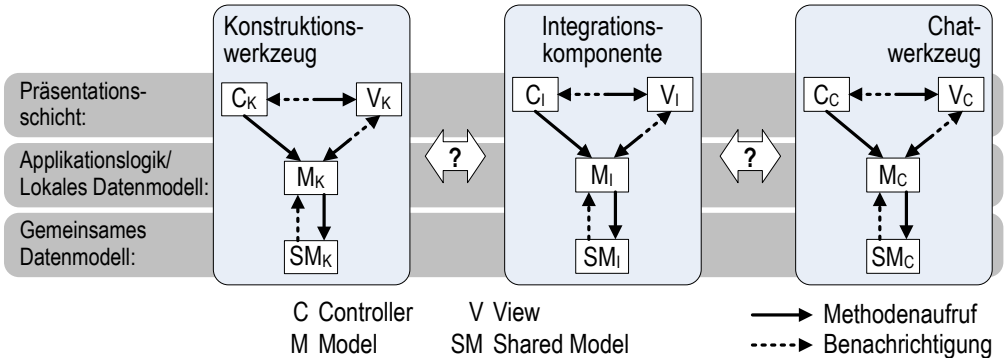


Abbildung 4.6: MVC-Architektur dualer integrierter Interaktionsräume

Mit dieser Aufteilung in drei Teilsysteme ergibt sich zusammen mit dem Konzept des kollaborativen MVC nun ein erstes allgemeines Architekturmodell integrierter Interaktionsräume (siehe Abb. 4.6).

Die beiden Interaktionsbereiche werden durch die Integrationskomponenten verbunden. Für jede Integrationskomponente können das Konstruktionswerkzeug, das Chatwerkzeug und diese Integrationskomponente in Form des kooperativen MVC-Konzeptes gedacht werden. Sie bestehen jeweils aus *View*, *Controller*, *Model* und *SharedModel* (zur Unterscheidung sind diese in der Abbildung mit einem Index versehen).

Während das MVC-Konzept die einzelnen Komponenten vertikal strukturiert, ergibt sich durch die Aufteilung in die drei Grundkomponenten eine horizontale Strukturierung.

Hiermit gilt es nun, für jeden der oben beschriebenen sechs Funktionsbereiche a) die notwendigen Integrationsfunktionalitäten zu entwickeln und b) die geeigneten softwaretechnischen Verknüpfungspunkte zu den beiden Werkzeugen zu identifizieren (dies ist in der Abbildung durch die Pfeile mit den Fragezeichen symbolisiert), so dass die Anforderungen erfüllt und die in Kapitel 4.1.1 genannten Designaspekte „Erweiterbarkeit“ und „Minimale Änderungen an Werkzeugen“ gewährleistet werden.

4.1.3 Zum Aufbau des Kapitels

Das weitere Kapitel ist wie folgt strukturiert. Zuerst wird in Abschnitt 4.2 auf die grundlegende Frage eingegangen, wie sichergestellt werden kann, dass die über alle Interaktionsbereiche hinweg erfolgenden Aktionen allen Teilnehmern in derselben Reihenfolge präsentiert werden (*ActionOrdering*-Komponente). Darauf basierend wird in Abschnitt 4.3 geklärt, wie diese Reihenfolge persistent gespeichert und später nach dem Betreten des Kollaborationsraums wieder verfügbar gemacht werden kann (*History*-Komponente). Darauf aufbauend können dann die Mechanismen zur Rekonstruktion des Kontextes realisiert werden (*ContextReconstruction*-Komponente). Dies wird in Abschnitt 4.4 dargelegt. Auch die *ActionMar-*

ker-Komponente zur bereichsübergreifenden Aktivitätsdokumentation, auf die in Abschnitt 4.5 eingegangen wird, nutzt die durch die *ActionOrdering*-Komponente sichergestellte globale Aktionsordnung. Weitgehend unabhängig von diesen Teilkomponenten kann die explizite Referenzierung in Abschnitt 4.6 konzipiert werden und in Abschnitt 4.7 die Integration der *Awareness*-Informationen erfolgen. Abschließend wird in Abschnitt 4.8 die gesamte Lösung dargestellt und dabei auf die Beziehungen der verschiedenen Komponenten untereinander eingegangen.

4.2 Konsistente Aktionsabfolgen

4.2.1 Einleitung

In örtlich verteilten Lernsituationen erfolgt jegliche Interaktion zwischen den Teilnehmern, die sich im selben (virtuellen) Kollaborationsraum befinden, über die Kollaborationsumgebung (siehe Kap. 2.2). In dualen Interaktionsräumen, die aus einem Chat- und einem Konstruktionswerkzeug bestehen, interagieren die Teilnehmer über den Austausch von Chatbeiträgen im Chatwerkzeug und die Manipulationen an den Artefakten im Konstruktionswerkzeug.

In der Anforderung **C 1** wird gefordert, dass für alle Teilnehmer die Chatbeiträge in derselben Reihenfolge sichtbar werden. Anforderung **I 5** erweitert dies auf die Gesamtabfolge aller Aktionen im dualen Interaktionsraum: Für alle Teilnehmer soll die Reihenfolge, in der die Chatbeiträge bzw. die Manipulationen an den Artefakten im Konstruktionswerkzeug sichtbar werden, dieselbe sein.

Im Folgenden wird das Konzept der einheitlichen Aktionsordnung genauer beleuchtet. Anschließend erfolgt die Modellierung eines allgemeinen Nachrichtenaustauschmodells, welches diese Aktionsordnung zusichert. Abschließend wird dargelegt, welche Möglichkeiten in Abhängigkeit von den verwendeten *Groupware*-Infrastrukturen bestehen, dieses Nachrichtenaustauschmodell in bestehenden Werkzeugen umzusetzen.

4.2.2 Konzeption der Aktionsordnung

Alle Aktionen im dualen Interaktionsraum sollen von allen Teilnehmern in derselben Reihenfolge empfangen werden. Was ist nun aber eine Aktion? Im Sinne einer Arbeitsdefinition soll im Folgenden unter einer Aktion eine atomare, für alle Teilnehmer beobachtbare Veränderung im Kollaborationsraum verstanden werden. Im Chat ist dies aufgrund der medialen Eigenschaften gleichbedeutend mit dem Verschicken eines Beitrags, jeder Chatbeitrag ist gleichzeitig die Beschreibung einer Aktion. Desgleichen gilt für die in dieser Arbeit betrachteten gemeinsamen Arbeitsbereiche in Form diskreter Medien (siehe Kap. 2.4), dass ihr Zustand allein von diskreten (zumeist nutzerinitiierten) Ereignissen abhängt. Diese Ereignisse sind die beobachtbaren Manipulationen an den gemeinsamen Artefakten, sind also die atomaren Aktionen. Die Granularität der Aktionen kann dabei sehr fein

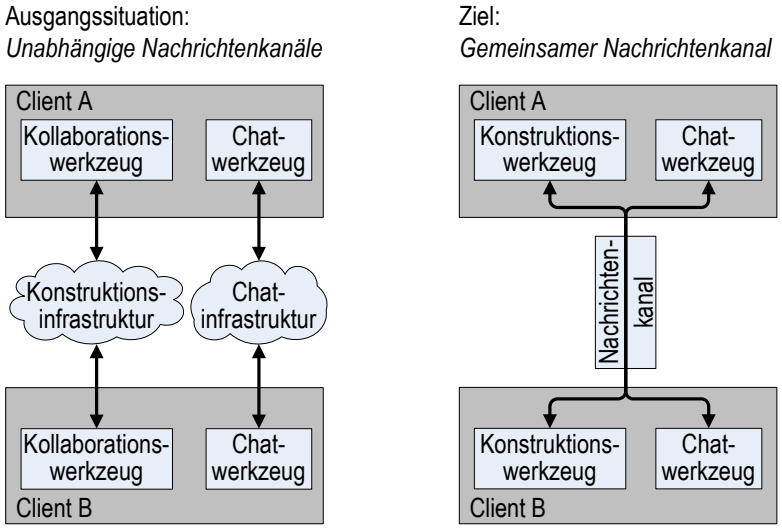


Abbildung 4.7: Schematische Darstellung des Aktionsordungsproblems

sein, wie es in *Shared Whiteboards* der Fall ist, oder aber eher grob, wenn als gemeinsames Material zum Beispiel ein Wiki[3] genutzt wird, bei dem das nichtbeobachtbare Editieren einer Seite durch einen Teilnehmer umfangreich sein kann.

Im allgemeinen Falle werden die Aktionen von jedem der beiden Werkzeuge über eine eigene Infrastruktur verschickt: Die Chatinfrastruktur erledigt die Zustellung der verschickten Beiträge, die Infrastruktur des Konstruktionswerkzeuges überträgt die Manipulationen an den Artefakten an die Teilnehmer im Kollaborationsraum. Dies ist in Abbildung 4.7 als Ausgangssituation dargestellt. Ziel ist es nun, die in den beiden Werkzeugen unabhängig voneinander verschickten Aktionen in eine gemeinsame Ordnung zu bringen. Dargestellt ist dies in Abbildung 4.7 auf der rechten Seite: Die Aktionen erscheinen als über einen gemeinsamen Nachrichtenkanal, der die Ordnung zusichert, verschickt. Die konkreten Anforderungen an diesen Nachrichtenkanal ergeben sich aus der folgenden Formalisierung der Aktionsordnung:[4] Seien

- $B = \{b_1, \ldots, b_n\}$ die Menge der Interaktionsbereiche in der Applikation,[5]

[3] Wikis sind Sammlungen von Webseiten, die nicht nur gelesen, sondern auch online unkompliziert erstellt und bearbeitet werden können. Das erste Wiki wurde 1995 von Ward Cunningham entwickelt. Der Name geht auf das hawaiische Wort „wiki“ für „schnell“ zurück (siehe `http://c2.com/doc/etymology.html`).

[4] Die Formalisierung folgt der von ter Hofte 1998, S. 235ff. Dort findet sich auch eine ausführliche Diskussion anderer – weniger restriktiver – Nachrichtenordnungen.

[5] Im konkreten Falle besteht die Menge aus dem Chat- und dem Konstruktionswerkzeug und somit gilt $n = 2$. In der Formalisierung wird zur leichteren Darstellung von einer beliebigen Menge an Interaktionsbereichen ausgegangen.

- $P = \{p_1, \ldots, p_n\}$ die Menge der sendenden und empfangenden Knoten im System (also die verschiedenen Laufzeitumgebungen der Teilnehmer), wobei jedes p_i aus den Interaktionsbereichen B besteht,

- A die Menge aller und $A_b \subseteq A$ die Menge der aus dem Interaktionsbereich $b \in B$ stammenden Aktionen,[6]

- $s_p(a)$ das Sendeereignis zur Aktion $a \in A$ in einem Knoten $p \in P$, wobei ein Sendeereignis zu einer Aktion natürlicherweise nur in genau einem Knoten des Systems (dem Sender) auftritt, und

- $r_p(a)$ für jeden Knoten $p \in P$ das Empfangsereignis zur Aktion $a \in A$.

Dann soll für die Ereignisse in einem Knoten gelten, dass diese nicht gleichzeitig auftreten können. Die Relation $<_p$ über alle Ereignisse aller Interaktionsbereiche eines Knotens $p \in P$ legt somit eine totale Ordnung dieser Ereignisse fest.

Des Weiteren gilt, dass der Sendezeitpunkt einer Aktion immer vor den zugehörigen Empfangszeitpunkten liegt. Sei $<_a$ die Ordnung der Ereignisse von $a \in A$ und sei $p \in P$ der Sender zu a, dann gilt für alle $p' \in P$: $s_p(a) <_a r_{p'}(a)$.

Hiermit lassen sich nun die Anforderungen an die Aktionsordnungen spezifizieren:

Bewahrung der Sendereihenfolge: Stammen zwei Aktionen vom selben Sender, dann werden diese von allen Knoten in derselben Reihenfolge empfangen. Somit soll für je zwei Aktionen $a_1, a_2 \in A, a_1 \neq a_2$ die Ordnung $<_{FIFO}$ gelten mit:

$$\big(\exists p \in P : s_p(a_1) <_p s_p(a_2)\big) \Rightarrow \big(\forall p \in P : r_p(a_1) <_{FIFO} r_p(a_2)\big)$$

Identische Empfangsreihenfolgen: Alle Empfänger erhalten die Aktionen, inklusive konkurrierender Aktionen von verschiedenen Sendern, in derselben Reihenfolge. Für je zwei Aktionen $a_1, a_2 \in A, a_1 \neq a_2$ muss also die (totale) Ordnung $<_{REC}$ gelten mit:

$$\big(\exists p \in P : r_p(a_1) <_p r_p(a_2)\big) \Rightarrow \big(\forall p \in P : r_p(a_1) <_{REC} r_p(a_2)\big)$$

Während die erste Bedingung festlegt, dass Aktionen, die von einem Knoten verschickt werden, auch in dieser Reihenfolge empfangen werden, beschreibt die zweite Bedingung, dass die Reihenfolge des Aktionsempfangs, selbst wenn die Aktionen nahezu gleichzeitig von verschiedenen Knoten verschickt wurden, überall dieselbe ist. Die beiden Bedingungen sind nicht redundant, denn während die erste den Sendezeitpunkt berücksichtigt, ist die zweite Bedingung allein über den Empfangszeitpunkten definiert.

[6] Da eine Aktion in genau einem Interaktionsbereich ausgelöst wird, gilt $\forall b_1, b_2 \in B, b_1 \neq b_2 : A_{b_1} \cap A_{b_2} = \emptyset$.

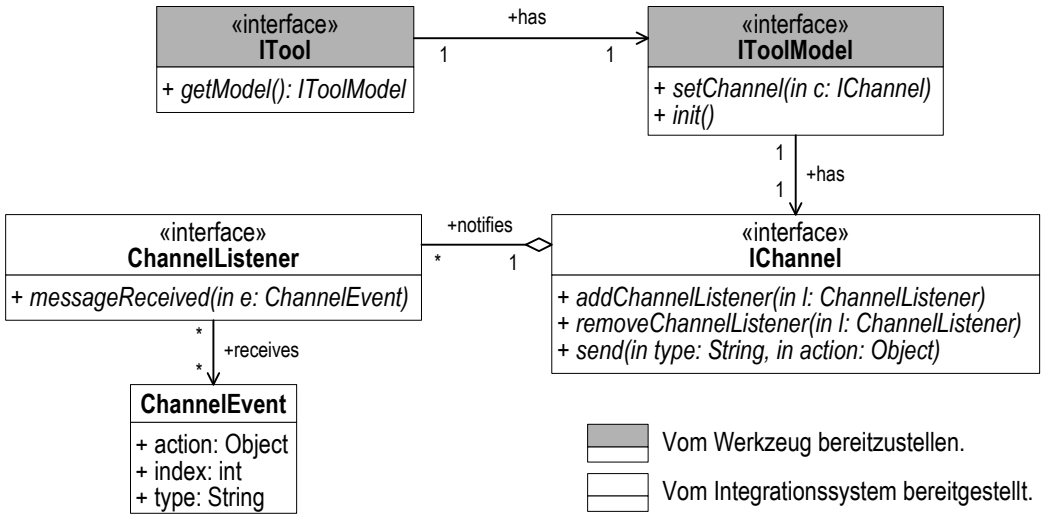


Abbildung 4.8: Klassendiagramm für den Nachrichtenkanal

4.2.3 Modellierung des Nachrichtenaustauschs

Das Nachrichtenaustauschmodell geht davon aus, dass alle Interaktionsbereiche denselben Kanal zum Versenden und Empfangen von Aktionen verwenden. Pro Kollaborationsraum steht genau ein solcher Kanal zur Verfügung. Die totale Ordnung der Aktionen wird durch diesen Kanal realisiert.

Abbildung 4.8 zeigt das Diagramm der relevanten Klassen und Schnittstellen. Jedes Werkzeug muss das *Interface* `ITool`[7] implementieren, über welches auf das zugehörige `IToolModel` zugegriffen werden kann. In diesem Modell kann nun der Kanal `IChannel` gesetzt werden.

Für das Versenden hat ein Kanal eine `send`-Methode, der die zu sendende Aktion und eine Typbezeichnung übergeben wird. Dieser Aktionstyp dient dazu, die Nachrichten aus den verschiedenen Interaktionsbereichen zu unterscheiden.

Entsprechend der Nachrichtenordnung können die Aktionen indiziert werden. Es sei im Folgenden $I \subseteq \mathbb{N}$ die Indexmenge zur Indizierung der Aktionen und $ind_A : A \rightarrow I$ die Abbildung der Aktionen auf diese Indexmenge, so dass gilt: $\forall a_1, a_2 \in A, a_1 \neq a_2 : \big(ind_A(a_1) < ind_A(a_2)\big) \Leftrightarrow \big(r_p(a_1) <_{REC} r_p(a_2)\big)$ mit $p \in P$.[8]

Der Empfangsmechanismus folgt dem *Observer*-Entwurfsmuster: An einem Kanal können `ChannelListener` registriert werden. Diese werden vom Kanal über

[7] Der Konvention folgend wird *Interface*-Namen ein „I“ vorangestellt, es sei denn, es handelt sich um ein *Listener-Interface*. Desweiteren werden konkrete Klassen-, *Interface*- und Methodennamen `nicht-proportional` gesetzt.

[8] Mit dieser Definition wird zugelassen, dass die Indizes aufeinander folgender Aktionen nicht unbedingt aufeinander folgende Zahlen n und $n+1$ sein müssen, es kann durchaus auch „Lücken“ in der Indizierung geben. Der Grund hierfür ist rein pragmatisch: Der Nachrichtenkanal wird eventuell auch für Nachrichten jenseits der Aktionen genutzt, zum Beispiel für das Verschicken von *Awareness*-Informationen. Technisch ist die Indizierung einfacher, wenn sie ungeachtet der Semantik der Nachrichten erfolgen kann.

Aufruf der `messageReceived`-Methode benachrichtigt, sobald eine neue Aktion eintrifft. Hierzu wird ihnen ein Ereignisobjekt vom Typ `ChannelEvent` übergeben, welches das Empfangsereignis beschreibt: Neben dem eigentlichen Aktionsobjekt und dem Aktionstyp liegen bei Empfang auch der sich aus der Nachrichtenordnung ergebende Index vor.

4.2.4 Integrationsvarianten

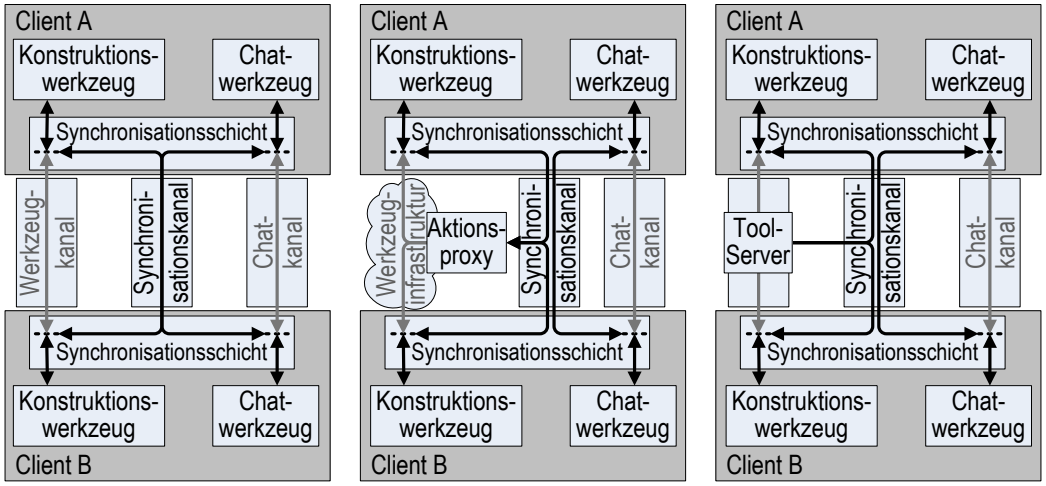
Wie kann diese totale Ordnung der Aktionen nun bei der Integration eines Konstruktionswerkzeugs mit einem Chat sichergestellt werden? Bei der Beantwortung dieser Frage müssen drei Aspekte bedacht werden:

1. Wie kann die totale Aktionsordnung prinzipiell hergestellt werden?
2. Wie können die (jeweils total geordneten) Aktionsströme aus den beiden Interaktionsbereichen in eine gemeinsame Ordnung gebracht werden?
3. Wie kann die Ordnung selbst für Konstruktionswerkzeuge, die diese bisher nicht zusichern, hergestellt werden?

Beginnen wir mit dem ersten Aspekt. Davon ausgehend, dass Aktionen als elementare Nachrichten im System verschickt werden, existiert für die Realisierung der totalen Aktionsordnung entsprechend der oben genannten zwei Bedingungen eine Reihe von Ansätzen. Der einfachste ist sicherlich die Verwendung einer zentralen Instanz im System (eines Servers). Jeder Knoten im System verschickt die Nachrichten an diesen Server und bestimmt dabei die knoteninterne Sendereihenfolge. Der Server bringt die eingehenden Nachrichtenströme von den verschiedenen Knoten in eine globale Reihenfolge und verschickt sie in dieser auch wieder an alle Knoten, womit jeder Knoten alle Nachrichten in derselben Reihenfolge erhält. Beispiele für eine solche Infrastruktur sind Habanero (Chabert et al., 1998) und Agilo[9] (Guicking et al., 2005). Andere (dezentrale) Ansätze, die ohne einen dedizierten Server auskommen, verwenden Sperrmechanismen. Diese stellen die Ordnung darüber her, dass es im System eine globale Sperre gibt, die immer nur ein Knoten besitzt. Dabei darf dann immer nur derjenige Knoten senden, in dessen Besitz sich die Sperre gerade befindet. Beispiele für Infrastrukturen, die diesen Mechanismus umsetzen, sind DreamTeam (Roth, 2000) und darauf aufbauend DreamObjects (Lukosch, 2003). Somit stehen Lösungen für *Groupware*-Infrastrukturen zur Verfügung, mit denen die totale Ordnung sichergestellt werden kann.

Wenden wir uns dem zweiten Aspekt zu: Wie können die beiden Aktionsströme in eine globale Ordnung gebracht werden? Am einfachsten lässt sich dies realisieren, indem beide Interaktionsbereiche dieselbe *Groupware*-Infrastruktur verwenden und diese die totale Ordnung zusichert. Damit sind die beiden Interaktionsbereiche nur Teilkomponenten innerhalb der Knoten und die Aktionen aus den

[9] Agilo wird vom *ConcertChat*-System, welches in Kapitel 5 vorgestellt wird, als Infrastruktur genutzt.



a) Synchronisierender Nachrichtenkanal

b) Synchronisierender Nachrichtenkanal mit Aktions-Proxy

c) Synchronisierender Nachrichtenkanal mit Tool-Server

Abbildung 4.9: Zusicherung der Nachrichtenordnung

beiden Bereichen entsprechend nur Nachrichten ein und desselben Sendestroms. Dies entspricht der in Abbildung 4.7 gezeigten Zielsituation.[10]

Ein anderer Fall liegt vor, wenn die Interaktionsbereiche unterschiedliche *Groupware*-Infrastrukturen nutzen, die jeweils die totale Ordnung zusichern. Hier muss die bereichsübergreifende Nachrichtenordnung durch einen eigenen Mechanismus realisiert werden. Dies ist möglich, indem für die Synchronisation eine separate, die Nachrichtenordnung zusichernde Infrastruktur zum Einsatz kommt. Clientseitig muss dann eine Synchronisationsschicht eingezogen werden (siehe Abb. 4.9.a). Wird innerhalb eines Interaktionsbereichs eine Nachricht msg verschickt, so wird in dieser Schicht eine zugehörige Synchronisationsnachricht $sync$ erzeugt und über den Synchronisationskanal gesendet. Auf Empfängerseite wird die bereichsinterne Nachricht msg bzw. die Synchronisationsnachricht $sync$ so lange durch die Synchronisationsschicht verzögert, bis die entsprechende Synchronisationsnachricht $sync$ bzw. die Nachricht msg eintrifft.

Dieser Ansatz ist jedoch nicht anwendbar, wenn die Aktionen bereits lokal optimistisch ausgeführt werden (wie z. B. in DyCE, siehe Tietze, 2001), denn hierbei können konfligierende Aktionen ein *Rollback* der bereits lokal ausgeführten Aktion verursachen. Um dem zu begegnen, kann die in Abbildung 4.9.b gezeigte Variante eingesetzt werden. Die Aktionen werden hierbei über den Synchronisationskanal an einen ausgezeichneten Knoten im System, den Aktionsproxy, geschickt. Erst dieser führt sie dann tatsächlich aus und sendet im Erfolgsfall eine entsprechende Notifikation an alle Knoten.

[10] Dies ist auch die vom *ConcertChat*-System (siehe Kap. 5) realisierte Variante.

Mit diesen Varianten lässt sich somit die totale Nachrichtenordnung auch für Werkzeuge herstellen, welche diese bisher nicht zusicherten: In Variante a werden die Aktionen nach dem Empfang sortiert, in Variante b durch die geordnete Ausführung im dedizierten Aktionsproxy.[11]

In manchen Fällen ist es sinnvoller, die Synchronisationsnachricht nicht clientseitig, sondern durch eine zentrale, werkzeugspezifische Instanz im System (den Tool-Server) zu erzeugen (siehe Abb. 4.9.c). Dies bietet sich an, wenn ein Werkzeug auch außerhalb der Kollaborationsumgebung genutzt wird, aber auf dieselben Ressourcen (die gemeinsamen Artefakte oder den Chatraum) zugreift. So könnten z. B. bei der Integration von Wiki-Seiten als gemeinsame Artefakte die Seiten auch über das entsprechende Web-Interface editiert werden.

Der Einsatz eines synchronisierenden Nachrichtenkanals hat den Nachteil, dass sie bei allen drei gezeigten Varianten zu einer Verdoppelung des Nachrichtenaufkommens führt, denn für jede werkzeuginterne Nachricht muss eine synchronisierende Nachricht erzeugt und verschickt werden. Jedoch kann damit prinzipiell die totale Aktionsordnung auf Ebene der Synchronisationsschicht realisiert werden. Aus Sicht des gemeinsamen Datenmodells sind die notwendigen Anpassungen transparent. Im Folgenden kann deshalb weitgehend von den konkreten Infrastrukturen der Interaktionsbereiche abstrahiert und das allgemeine Modell eines Nachrichtenkanals, welcher von den Interaktionsbereichen gemeinsam genutzt wird, angewendet werden.

Die totale Ordnung und die damit mögliche Indizierung der Aktionen sind die Basis für die Bereitstellung einer vollständigen Interaktionshistorie, die Kontextrekonstruktion und die bereichsübergreifende Aktivitätsdokumentation. Auf diese Funktionsbereiche wird nun in den folgenden Abschnitten eingegangen.

4.3 Integrierte Historien

4.3.1 Einleitung

Wie in Kapitel 2 ausgeführt, muss den Teilnehmern der Interaktionsverlauf verfügbar gemacht werden, ihnen soll nach dem Betreten eines Kollaborationsraums der bisherige Interaktionsverlauf zum Nachlesen, Reflektieren und als Gedächtnisstütze bereitstehen. Dies setzt voraus, dass der Chatdiskurs (Minimalanforderung **C 3**) sowie die Artefakthistorie (Minimalanforderung **K 3**) systemweit persistent verfügbar sind, und dass natürlich auch die ineinander verwobene Abfolge von Chatbeiträgen und Artefaktänderungen erhalten bleibt (Anforderung **I 7**).

In diesem Kapitel wird dargestellt, wie die vollständige Interaktionshistorie verfügbar gemacht werden kann. Auf die Funktionalitäten, mit denen die Nutzer den

[11] *Groupware*-Systeme, die von der totalen Nachrichtenordnung abweichen, zielen meist auf möglichst geringe Systemantwortzeiten. Diese werden durch das Herstellen der totalen Ordnung jedoch in der Regel verlängert. In dieser Arbeit ist die Nachrichtenordnung jedoch das „höhere Gut“.

Interaktionsverlauf explorieren können, wird später in den Abschnitten zur Kontextrekonstruktion (Kap. 4.4) und Aktivitätsdokumentation (Kap. 4.5) eingegangen.

Die Darstellung der integrierten Historien beginnt mit einer Konzeptualisierung integrierter Historien. Anschließend erfolgt die Dekomposition der notwendigen Funktionalitäten, die dann für die Modellierung des *History*-Systems genutzt wird. In Kapitel 4.3.5 wird genauer auf den Mechanismus eingegangen, mit dem die verschiedenen persistenten Historien geladen und integriert werden. Die Betrachtung des *History*-Systems endet mit einer Diskussion verschiedener Integrationsvarianten.

4.3.2 Konzeption integrierter Historien

Was ist nun eine Interaktionshistorie? Wie in Kapitel 2.4.1 diskutiert und bereits bei der Betrachtung der konsistenten Aktionsabfolgen berücksichtigt, werden in dieser Arbeit als Konstruktionswerkzeuge nur diskrete Medien betrachtet. In diesen wird der Zustand der gemeinsamen Artefakte allein durch die elementaren Aktionen verändert. Auch der Chat ist ein diskretes Medium, jedes Verschicken eines Beitrags ist eine elementare Aktion, die den Zustand des Chattranskripts ändert.

Im Chat besteht die vollständige Historie somit aus allen bisher verschickten Beiträgen. Die Interaktionshistorie des Konstruktionswerkzeugs muss an dieser Stelle jedoch abstrakt behandelt werden, da diese je nach Werkzeug unterschiedliche Ausprägungen annehmen kann: In einem *Shared Whiteboard* ist die Historie die sich aus den elementaren Manipulationen ergebende Abfolge der Zeichnungszustände, in einem Wiki ist es die Liste der unterschiedlichen Seitenversionen, in einem Brainstorming-Werkzeug zum Sammeln von Ideen die Abfolge der von den Teilnehmern eingegebenen Ideen. Für die Speicherung und Rekonstruktion aller Zustände der vollständigen Historien diskreter Medien mit feingranularen elementaren Operationen wurden bereits Lösungen entwickelt (siehe z. B. Mauve, 2000; Landsman, 2006[12]). Für diese Arbeit mit dem Fokus auf den Fragen der Integration der Interaktionsbereiche kann deshalb davon ausgegangen werden, dass die unterschiedlichen Zustände der Historie im Konstruktionswerkzeug verfügbar sind.

Für die kombinierte Interaktionshistorie gilt somit, dass durch jede Aktion in einem der Interaktionsbereiche der Gesamtzustand der Kollaborationsumgebung

[12] Die beiden zitierten Arbeiten fokussieren auf die Verfügbarkeit einer vollständigen Interaktionshistorie, die im Nachhinein insbesondere zu Analysezwecken wieder abgespielt werden kann. Die dort vorgestellten Mechanismen lassen sich jedoch auf den hier beschriebenen Fall verallgemeinern. Sie beruhen darauf, die Abfolge der Operationen zu speichern. Indem zu jeder Operation auch eine inverse Operation definiert wird, kann von jedem Zustand aus jeder andere Zustand über Ausführen der Operationen bzw. der inversen Operationen berechnet werden. Zur Steigerung der Performanz werden gegebenenfalls Zwischenzustände gespeichert, so dass der aktuelle Zustand nicht immer über Ausführung aller Operationen, sondern nur derjenigen, die nach dem letzten bekannten Zwischenzustand erfolgten, rekonstruiert werden kann.

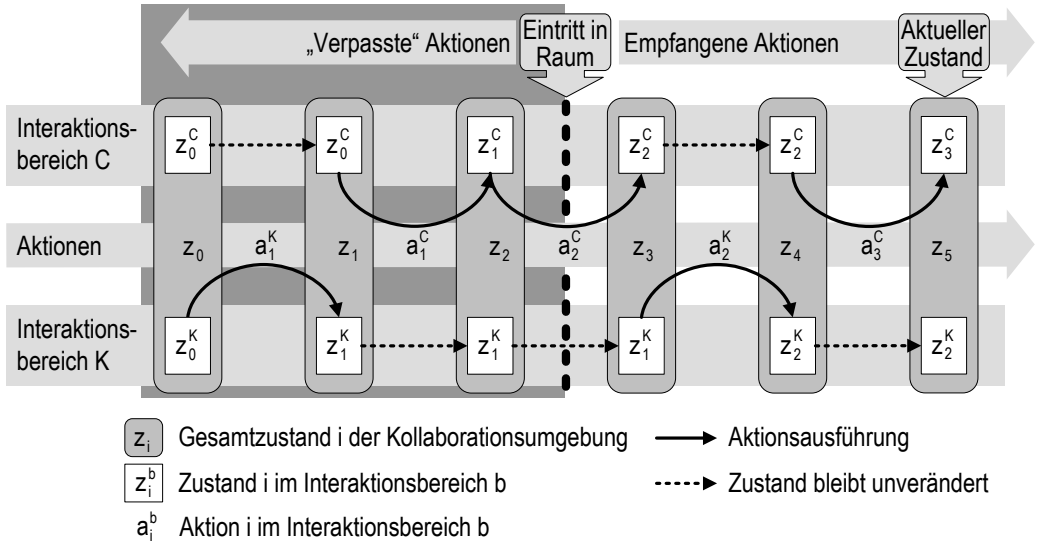


Abbildung 4.10: Zustände, Aktionen und deren Beziehung

verändert wird (siehe Abb. 4.10): Jede Aktion a_i^b in einem der Interaktionsbereiche $b \in B$ überführt den dortigen Zustand z_i^b in den Zustand z_{i+1}^b und somit die Kollaborationsumgebung in einen neuen Gesamtzustand. Aufgrund der eindeutigen Ordnung der Aktionen über die Interaktionsbereiche hinweg ergibt sich damit auch eine eindeutige Ordnung dieser Gesamtzustände.

Um den Interaktionsverlauf verfügbar zu machen, müssen in den Laufzeitumgebungen der Teilnehmer die einzelnen Zustände bzw. die sie verursachenden Aktionen bekannt sein. Der Verlauf umfasst einerseits all die Aktionen, die nach dem Betreten des Kollaborationsraums über den gemeinsamen Nachrichtenkanal in der Laufzeitumgebung eintreffen (siehe die rechte Hälfte in Abb. 4.10). Darüber hinaus können aber bereits vor dem Betreten Aktionen im Raum ausgeführt worden sein: von vorher eingetroffenen Teilnehmern, in vorangegangenen Sitzungen oder vom Tutor, der den Raum für die Kollaboration vorbereitete. Diese Aktionen müssen demzufolge im System vorrätig gehalten und bei Bedarf in die Laufzeitumgebung des Teilnehmers geladen werden.

Abbildung 4.11 verdeutlicht das Problem der Bereitstellung einer integrierten Gesamthistorie: Beide Interaktionsbereiche verfügen jeweils über eine eigene Historie der im Bereich getätigten Aktionen bzw. der daraus resultierenden Bereichszustände. Im Client werden pro Interaktionsbereich jeweils die vor dem Betreten des Raumes erfolgten Aktionen mit den über den entsprechenden Kanal empfangenen kombiniert. Um aus diesen beiden separaten Historien die Gesamthistorie der Kollaborationsumgebung zu bilden, bedarf es zusätzlich jedoch noch der Information über die bereichsübergreifende Aktionsabfolge, die sich wiederum aus der persistent gespeicherten Aktionshistorie wie aus den über den globalen Nachrichtenkanal empfangenen Nachrichten speist.

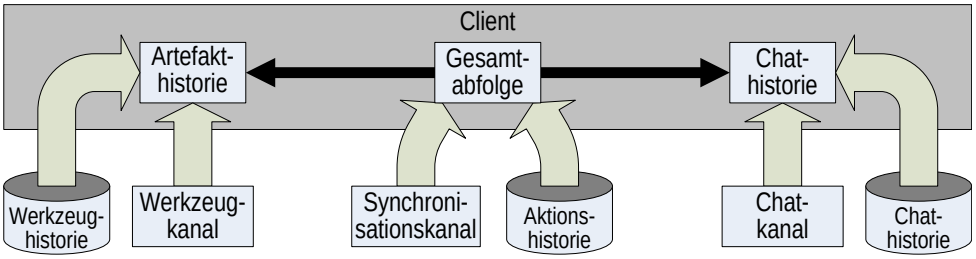


Abbildung 4.11: Die drei Aspekte der Interaktionshistorie: Werkzeug- und Chathistorie sowie deren werkzeugübergreifende Abfolge

Für die weitere Modellierung wird für das Konstruktions- und das Chatwerkzeug folgende interne Struktur bezüglich der Historienbehandlung angenommen (siehe Abb. 4.12):Beide Werkzeuge besitzen jeweils eine Infrastruktur zum dauerhaften Speichern ihrer Historien (das `PersistencyBackend`). Im Chatwerkzeug wird hierzu jeder vom `MessageSender` über den gemeinsamen Nachrichtenkanal (siehe Kap. 4.2.3) verschickte Beitrag auch an das `PersistencyBackend` gereicht. Das Laden der vor dem Betreten des Raumes verschickten Beiträge in die Laufzeitumgebung erfolgt über die `HistoryLoader`-Komponente, die den Zugriff auf das `PersistencyBackend` kapselt. In einem Datenmodell (dem `ChatModel`) werden die aus dem `PersistencyBackend` geladenen Beiträge mit denen vom `MessageReceiver` nach dem Betreten des Raumes empfangenen gemischt.

Analog hält im Konstruktionswerkzeug ein `VersionsModel` die durch elementare Manipulationen an den gemeinsamen Artefakten entstandenen Artefaktversionen vor. Diese ergeben sich entsprechend sowohl aus den nach dem Betreten vom `OperationReceiver` empfangenen neuen sowie den aus dem `PersistencyBackend` geladenen alten Versionen.

Ziel des *History*-Systems ist es nun, clientseitig eine Sicht auf die Abfolge aller Aktionen und der daraus resultierenden Gesamtzustände zu ermöglichen, und dies unabhängig davon, ob die Aktionen vor oder nach dem Betreten des Raumes erfolgten.

4.3.3 Funktionale Dekomposition

Ausgangspunkt für die funktionale Dekomposition sind die werkzeugintern verfügbaren Historien. Diese müssen in ihrer Abfolge vom *History*-System ansprechbar sein. Hierfür wird für jede elementare Manipulation (die zu einer neuen Artefaktversion führt) eine Aktionsnachricht erzeugt, die eine Referenz auf die werkzeuginterne Manipulation bzw. Artefaktversion enthält. Die Aktionsnachrichten werden vom *History*-System persistent gespeichert. Da die Aktionsnachrichten total geordnet sind, bleibt die globale Ordnung aller Aktionen erhalten. Indem die Aktionsnachrichten auf die werkzeuginternen Manipulationen bzw. Artefakt-

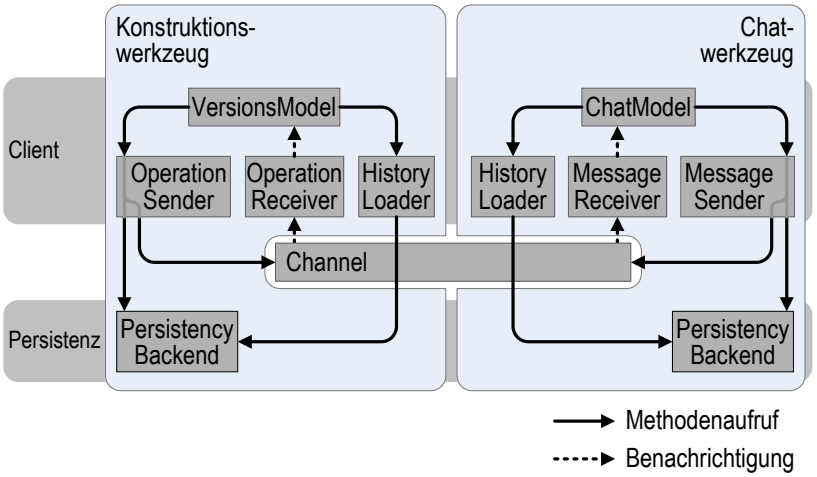


Abbildung 4.12: Schematische Darstellung der werkzeuginternen Historiensysteme *vor* der Integration

versionen verweisen, können diese anhand der Aktionsnachrichten rekonstruiert werden.

Werden von den Werkzeugen nach dem Betreten des Kollaborationsraums die vergangenen Versionen der Historie abgerufen, müssen synchron im *History*-System die entsprechenden Aktionsnachrichten geladen werden, um die werkzeuginternen Versionen clientseitig in die globale Interaktionshistorie zu integrieren.

Somit spalten sich die Funktionalitäten zur Bereitstellung der vollständigen Interaktionshistorie in die folgenden Aspekte auf:

Erzeugen von Aktionen: Die Werkzeuge müssen für jede Version/Manipulation eine Aktionsnachricht versenden. Diese Aktionsnachrichten müssen systemweit gespeichert werden.

Zuordnen von Aktionen zu Versionen: Es muss die Zuordnung von Aktionsnachricht zur werkzeuginternen Version und umgekehrt ermöglicht werden.

Laden der Historien: Lädt ein Werkzeug intern Versionen aus der Historie, so müssen auch die korrespondierenden Aktionsnachrichten geladen werden.

Mit dieser Aufspaltung in drei Funktionsbereiche lässt sich das System zur Verwaltung der integrierten Historien modellieren, worauf im folgenden Abschnitt eingegangen wird.

4.3.4 Modellierung des *History*-Systems

Wie Abbildung 4.13 zeigt, besteht das *History*-System aus einer `History`-Komponente und bedarf einer Reihe von Erweiterungen an den Interaktionsberei-

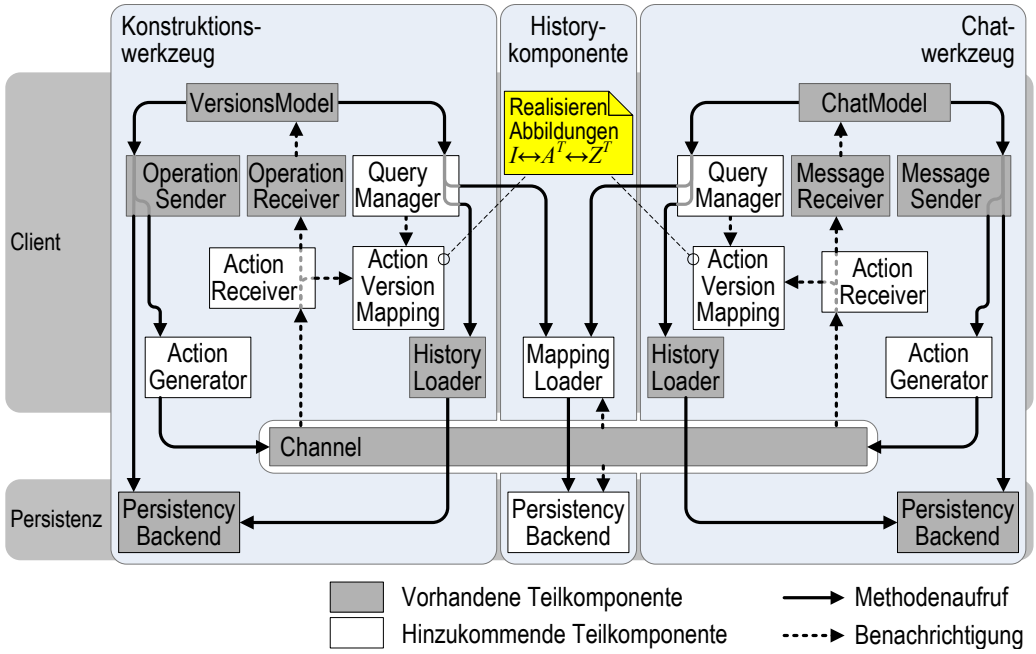


Abbildung 4.13: Das *History*-System

chen. Wie oben ausgeführt, werden für die werkzeugspezifischen Historien-Mechanismen analoge interne Strukturen angenommen. Insofern sind für die Integration auch analoge Erweiterungen notwendig. Auf diese soll nun eingegangen werden.

Zu jeder werkzeuginternen Nachricht msg zur Erzeugung einer neuen Version wird für die Bereitstellung der vollständigen Historien von einem geeigneten `ActionGenerator` eine Aktionsnachricht act erzeugt, welche auf die Version bzw. Manipulation verweist.[13] Während die werkzeuginternen Nachrichten msg den spezifischen *Inhalt* einer Aktion beschreibt (zum Beispiel den modifizierten Inhalt einer Wiki-Seite), enthält die Aktionsnachricht act allein einen *Verweis* auf die Aktion (z. B. die Nummer der neuen Seitenversion).

Die Aktionsnachricht wird über den gemeinsamen Nachrichtenkanal verschickt und somit mit einem eindeutigen Index im Nachrichtenstrom versehen. Vom `ActionReceiver` werden auf Empfängerseite die korrespondierenden Nachrichten msg und act zusammengeführt und die werkzeuginterne Nachricht msg an den entsprechenden `Receiver` weitergereicht.

Der `ActionReceiver` ist ein `ChannelListener` am gemeinsamen Kanal (siehe Abb. 4.8). Er erhält damit auch den zur Nachricht gehörenden globalen Index. Die Aktionsnachricht zusammen mit dem Index werden an das `ActionVersion-`

[13] Falls die Werkzeuge eine eigene Infrastruktur zur Nachrichtenübertragung nutzen (siehe Kap. 4.2.3), kann die Aktionsnachricht in die zur Herstellung der Aktionsordnung genutzte *sync*-Nachricht „eingepackt“ werden.

`Mapping` übermittelt, womit dieses die Aktionsnachrichten A^b den werkzeuginternen Versionen (bzw. Zuständen) Z^b und den globalen Indizes I zuordnen kann. Hiermit können die später im Kapitel 4.4 pro Interaktionsbereich benötigten Abbildungen realisiert werden:

- die (partielle[14]) Abbildung der globalen Indizes auf die Menge der Aktionsnachrichten $act^b : I \rightarrow A^b$,
- die Abbildung der Aktionsnachrichten auf die resultierenden Zustände $result : A^b \rightarrow Z^b$ und umgekehrt die Abbildung der Zustände auf die sie verursachenden Aktionsnachrichten $caused\_by : Z^b \rightarrow A^b$.

Die Abbildung $act^b : I \rightarrow A^b$ kann bereits mit den Informationen im `ChannelEvent`, welches dem `ChannelListener` übergeben wird, aufgebaut werden. Die Zuordnungen der Aktionen zu den Zuständen bzw. Versionen $result : A^b \rightarrow Z^b$ und $caused\_by : Z^b \rightarrow A^b$ muss – da die Version ein werkzeuginternes Konstrukt ist – werkzeugspezifisch verwaltet werden.

Alle (über den gemeinsamen Nachrichtenkanal verschickten) Aktionsnachrichten werden auch vom `PersistencyBackend` des *History*-Systems empfangen und dort zusammen mit dem Typ – über den die „Herkunft“ aus einem Werkzeug kodiert wird – und dem globalen Index gespeichert. Das `PersistencyBackend` muss im System unabhängig von den Laufzeitumgebungen der Teilnehmer verfügbar sein, z. B. auf einem dedizierten Knoten.

Auf die Daten im `PersistencyBackend` muss zugegriffen werden, wenn in einem der Werkzeuge die interne Versionshistorie vom werkzeugspezifischen `HistoryLoader` geladen wird. Die verschiedenen `HistoryLoader` der Werkzeuge wie auch der `MappingLoader` der `History`-Komponente stellen Dienste zum Laden der im System persistent gespeicherten Historien zur Verfügung. Damit werden die Details der Anbindung an das jeweilige `PersistencyBackend` gekapselt. Wird nun in einem der Werkzeuge die werkzeuginterne Historie geladen, so müssen auch die entsprechenden Aktionsnachrichten aus dem `PersistencyBackend` der `History`-Komponente abgerufen werden. Hierzu werden die werkzeuginternen *History*-Abfragen an den `QueryManager` umgelenkt. Dieser leitet einerseits die Abfrage über den `HistoryLoader` an das werkzeugeigene `PersistencyBackend` weiter und erzeugt andererseits eine korrespondierende Abfrage, die über den `MappingLoader` der *History*-Komponente an das zugehörige `PersistencyBackend` geschickt wird. Nachdem von beiden Abfragen die Ergebnisse vorliegen, werden diese an das `ActionVersionMapping` gereicht, wo mit ihnen die oben erwähnten Abbildungen erweitert werden können. Hierdurch bleiben die durch das `ActionVersionMapping` verwalteten Abbildungen synchron zu den im Werkzeug verfügbaren Versionen.

[14] Die Abbildung ist partiell, weil nicht jedes Element der globalen Indexmenge I auf eine Aktion in A^b abgebildet werden kann.

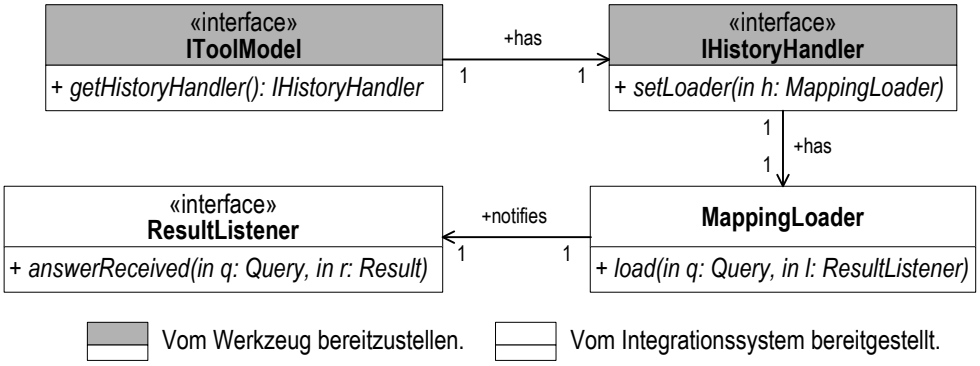


Abbildung 4.14: Klassendiagramm der für die Schnittstelle zwischen den Werkzeugen und dem *History*-System relevanten Klassen

Sowohl der `Actionenerator` wie auch das `ActionVersionMapping` bedürfen werkzeugspezifischer Anpassungen, da hier die Abbildung zwischen werkzeuginternen Versionsinformationen und Aktionsnachrichten erfolgen muss. Im Gegensatz dazu bietet der `MappingLoader` einen generischen Dienst zum Laden der im `PersistencyBackend` gespeicherten Aktionsnachrichten an. Auf die Art und Weise, wie die Historien geladen werden können, wird im nächsten Abschnitt eingegangen.

Zuvor soll jedoch zum Abschluss der Modellierung des *History*-Systems beleuchtet werden, welche Schnittstellen das *History*-System zu den Werkzeugen besitzt. Wie Abbildung 4.13 zeigt, ist die einzige horizontale Beziehung zwischen Werkzeug und `History`-Komponente der Zugriff auf den `MappingLoader`. Insofern reduziert sich die Schnittstelle darauf, den Werkzeugen den `MappingLoader` bereitzustellen. Dies erfolgt entsprechend dem *Dependency-Injection*-Entwurfsmuster. Hierbei wird den Werkzeugen bei der Initialisierung im Integrationssystem der `MappingLoader` übergeben. Abbildung 4.14 zeigt die beteiligten Klassen: Über das `IToolModel` (siehe Abb. 4.8, S. 74) des Werkzeugs wird auf den werkzeugeigenen `IHistoryHandler` zugegriffen, dem der `MappingLoader` übergeben wird. Aus Sicht der Werkzeuge stellt der `MappingLoader` eine `load`-Methode bereit, über die die Anfragen gestellt werden können. Auf den Lademechanismus wird detailliert im folgenden Abschnitt eingegangen.

4.3.5 Laden von Historien

Zur Verdeutlichung der notwendigen Änderungen soll zu Beginn nochmals der angenommene werkzeuginterne Ladevorgang beschrieben werden. Dieser ist in Abbildung 4.15.a in Form eines Aktivitätsdiagramms zu sehen: Vom werkzeuginternen `VersionsModel` geht an den vorhandenen `HistoryLoader` die Aufforderung zum Laden älterer Versionen der Werkzeughistorie. Dieser leitet die Anfrage

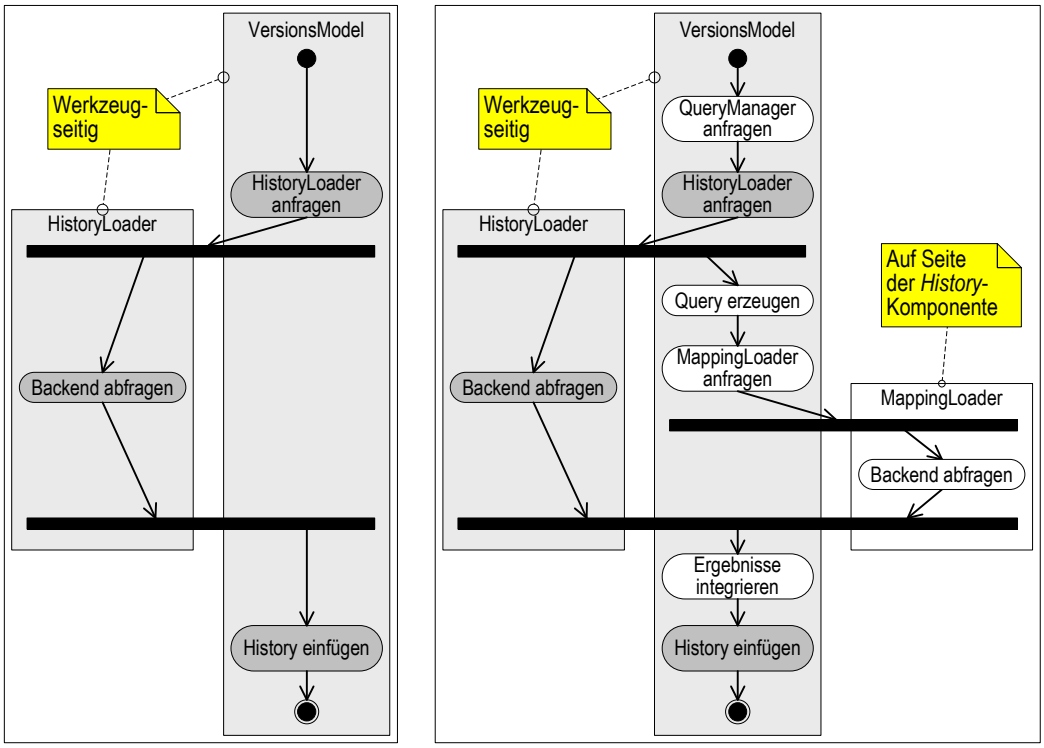


Abbildung 4.15: Sequenzdiagramm zum Abfragen der Historie

an das `PersistencyBackend` weiter. Nach dem Erhalt wird das Ergebnis in das `VersionsModel` eingefügt.

Für die Integration der Historien ist es nun notwendig, nicht nur die Inhalte der älteren Versionen zu laden, sondern auch die korrespondierenden Aktionszuordnungen. Diese befinden sich im PersistencyBackend der `History`-Komponente. Abbildung 4.15.b zeigt das Aktivitätsdiagramm des gemeinsamen Ladens von Inhalt und Zuordnungen älterer Versionen: Die werkzeuginterne Ladeaufforderung vom `VersionsModel` wird an den hinzukommenden `QueryManager` geleitet. Dieser reicht die Aufforderung nicht-blockierend an den `HistoryLoader` weiter. Die Bearbeitung der Anfrage kann unter Umständen zeitaufwendig sein, sie wird deshalb asynchron gestartet. Unterdessen wird eine korrespondierende Anfrage an den `MappingLoader` gestellt, mit der die zugehörigen Aktionszuordnungen geladen werden. Der `MappingLoader` schickt die Anfrage an das `PersistencyBackend` der `History`-Komponente. Der `QueryManager` muss auf das Eintreffen der beiden Ergebnisse warten, bevor diese integriert an das `ActionVersionMapping` geliefert und die ursprüngliche Anfrage vom `VersionsModel` beantwortet werden kann. Um den Programmfluss im `VersionsModel` durch den unter

Umständen langwierigen Abfrageprozess nicht zu blockieren, bietet sich auch hier ein *Callback*-Mechanismus an.

Wie oben schon angedeutet, können die Werkzeuge unterschiedliche Strategien anwenden, wie die Historien während der Laufzeit geladen werden. Denkbar sind unter anderem:

Vollständiges Laden der Historie: Die allgemeinste Variante ist, die vollständige bisherige Historie nach Betreten des Kollaborationsraums zu laden.

Sukzessives Zurückschreiten in der Historie: Eine andere Variante besteht darin, schrittweise Teile der Historie nachzuladen. Dies ist zum Beispiel für den Chat sinnvoll, um nicht immer den – unter Umständen sehr umfangreichen – vollständigen Diskurs, sondern je nach Bedarf des Nutzers nur sukzessiv ältere Teile (des vorangegangenen Diskurses) abzurufen.

Der vom `MappingLoader` der `History`-Komponente angebotene Dienst zum Laden von Aktionsnachrichten muss deshalb generisch gehalten werden. Im werkzeugspezifischen `QueryManager` wird deshalb ein zur jeweiligen Strategie passendes Abfrageobjekt erzeugt, welches über den `MappingLoader` an das `PersistencyBackend` der `History`-Komponente geschickt wird. Dort muss für jeden Abfragetyp ein geeigneter `QueryHandler` registriert sein, welcher dann die eigentliche Bearbeitung der Abfrage durchführt.[15]

4.3.6 Integrationsvarianten

Die im Kapitel 4.3.4 vorgestellte Modellierung des *History*-Systems ging von dem allgemeinen Fall aus, dass das Konstruktions- sowie das Chatwerkzeug jeweils ein eigenes `PersistencyBackend` und eine eigene *Messaging*-Infrastruktur nutzen. Entsprechend aufwendig sind damit auch Datenaustausch und Synchronisation zwischen den Werkzeugen und dem *History*-System. Je nach Wahl von Infrastruktur und Verteilungsarchitektur vereinfachen sich diese jedoch. In Abbildung 4.16 sind zwei Varianten dargestellt, die für beide Werkzeuge analog und unabhängig voneinander umsetzbar sind.

Abbildung 4.16.a zeigt den Fall, dass das Werkzeug und das *History*-System zwar unterschiedliche `PersistencyBackends`, aber eine gemeinsame *Messaging*-Infrastruktur nutzen, wobei die Erzeugung der werkzeugseitigen Aktionsnachrichten zentralisiert erfolgt. Diese Variante bietet sich z. B. für die Integration von Wiki-Seiten als gemeinsame Artefakte an. Jedes Speichern einer neuen Version wird vom Wiki-Server über den `ActionGenerator` an die Nachrichtenkanäle aller Kollaborationsräume, welche diese Seite als Material nutzen, gesendet. Der Abruf einer Version der Historie entspricht dann dem Abruf einer speziellen URL.

Noch stärker vereinfacht sich die Realisierung der gemeinsamen Historien, wenn nicht nur eine gemeinsame *Messaging*-Infrastruktur, sondern auch eine ge-

[15] Dies entspricht dem *Factory*-Entwurfsmuster (Gamma et al., 1995).

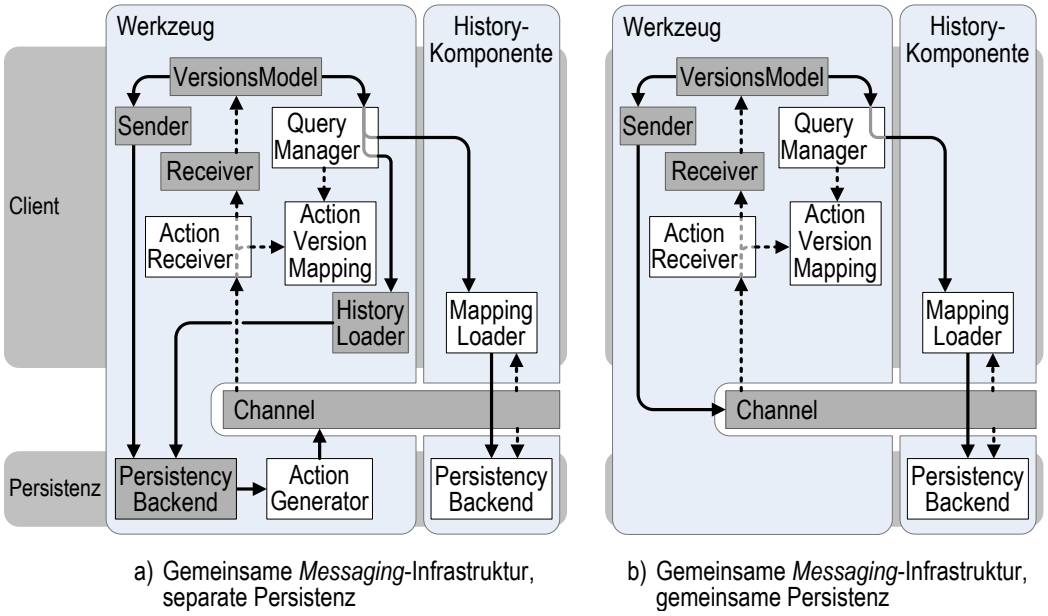


Abbildung 4.16: Vereinfachte Varianten des *History*-Systems

meinsame Persistenz genutzt wird (siehe Abb. 4.16.b). Hierbei entfällt die Doppelung der werkzeuginternen Nachrichten durch Aktionsnachrichten und entsprechend auch die Doppelung der Historienabfragen. Die im `PersistencyBackend` gespeicherten Nachrichten werden hierbei direkt zur Rekonstruktion der einzelnen Versionen genutzt.[16]

4.4 Kontextrekonstruktion

4.4.1 Einleitung

Basierend auf der globalen Indizierung aller Aktionen (siehe Kap. 4.2) und ihrer vom *History*-System bereitgestellten Zuordnung zu den resultierenden Zuständen (siehe Kap. 4.3) kann die Rekonstruktion des Kontextes zu einer Aktion realisiert werden. Die Kontextrekonstruktion soll es den Nutzern erleichtern, den bisherigen Interaktionsverlauf zu explorieren und nachzuvollziehen. Nach Anforderung **I8** soll es möglich sein, zu beliebigen Chatbeiträgen den Artefaktkontext zu rekonstruieren. Analog soll es nach Anforderung **I9** möglich sein, den Chatdiskurs zu beliebigen Artefaktzuständen zu identifizieren. Ziel des Systems zur Kontextrekonstruktion ist es, hierfür Funktionalitäten bereitzustellen.

[16] Dies ist auch die vom *ConcertChat*-System, welches in Kapitel 5 vorgestellt wird, realisierte Variante.

Nach der im folgenden Abschnitt dargestellten Konzeptualisierung des Aktionskontextes erfolgt die funktionale Dekomposition, die zur abschließenden Modellierung des Systems zur Kontextrekonstruktion genutzt wird.

4.4.2 Konzeption des Aktionskontextes

Wie in Kapitel 2.5 ausgeführt, sind die Aktionen der Teilnehmer in der Kollaborationsumgebung aufeinander bezogen: Mit Chatbeiträgen wird auf andere Chatbeiträge oder Manipulationen der gemeinsamen Artefakte reagiert; Änderungen an den Artefakten können durch den vorangegangenen Chat motiviert und im nachfolgenden Diskurs kommentiert werden. Indem es den Nutzern ermöglicht wird, den jeweiligen Kontext zu einer Aktion zu ermitteln, wird der Interaktionsverlauf leichter nachvollziehbar. Hierfür bedarf es zweier Varianten der Kontextrekonstruktion:

Kontext eines Chatbeitrags: Ein Chatbeitrag wurde im Kontext eines bestimmten Artefaktzustandes bzw. einer bestimmten Manipulation am Artefakt erstellt. Um den Beitrag im Nachhinein zu verstehen, bedarf es deshalb der Kenntnis dieses Artefaktzustandes bzw. der getätigten Manipulation. In graphisch orientierten Konstruktionswerkzeugen wie einem *Shared Whiteboard* kann hierfür der entsprechende Artefaktzustand rekonstruiert werden. In Werkzeugen, die eher der Sammlung von Elementen dienen (wie dem bereits oben als Beispiel erwähnten Brainstorming-Werkzeug), kann dies aber auch durch ein Hervorheben der im Kontext des Chatbeitrags generierten Elemente geschehen (eine vollständige Rekonstruktion des Werkzeugzustandes wäre unter Umständen sogar kontraproduktiv).

Kontext einer Artefaktmanipulation: Umgekehrt gilt für eine Artefaktmanipulation, dass diese im Kontext des Chatdiskurses erfolgte. Um den Sinn dieser Manipulation später nachvollziehen zu können, kann der sprachliche Austausch im Chat, der diese Manipulation begleitete, ihr vorausging und ihr folgte, herangezogen werden. Hierfür ist es jedoch notwendig, dass diese Diskurssequenz leicht identifiziert werden kann. Dies kann zum Beispiel durch ein graphisches Hervorheben erfolgen.

In Kapitel 4.3.2 wurde bereits bei der Konzeption der integrierten Historien dargelegt, dass jede Aktion – egal ob Chatbeitrag oder Manipulation der Artefakte – in einem der Interaktionsbereiche diesen in einen neuen Zustand und damit die Kollaborationsumgebung in einen neuen Gesamtzustand überführt (siehe Abb. 4.10). Obwohl bei der Kontextrekonstruktion nicht unbedingt der vollständige Originalzustand wiederhergestellt werden muss, sondern eventuell korrespondierende Aktionen oder Beiträge hervorgehoben werden, kann – wie in der folgenden Formalisierung deutlich wird – die Information über den Zustand zur Kontextdarstellung herangezogen werden:

Kontextrekonstruktion zum Chatbeitrag: Im Chat sind die elementaren Aktionen identisch mit den Chatbeiträgen. Der Artefaktkontext zu einem Chatbeitrag ist derjenige Zustand, in dem sich die gemeinsamen Artefakte beim Empfang des Beitrags befanden.[17] Sei also $a_i^C \in A^C$ ein Chatbeitrag. Der zugehörige Artefaktkontext ist derjenige Zustand $z_j^K \in Z^K$, für den mit $a_j^K = caused\_by(z_j^K)$ gilt:

$$ind(a_j^K) < ind(a_i^C) \text{ und } \nexists a_k^K \in A^K \text{ mit } ind(a_j^K) < ind(a_k^K) < ind(a_i^C).$$

Das heißt, es muss in A_K die Aktion mit dem maximalen globalen Index, der kleiner als der globale Index von a_i^C ist, ermittelt werden. Der entsprechende Zustand ergibt sich dann über $z_j^K = result(a_j^K)$.

Kontextrekonstruktion zum Artefaktzustand: Der Diskurskontext zu einem Artefaktzustand ist bestimmt durch den Chatbeitrag, der zuletzt empfangen wurde, bevor die zu diesem Artefaktzustand führende Aktion ausgeführt wurde. Sei also $z_j^K \in Z^K$ ein Artefaktzustand, verursacht durch die Aktion $a_j^K = caused\_by(z_j^K)$. Der zugehörige Chatbeitrag ist diejenige Aktion $a_i^C \in A^C$, für die gilt:

$$ind(a_i^C) < ind(a_j^K) \text{ und } \nexists a_l^C \in A^C \text{ mit } ind(a_i^C) < ind(a_l^C) < ind(a_j^K).$$

Das heißt auch hier, dass diejenige Aktion in A^C mit maximalem globalem Index, der kleiner als der von a_j^K ist, ermittelt werden muss.

Daraus folgt, dass der Kontext zu einem Chatbeitrag bzw. einer Artefaktversion über deren globale Indizes bestimmt werden kann.

4.4.3 Funktionale Dekomposition

Aus dieser Konzeptionalisierung der Aktionskontexte lassen sich nun direkt die benötigten Funktionalitäten ableiten:

Indexdetektion: Für die Rekonstruktion des Kontextes zu einem Chatbeitrag bzw. einer Artefaktversion muss der entsprechende globale Index ermittelt werden.

Notifikation: Der komplementäre Interaktionsbereich muss darüber benachrichtigt werden, dass der Kontext entsprechend dieses globalen Indexes dargestellt werden soll.

Kontextdarstellung: Werkzeugspezifisch muss nun der Kontext zu diesem globalen Index dargestellt werden.

[17] Es wird hier nicht die Perspektive des Beitragsautors (dazu müsste der Kontext während der Beitragserstellung rekonstruiert werden), sondern die aller anderen Kollaborationspartner eingenommen.

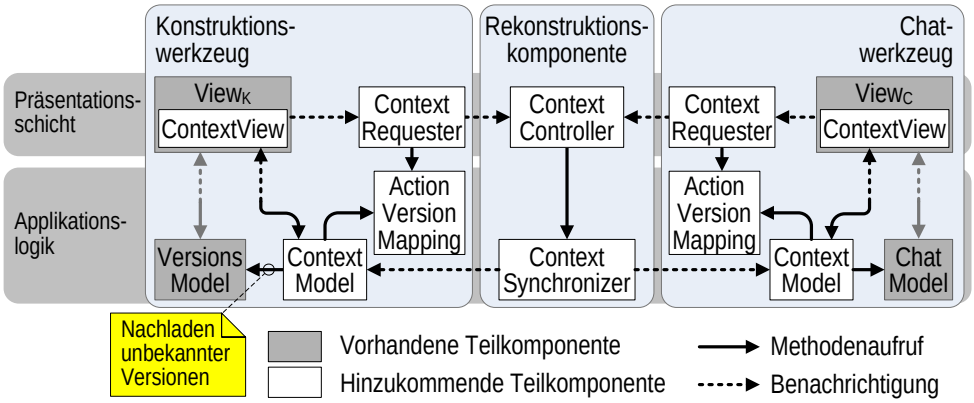


Abbildung 4.17: Das Kontextrekonstruktionssystem

4.4.4 Modellierung des Systems zur Kontextrekonstruktion

Die Modellierung des Systems zur Kontextrekonstruktion ergibt sich aus der Dekomposition der benötigten Funktionalitäten (siehe Abb. 4.17): Jeder Interaktionsbereich wird in der Präsentationsschicht um einen `ContextRequester` erweitert, der über eine geeignete Eingabeaktion im `View` (z. B. den Druck auf einen Button), angesprochen wird. Der `ContextRequester` fragt das vom *History*-System bereitgestellte `ActionVersionMapping` nach dem globalen Index für die vom Nutzer selektierte (und über die Eingabeaktion identifizierbare) Aktion bzw. Version. Dieser Index wird dem auf der Präsentationsschicht angesiedelten `ContextController` mitgeteilt, welcher im Modell der Rekonstruktionskomponente, dem `ContextSynchronizer`, den neuen Index einträgt. Der `ContextSynchronizer` wird von den `ContextModels` der Interaktionsbereiche über das *Observer*-Entwurfsmuster beobachtet. Das Eintragen des neuen Indexes wird dadurch von den `ContextModels` detektiert. Dort wird über das jeweilige `ActionVersionMapping` die für die im Werkzeug zur Darstellung des Kontextes notwendigen Aktionen ermittelt und gegebenenfalls der `ContextView`, der für die Darstellung des Kontextes zuständig ist, aktualisiert.

Die Rekonstruktion des Kontextes soll natürlich nur im komplementären Interaktionsbereich erfolgen. Dies ist entweder dadurch realisierbar, dass jedes `ContextModel` die Existenz einer Aktion mit dem aktuellen globalen Index prüft. Ist dies der Fall, dann entstammt die Nutzeraktion dem zugehörigen Interaktionsbereich und es muss keine Kontextrekonstruktion erfolgen. Oder aber es erfolgt bereits innerhalb der Rekonstruktionskomponente eine Filterung der zu benachrichtigenden `ContextModel`. Dies kann erreicht werden, indem der `ContextRequester` die „Herkunft“ kodiert, welche dann vom `ContextSynchronizer` bei der Auswahl des zu benachrichtigenden `ContextModels` ausgewertet wird.

Da das Rekonstruktionssystem die zum globalen Index gehörende Aktion bzw. Version über das `ActionVersionMapping` der `History`-Komponente ermittelt, können an dieser Stelle auch gegebenenfalls ältere Versionen nachgeladen wer-

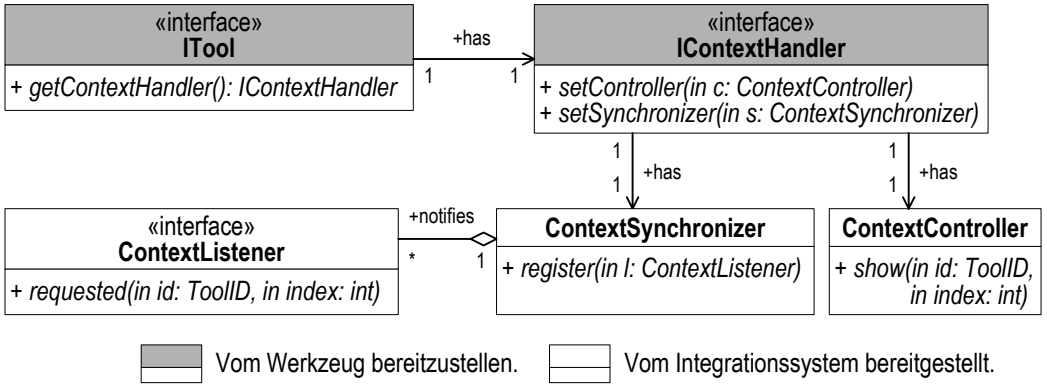


Abbildung 4.18: Klassendiagramm der für die Schnittstelle zwischen den Werkzeugen und dem System zur Kontextrekonstruktion relevanten Klassen

den. Dies ist zum Beispiel der Fall, wenn zu einer Artefaktversion der zugehörige Chatdiskurs angezeigt werden soll, dieser aber noch nicht in die Laufzeitumgebung geladen wurde. Da die Abfrage der Historie asynchron über einen *Callback*-Mechanismus erfolgt, muss auch entsprechend das `ContextModel` mit einem *Callback* über das Eintreffen der angefragten älteren Version bzw. Aktion benachrichtigt werden.

Abbildung 4.18 zeigt die für die Kommunikation zwischen den Werkzeugen und dem System zur Kontextrekonstruktion relevanten Klassen. Die Werkzeuge müssen in ihrem `ITool`-*Interface* eine Methode zum Zugriff auf den werkzeugeigenen `IContextHandler` bereitstellen. Dieser wird wiederum entsprechend dem *Dependency-Injection*-Entwurfsmuster mit dem `ContextController` und dem `ContextSynchronizer` initialisiert, wobei er sich über einen `ContextListener` am `ContextSynchronizer` registriert. Der `ContextSynchronizer` wird immer dann notifiziert, wenn der Kontext zu einer Aktion hergestellt werden soll. Der `ContextController` stellt aus Sicht der Werkzeuge die `show`-Methode zur Verfügung, mit der die Werkzeuge die Kontextrekonstruktion anfordern können.

Die Funktionalitäten zur Kontextrekonstruktion erlauben es, zu einer Aktion (einen Beitrag im Chat oder eine Artefaktversion) den Interaktionskontext zu ermitteln. Im Folgenden wird dargestellt, wie die im Kontext eines Chatbeitrags erfolgten Aktionen im gemeinsamen Arbeitsbereich innerhalb des Chattranskripts dokumentiert werden können.

4.5 Bereichsübergreifende Aktivitätsdokumentation

4.5.1 Einleitung

Die Kombination des quasi-synchronen Mediums Chat mit einem Konstruktionswerkzeug, bei dem jederzeit Änderungen des Zustands der gemeinsamen Arte-

fakte durch simultane Aktionen der anderen Teilnehmer möglich sind, kann dazu führen, dass diese Änderungen der Aufmerksamkeit entgehen. Deshalb wird in Anforderung **I4** gefordert, dass die Aktionen analog zu den Chatbeiträgen im Chattranskript dokumentiert werden. Dies zu realisieren ist Aufgabe des Systems zur bereichsübergreifenden Aktivitätsdokumentation. Damit werden im Chatwerkzeug sowohl die Chatbeiträge als auch die Aktionen im Konstruktionswerkzeug in ihrer zeitlichen Abfolge angezeigt, womit im Übrigen auch der vollständige Interaktionsverlaufs dokumentiert wird.

Die folgenden Abschnitte gehen auf die Konzeption der Aktivitätsdokumentation und die Dekomposition der Funktionalitäten ein. Anschließend erfolgt die Modellierung des Systems zur bereichsübergreifenden Aktivitätsdokumentation.

4.5.2 Konzeption der Aktivitätsdokumentation

Die Dokumentation der im Konstruktionswerkzeug erfolgten Aktionen durch das Chatwerkzeug erfordert dort Modifikationen: erstens müssen die Aktionen im Chattranskript geeignet visualisiert und hierfür zweitens die Aktionen in das Datenmodell des Chatwerkzeugs integriert werden.

Die Visualisierung der Aktionen kann z. B. durch eine textuelle Beschreibung oder eine graphische Darstellung der Aktionen erfolgen (siehe Kap. 5.6). Die Art und Weise der Visualisierung wie auch der Aufwand zur Umsetzung wird maßgeblich vom Funktionsumfang der gewählten Programmiersprache und der damit zur Verfügung stehenden Bibliotheken beeinflusst. Es soll deshalb im Folgenden der Fokus auf dem zweiten Aspekt der Modifikationen liegen, nämlich wie die Aktionen in das Datenmodell des Chatwerkzeugs integriert werden können.

Hierbei wird (wie auch schon in den vorangegangenen Ausführungen zu den integrierten Historien und zur Kontextrekonstruktion) davon ausgegangen, dass das Chatwerkzeug über ein Datenmodell (im Folgenden `ChatModel` genannt) verfügt, welches die Chatbeiträge in ihrer durch die Empfangsreihenfolge vorgegebenen Ordnung umfasst. Durch den Empfang neuer Beiträge wird die Liste der Chatbeiträge am einen Ende und durch das Laden von älteren Diskurssequenzen aus der Historie am anderen Ende erweitert.

Für die Dokumentation der im Konstruktionswerkzeug erfolgten Aktionen bietet es sich an, das `ChatModel` um die Aktionen anzureichern. Hierzu müssen die Aktionen jedoch in einer zu den Chatbeiträgen analogen Datenstruktur vorliegen, so dass sie vom `ChatModel` direkt verarbeitet werden können.

An welcher Stelle müssen die Aktionen nun aber im `ChatModel` eingefügt werden? Für die über den Nachrichtenkanal empfangenen Aktionen gilt, dass diese direkt an die Liste der Chatbeiträge angefügt werden können, denn der Nachrichtenkanal sichert eine bereichsübergreifende totale Ordnung aller Nachrichten zu. Für die Aktionen, die aus der Artefakthistorie geladen werden, gilt, dass sie entsprechend der globalen Indizes in die Folge der (ebenfalls global indizierten) Chatbeiträge einsortiert werden können. Hierbei ist jedoch zu beachten, dass die geladenen Teile der Diskurshistorie sowie der Artefakthistorie nicht vollständig über-

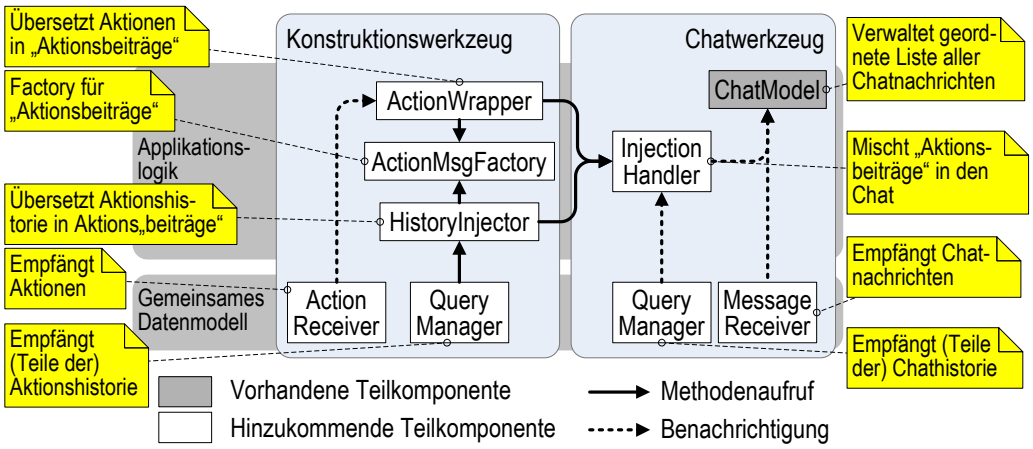


Abbildung 4.19: Schematische Darstellung der Aktionsmarkierung

lappend sein müssen, z. B. kann bereits die vollständige Artefakthistorie, aber nur ein Teil des vergangenen Chatdiskurses in die Laufzeitumgebung geladen sein.

4.5.3 Funktionale Dekomposition

Somit lassen sich die Funktionalitäten zur bereichsübergreifenden Aktivitätsdokumentation aufspalten in:

Konvertierung der Aktionsnachrichten: Die Aktionen müssen in eine zu den Chatbeiträgen analoge Datenstruktur konvertiert werden.

Einfügen empfangener Aktionen: Die vom Konstruktionswerkzeug über den Nachrichtenkanal empfangenen Aktionen müssen an das Chatwerkzeug übermittelt werden.

Einfügen von Aktionen aus der Historie: Analog müssen die aus der Konstruktionshistorie geladenen Aktionen an das Chatwerkzeug übermittelt werden.

Mischen der Historien: Die aus der Historie geladenen Aktionen müssen abhängig von ihrer zeitlichen Überlappung mit der Chathistorie in den Chatdiskurs integriert werden.

4.5.4 Modellierung des *ActionMarker*-Systems

Damit ergibt sich eine recht einfache Struktur des Systems zur bereichsübergreifenden Aktivitätsdokumentation (siehe Abb. 4.19): Im Konstruktionswerkzeug wird vom dortigen `ActionReceiver`, der bereits im Rahmen des *History*-Systems zum Einsatz kam (siehe Kap. 4.3.4), jede eintreffende Aktionsnachricht an den `ActionWrapper` gereicht. Dieser nutzt entsprechend des *Factory*-Entwurfsmusters (Gamma et al., 1995) eine `ActionMsgFactory`, um die Aktionsnachricht in ein

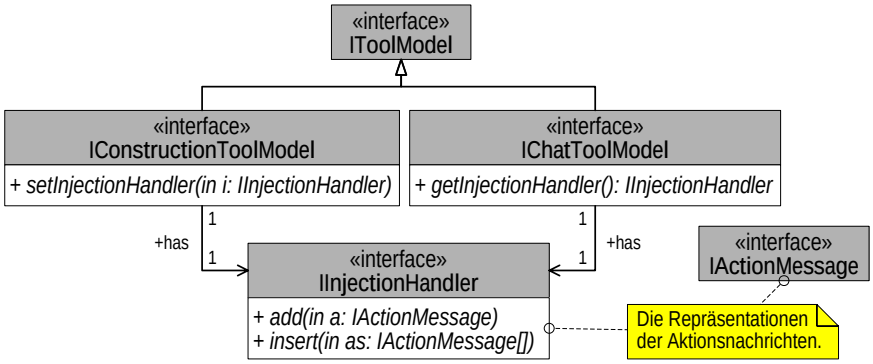


Abbildung 4.20: Klassendiagramm der für die Schnittstelle zwischen den Werkzeugen und dem `ActionMarker`-System relevanten Klassen

Austauschformat für die Kommunikation zwischen den Werkzeugen umzukodieren. Diese wird dem `InjectionHandler` auf Seiten des Chatwerkzeugs übergeben, welcher sie dann gegebenenfalls in eine vom `ChatModel` erwartete Datenstruktur umwandelt und in das `ChatModel` einfügt.

Analog werden im Konstruktionswerkzeug die vom `QueryManager` des *History*-Systems empfangenen Aktionsnachrichten aus der Historie behandelt. Der `QueryManager` reicht sie an den `HistoryInjector` weiter, welcher wiederum mittels der `ActionMsgFactory` die Aktionen konvertiert und an den `InjectionHandler` des Chatwerkzeugs übergibt. Dort erfolgt das dynamische Mischen der Historien: Alle übergebenen Aktionsrepräsentationen werden – sofern sie vom momentan verfügbaren Teil der Chathistorie abgedeckt werden – an die über den globalen Index determinierte Stelle einsortiert. Hierbei müssen gegebenenfalls Aktionen, die älter sind als der angezeigte Diskurs, zwischengespeichert und erst wieder berücksichtigt werden, wenn über den `QueryManager` des Chatwerkzeugs Teile der Diskurshistorie nachgeladen werden.

Wie in Abbildung 4.19 zu sehen, reduziert sich die Aufgabe des Integrationssystems auf die Herstellung der Verbindung zwischen Konstruktionswerkzeug und dem `InjectionHandler` des Chatwerkzeugs. Das wird in der einfachen Schnittstelle, die sich aus den in Abbildung 4.20 zu sehenden Klassen zusammensetzt, deutlich. Das Modell des Chatwerkzeugs gibt in seinem Modell vom Typ `IChatToolModel` Zugriff auf den `IInjectionHandler`, welcher im Modell des Konstruktionswerkzeugs (`IConstructionToolModel`) gesetzt werden kann.

Bei der Visualisierung der Aktionen im Chatwerkzeug kann das im folgenden Abschnitt vorgestellte Referenzierungssystem genutzt werden. Hierfür wird bei der Konvertierung einer Aktion durch die `ActionMsgFactory` das resultierende Datenobjekt z. B. um eine Referenz auf die durch die Aktion erzeugte Artefaktversion und die von der Aktion manipulierten Artefaktbestandteile angereichert. Vom Referenzierungssystem wird dann die Aktionsrepräsentation wie ein „normaler“ Chatbeitrag mit einer expliziten Referenz behandelt.

4.6 Explizite Referenzen

4.6.1 Einleitung

Die Problemanalyse in Kapitel 2 zeigt die Schwierigkeit bei der Kommunikation in dualen Interaktionsräumen auf, im Chatdiskurs auf Objekte im gemeinsamen Arbeitsbereich zu verweisen. Bereits die Etablierung von Bezügen innerhalb der Chatkommunikation selbst ist problematisch. Daraus resultiert die Forderung nach expliziten Referenzen als ein zusätzliches Ausdrucksmittel, mit dem deiktische Bezüge leichter durch die Nutzer herzustellen sind.

Im folgenden Abschnitt wird das Konzept expliziter Referenzen genauer beleuchtet. Anschließend erfolgt eine Dekomposition der für die Referenzierung notwendigen Funktionalitäten, bevor dann auf die Modellierung des Referenzierungssystems eingegangen wird.

4.6.2 Konzeption expliziter Referenzen

Eine explizite Referenz ist eine Verknüpfung eines Chatbeitrags mit einem Objekt in der gemeinsamen Kollaborationsumgebung. Hierbei sind entsprechend der Anforderungen **C 2** und **I 2** vier Fälle zu unterscheiden:

- Die Verknüpfung mit einem anderen Chatbeitrag (siehe Abb. 4.21.a).
- Die Verknüpfung mit einem Teil (z. B. einem Wort oder einem Satz) eines anderen Chatbeitrags (4.21.b).
- Die Verknüpfung mit einem Objekt im gemeinsamen Arbeitsbereich des Konstruktionswerkzeugs (4.21.c).
- Die Verknüpfung mit einem Ausschnitt des gemeinsamen Arbeitsbereichs (4.21.d).

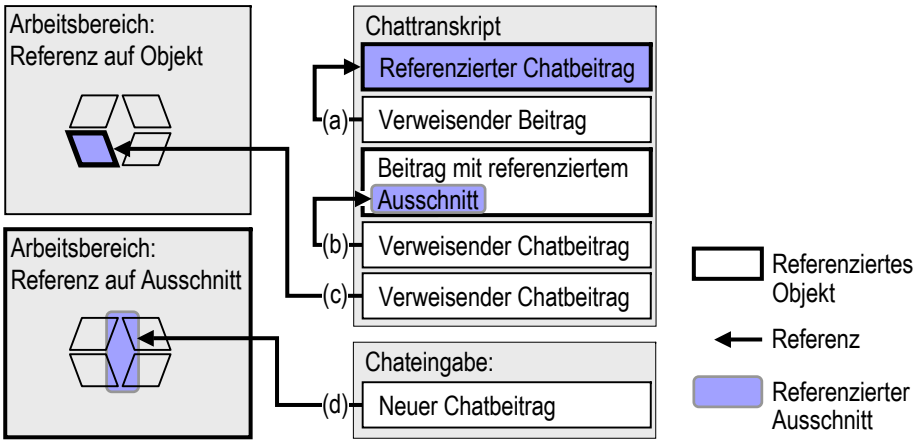


Abbildung 4.21: Varianten der Referenzierung

Welche referenzierbaren Objekte ein Konstruktionswerkzeug bereitstellt und wie innerhalb des gemeinsamen Arbeitsbereichs sinnvoll ein Ausschnitt bzw. Detail definiert werden kann, hängt stark von der Art der gemeinsamen Artefakte ab. Sind dies zum Beispiel wie in einem *Shared Whiteboard* graphische Darstellungen, die bei allen Kollaborationspartnern auf dieselbe Art dargestellt werden, so kann das Zeigen auf einen Ausschnitt über eine geometrisch festgelegte Region des gemeinsamen Arbeitsbereichs erfolgen. Sind die Artefakte jedoch zum Beispiel Texte, die je nach Größe der Benutzungsschnittstelle bei den Teilnehmern unterschiedlich umbrechen können, muss die Definition eines Ausschnitts eher der internen Struktur des angezeigten Dokuments folgen.

Die expliziten Referenzen sind ein zusätzliches Ausdrucksmittel für das Medium Chat. Alle sonstigen Eigenschaften des Mediums, wie zum Beispiel die Darstellung der Nachrichten in ihrer zeitlichen Ordnung oder die zeitliche Trennung von nicht beobachtbarer Produktion und Rezeption, sollen davon unberührt bleiben.

Für die explizite Referenzierung sind drei primäre Funktionalitäten notwendig: Die Erzeugung, die Übertragung und die Anzeige von Referenzen. Diese sollen nun genauer beleuchtet werden, so dass die konkreten Eigenschaften der Funktionalitäten deutlich werden:

Erzeugung von Referenzen Explizite Referenzen dienen als zusätzliches Ausdrucksmittel der leichteren Formulierung von Chatbeiträgen. Sie werden demzufolge während der Erstellung eines Chatbeitrags erzeugt. Wie die Diskussionen der *Grounding Theory* in Kapitel 2.3.1 sowie der Chatkommunikation in Kapitel 2.3.2 gezeigt haben, tendieren die Kommunizierenden dazu, den Aufwand für einen Beitrag zu minimieren. Die Erzeugung zusätzlicher expliziter Referenzen muss deshalb für die Nutzer mit einem geringen Aufwand verbunden sein. Hierfür sollten die Eingabemöglichkeiten den in den momentanen Benutzungsschnittstellen populären Mitteln der direkten Manipulation über die Maus sowie der Auswahl über die Cursortasten folgen.

Die expliziten Referenzen sind Teil eines Chatbeitrags. Ihre „Veröffentlichung“ und Übertragung folgt deshalb den Mechanismen der Chatkommunikation: Die Erstellung und Bearbeitung ist durch die anderen Teilnehmer nicht beobachtbar, die Referenzen werden erst zusammen mit der Chatnachricht übertragen.

Übertragung und Empfang von Referenzen Hat ein Teilnehmer einen Chatbeitrag mit expliziten Referenzen versehen und verschickt den Beitrag, so müssen auch die expliziten Referenzen übermittelt werden. Wird nach dem Empfang die eingetroffen Nachricht angezeigt, so müssen auch die zugehörigen Referenzen präsentiert werden, da die mit ihnen formulierten Bezüge zum Verstehen der Nachricht notwendig sind.

Anzeige von Referenzen Analog zum geringen Aufwand zur Referenzerzeugung muss angestrebt werden, den Aufwand zur Rezeption gering zu hal-

ten. Dieser wird wesentlich dadurch beeinflusst, wie die Referenzen angezeigt werden. Wie in Kapitel 2 ausgeführt, stellen das Chat- und das Konstruktionswerkzeug zwei visuell getrennte Bereiche der Benutzungsschnittstelle dar. Eine explizite Referenz dient dazu, in einem Chatbeitrag auf gemeinsame Artefakte zu „zeigen“. Der Empfehlung von van Bruggen et al. (2002) (siehe S. 38) folgend, dass die Rezipienten bei der semantischen Integration durch eine geeignete visuelle Integration unterstützt werden müssen, sollte die Referenz zwischen einer referenzierenden Nachricht und dem referenzierten Objekt durch eine visuelle Verknüpfung, z. B. durch eine verbindende Linie, dargestellt werden. Wie genau die graphische Darstellung benutzerfreundlich erfolgen muss, geht über den Rahmen dieser Arbeit hinaus.[18] In der softwaretechnischen Konzeption muss jedoch berücksichtigt werden, dass die Darstellung einer Referenz einer die Interaktionsbereiche überspannenden Visualisierung bedarf.

Da die expliziten Referenzen mit der üblichen Darstellung des Chattranskripts als zeitlich geordnete Liste der eingetroffenen Nachrichten kombiniert werden sollen, ergibt sich das Problem, dass nicht immer alle Referenzen sichtbar sein können, da ansonsten – wie Abbildung 4.21 bereits erahnen lässt – die Lesbarkeit stark leidet. Insofern müssen Nachrichten, die ausgehende Referenzen besitzen, im Chattranskript geeignet markiert werden, und es muss die Anzeige dieser Referenzen durch die Nutzer steuerbar sein.

Entsprechend Anforderung **I 3** muss bei den Referenzen auf den gemeinsamen Arbeitsbereich beachtet werden, dass die dortigen Artefakte im Laufe der Kollaboration erzeugt, verändert und gelöscht werden können. Zeigen Referenzen auf diese Artefakte, können sich der referenzierte und der aktuell beim Benutzer angezeigte Zustand unterscheiden. Dies muss bei der Darstellung der Referenz deutlich werden. Hierzu muss jedoch in der Referenz der Zustand des gemeinsamen Arbeitsbereichs kodiert sein, so dass die Diskrepanz zwischen angezeigter und referenzierter Version detektiert und entsprechend visualisiert werden kann.

4.6.3 Funktionale Dekomposition

Die eben beschriebenen Funktionalitäten des Referenzierungssystems lassen sich über eine Reihe von Diensten realisieren (siehe Abb. 4.22), welche einerseits von der zentralen Referenzierungskomponente für das Konstruktions- und das Chatwerkzeug und andererseits von den beiden Werkzeugkomponenten für die Referenzierungskomponente bereitgestellt werden:

Alle drei Komponenten bieten auf der Präsentationsschicht Dienste zur *Referenzanzeige* an, wobei die zentrale Referenzierungskomponente die wesentlichen

[18] Auch wenn das in Kapitel 5 vorgestellte System zur Referenzdarstellung die verbindende Linie verwendet und die Erfahrungen damit positiv sind, kann aus Ermangelung vergleichender Studien nicht ausgeschlossen werden, dass es benutzerfreundlichere Darstellungen gibt.

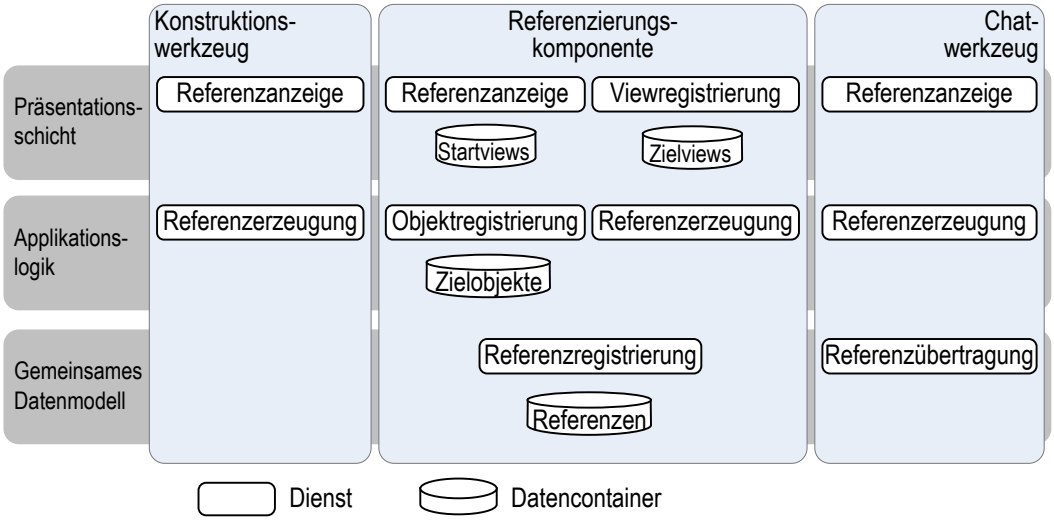


Abbildung 4.22: Konzeptuelles Modell der Referenzierungskomponente

Aufgaben zur Berechnung der verbindenden Referenzdarstellungen übernimmt. Hierzu benötigt sie Angaben darüber, welche *Views* den referenzierenden Chatbeitrag und das referenzierte Objekt an welchen Positionen anzeigen. Diese werden über die bei der *Viewregistrierung* angemeldeten Start- und Zielviews ermittelt. Ein Startview ist hierbei eine GUI-Komponente, welche Chatbeiträge mit ausgehenden Referenzen anzeigt, also die das Chattranskript zeigenden Komponenten sowie auch der Eingabebereich für neue Chatbeiträge. Analog ist ein Zielview eine GUI-Komponente, welche referenzierbare Objekte anzeigt, also die das Chattranskript zeigende Komponente sowie die Komponente des Konstruktionswerkzeugs, die die gemeinsamen Artefakte anzeigt.

Ebenfalls bieten alle drei Komponenten Dienste zur *Referenzerzeugung* an. Dies betrifft seitens der beiden Werkzeuge Funktionalitäten zur Auswahl der zu referenzierenden Objekte und von artefaktspezifischen referenzierbaren Ausschnitten. In der zentralen Referenzierungskomponente erfolgt die Kodierung der Verknüpfung, welche dann durch den vom Chatwerkzeug angebotenen Dienst zur *Referenzübertragung* übermittelt wird. Dieser Dienst befindet sich auf der untersten Schicht – des gemeinsamen Datenmodells. Hier sind die Funktionalitäten angesiedelt, die dem Austausch der Referenzen dienen. Wird seitens des Chatwerkzeugs eine Nachricht mit Referenzen empfangen, so werden diese vom Chatwerkzeug bei der *Referenzregistrierung* angemeldet, welche sie an die darüber liegende Schicht weiter reicht, wo die Referenz mit den lokalen Repräsentationen des referenzierenden und des referenzierten Objekts abgeglichen wird. Hierzu bedarf es Kenntnis über diese in der Laufzeitumgebung verfügbaren Objekte. Erreicht wird dies durch die *Objektregistrierung*, bei der das Chat- und das Konstruktionswerkzeug die jeweiligen referenzierbaren Zielobjekte anmelden.

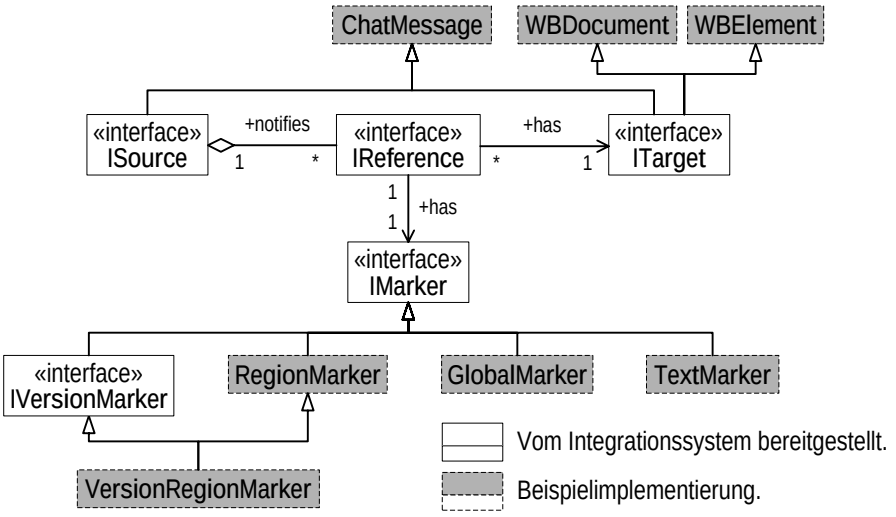


Abbildung 4.23: Klassendiagramm für Referenzen

4.6.4 Modellierung des Referenzierungssystems

Das Referenzierungssystem ist umfangreicher als die für die anderen Integrationsmaßnahmen notwendigen Systeme. Deshalb geschieht die Modellierung in folgenden aufeinander aufbauenden Schritten: Begonnen wird mit der Darstellung, wie Referenzen repräsentiert werden können. Darauf basierend wird dann dargelegt, wie Referenzen erzeugt, übertragen und angezeigt werden. Für die leichtere Darstellung wird zuerst die Referenzanzeige, anschließend die Erzeugung und schlussendlich die Übertragung betrachtet.

Modellierung expliziter Referenzen

Wie können nun explizite Referenzen modelliert werden? Wie in Kapitel 4.6.1 ausgeführt, ist eine Referenz eine gerichtete Verknüpfung zwischen zwei Objekten: ausgehend von einer Chatnachricht als Startobjekt hin zu einem Zielobjekt – einem beliebigen referenzierbaren Objekt innerhalb der gemeinsamen Arbeitsumgebung bzw. einem älteren Chatbeitrag. Da mit einer Referenz auch auf Ausschnitte gezeigt werden kann, enthält die Referenz darüber hinaus eine Ausschnittsbeschreibung.

Dies kann direkt in eine Reihe von Klassen und *Interfaces* übertragen werden (siehe das Klassendiagramm in Abb. 4.23):

`ISource`**:** Das referenzierende Objekt implementiert das *Interface* `ISource`. Es besitzt eine (u. U. leere) Menge von Referenzen.

`IReference`**:** Die Referenz kennt das referenzierende sowie das referenzierte Objekt und besitzt darüber hinaus eine Ausschnittsbeschreibung vom Typ `IMarker`.

`IMarker`: Von diesem *Interface* leiten sich alle unterschiedlichen Ausschnittsbeschreibungen ab. Je nach Art des referenzierten Dokumentes sind unterschiedliche Beschreibungen notwendig. Im Klassendiagramm sind drei Beispiele hierfür angegeben: Ein `GlobalMarker` beschreibt eine Referenz auf das gesamte Objekt, ein `TextMarker` auf einen Textbestandteil, wie es z. B. für die Referenzierung auf Teile eines Chatbeitrags notwendig ist, und ein `RegionMarker` beschreibt die Referenzierung eines rechteckigen Bereichs, wie es z. B. für Graphiken sinnvoll ist.

Die Ausschnittsbeschreibung ist auch die Stelle, an der gegebenenfalls die „Version" der gemeinsamen Artefakte gespeichert wird (siehe Anforderung **I 3** auf S. 40). So enthält z. B. der `VersionRegionMarker` für den Verweis auf einen rechteckigen Bereich eines *Shared Whiteboards* nicht nur die geometrischen Daten, sondern kodiert auch die referenzierte Artefaktversion.

`ITarget`: Dieses *Interface* wird von dem referenzierten Objekt implementiert.

Wie in Abbildung 4.23 veranschaulicht, ist jeder einzelne Chatbeitrag potentiell sowohl ein Start- wie auch ein Zielobjekt, er kann ausgehende Referenzen besitzen und Referenzen können auf diesen Beitrag verweisen. Desweiteren ist beispielhaft angegeben, dass in einem *Shared Whiteboard* jedes Artefaktelement wie auch das gesamte Dokument ein potentielles Zielobjekt ist, auf das referenziert werden kann.

Die vier *Interfaces* `ISource`, `ITarget`, `IReference` und `IMarker` bilden die Grundlage für die Anzeige, Erzeugung und Übertragung der Referenzen. Auf diese wird nun im Einzelnen eingegangen.

Anzeige von Referenzen

In der Anzeige der Referenzen liegt die eigentliche Komplexität der Referenzierungsfunktion, da Referenzen aus dem Chat in den gemeinsamen Arbeitsbereich als Verbindungen visualisiert werden sollen. Dies hat zur Konsequenz, dass die Berechnung der Referenzdarstellung und deren Zeichnen mehrere *Views* betreffen kann (siehe z. B. die als verbindenden Pfeil dargestellte Referenz in Abb. 4.24):

Berechnung der Referenzdarstellung: Die Referenzdarstellung ist abhängig von der Position des referenzierten Objekts innerhalb des Konstruktionswerkzeugs und der Position der referenzierenden Nachricht innerhalb des Chatwerkzeugs. Erstere ändert sich beispielsweise, wenn der sichtbare Ausschnitt verschoben oder die Größe des Werkzeugfensters verändert wird, letztere zum Beispiel beim Eintreffen einer neuen Nachricht, welche den referenzierenden Beitrag nach oben verschiebt. Jede solche Positionsänderung hat zur Folge, dass auch die Referenzdarstellung aktualisiert werden muss.

Zeichnen der Referenz: Entsprechend der in MVC-basierten Benutzungsschnittstellen üblichen Teil-Ganzes-Hierarchisierung, welche die Schnittstelle in

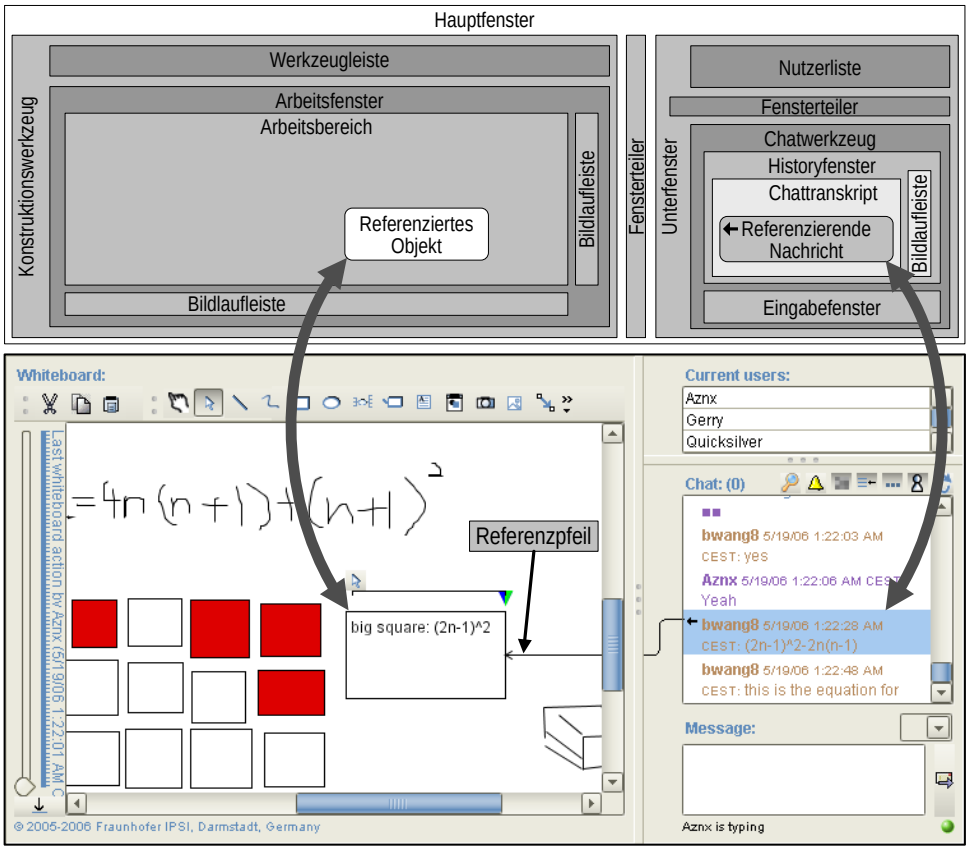


Abbildung 4.24: Einbettungshierarchie einer Benutzungsschnittstelle

nichtüberlappende Flächen aufteilt (siehe Abb. 4.24), wäre für das Zeichnen der Referenz diejenige Komponente zuständig, die sowohl die Darstellung der referenzierenden Chatbeiträge (und somit das Chatwerkzeug) als auch die der referenzierten Objekte (also das Konstruktionswerkzeug) beinhaltet. In der Abbildung ist dies das Hauptfenster. Wie dort auch zu sehen ist, kann die Referenzdarstellung aber nicht vollständig über die Benutzungsschnittstelle gezeichnet werden, denn diese beinhaltet nicht nur die eigentlichen Views des Chattranskripts und des gemeinsamen Arbeitsbereichs, sondern üblicherweise auch eine Reihe von Steuer- und Strukturierungselementen, wie z. B. Bildlaufleisten und Fensterteiler. Die Referenzdarstellung darf nun nicht über diese Steuer- und Strukturierungselemente hinweg gezeichnet werden.[19] Damit das vom Hauptfenster realisiert werden kann, müsste dieses wissen, wo die Referenzdarstellung *nicht* zu zeichnen ist. An dieser Stelle

[19] Die über Bildlaufleisten gesteuerten Ausschnittsanzeigen folgen der Fenstermetapher: durch das Fenster ist nur ein Ausschnitt der „dahinterliegenden" größeren Arbeitsfläche bzw. Chatanzeige sichtbar. Würde die Referenzdarstellung jedoch vollständig über die Steuerelemente gezeichnet, wäre dies ein Bruch mit der Metapher.

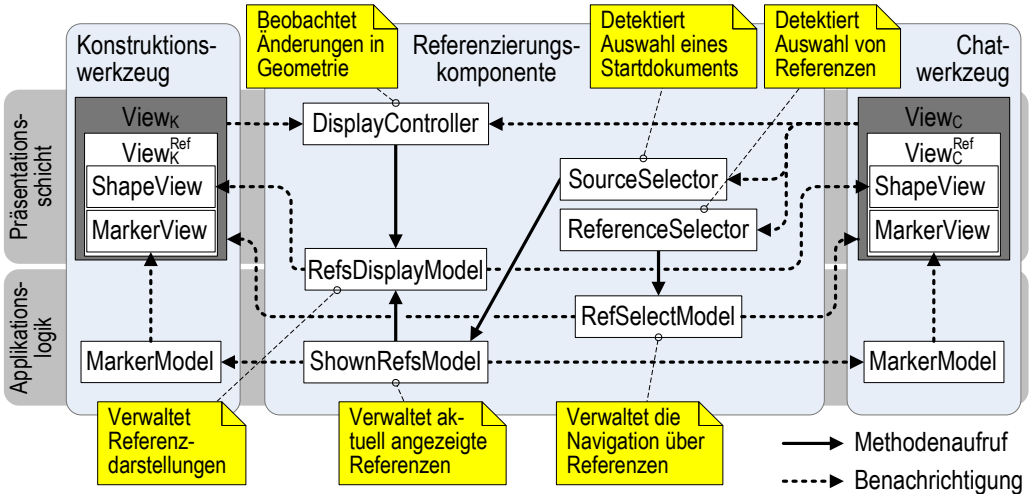


Abbildung 4.25: Schematische Darstellung des Anzeigesystems für die Referenzierung. Zur besseren Lesbarkeit sind nur modifizierende Methodenaufrufe dargestellt

ist es praktikabler, die Zuständigkeit an die Komponenten zu verlagern, in denen die Referenzdarstellung tatsächlich *gezeichnet* werden soll.

Abbildung 4.25 zeigt schematisch das Anzeigesystem zur Referenzierung, welches diese Aspekte berücksichtigt. Für die Referenzanzeige werden die Anzeigekomponenten des Konstruktionswerkzeugs ($\texttt{View}_\texttt{K}$) und des Chatwerkzeugs ($\texttt{View}_\texttt{C}$) um jeweils die Referenzierungsanzeige umsetzenden Teile $\texttt{View}_\texttt{K}^\texttt{Ref}$ und $\texttt{View}_\texttt{C}^\texttt{Ref}$ erweitert. Diese beinhalten

- einen `ShapeView`, welcher den über den *View* verlaufenden Teil der Referenzdarstellung zeichnet, sowie
- einen `MarkerView`, welcher – sofern eine Referenz auf einen Ausschnitt verweist – diesen hervorhebt.

Da Änderungen in der Geometrie der `Views` (wie z. B. aufgrund von Größenänderungen oder Scrolling) auch eine Änderung der Referenzdarstellung nach sich ziehen können, wird über das *Observer*-Entwurfsmuster ein bedarfsabhängiges Aktualisieren der Referenzdarstellungen umgesetzt. Hierzu beobachtet der `DisplayController` die beteiligten `Views` und löst gegebenenfalls im `RefsDisplayModel` eine Neuberechnung der momentan angezeigten Referenzdarstellungen aus. Die Änderung in den Referenzdarstellungen wird den `ShapeViews` mitgeteilt, so dass diese die entsprechenden Teile neu zeichnen können.

Das `RefsDisplayModel` spielt in diesem Zusammenhang eine Zwitterrolle. Es ist einerseits Datenmodell der angezeigten Referenzen, andererseits muss es notgedrungen sehr nah an der Präsentationsschicht sein. Um die Referenzdarstellun-

gen berechnen zu können, bedarf es Kenntnis über die jeweiligen internen Positionen von referenzierendem Chatbeitrag im `Chatview` und des referenzierten Objekts im anzeigenden `View` (dies kann sowohl der `View` des Konstruktionswerkzeugs sein, sofern die Referenz dorthin verweist, oder auch des Chatwerkzeugs, sofern die Referenz auf eine andere Nachricht zeigt).

Bezüglich dieser Aspekte sind das Chatwerkzeug und das Konstruktionswerkzeug völlig symmetrisch, da beide referenzierbare Objekte anzeigen. Das Chatwerkzeug hat jedoch als weitere Aufgabe die Anzeige der referenzierenden Objekte. Dies sind zum einen die referenzierenden Chatbeiträge im Chattranskript und zum anderen die aktuell vom Nutzer erstellte Nachricht, die von diesem mit (noch nicht veröffentlichten) Referenzen versehen sein kann.

Für erstere gilt, dass die Nutzer durch Auswahl referenzierender Chatbeiträge im Chattranskript die Anzeige der zugehörigen Referenzen auslösen können. Hierfür ist der `SourceSelector` zuständig. Er reagiert auf Nachrichten vom $\texttt{View}_{\texttt{C}}$ des Chatwerkzeugs und übersetzt die Eingabeereignisse in Modifikationen des `ShownRefsModels`, welches alle momentan angezeigten Referenzen verwaltet. Änderungen in diesem Modell werden a) den `MarkerModels` mitgeteilt, so dass die zugehörigen `MarkerViews` die entsprechenden Hervorhebungen aktualisieren, und ziehen b) natürlich auch Änderungen im `RefsDisplayModel` nach sich – es müssen für die nicht mehr angezeigten Referenzen die Referenzdarstellungen entfernt und für hinzukommende neue Darstellungen berechnet werden.

In der bisherigen Betrachtung des Anzeigesystems spielte der dynamische Charakter der Artefakte im Konstruktionswerkzeug keine Rolle. Wo wird dieser nun aber berücksichtigt? Zum einen durch die materialspezifischen `MarkerViews`. Diese sind für die Darstellung der `IMarker` (siehe Abb. 4.23, S. 99) zuständig und können hierbei die gegebenenfalls im `IMarker` enthaltene Versionsinformation auswerten. Zum anderen erlaubt der `ReferenceSelector` den Nutzern eine Interaktion mit den angezeigten Referenzen.

Im Chattranskript wird (zum Beispiel durch ein Icon) angezeigt, dass ein Beitrag ausgehende Referenzen besitzt. Klickt ein Benutzer auf einen referenzierenden Beitrag, so wird – wie oben beschrieben – über den `SourceSelector` die Anzeige der Referenzdarstellungen ausgelöst. Diese Anzeige verändert jedoch nicht die Darstellungen der Artefakte. Zeigt eine Referenz nun aber z. B. auf eine andere Version modifizierbarer Artefakte oder auf eine andere Seite mehrseitiger Dokumente, so müsste der Nutzer diese Version bzw. diese Seite im jeweiligen Werkzeug erst einstellen. Hier kommt nun der `ReferenceSelector` ins Spiel. Um der Referenz leicht folgen, aber auch zur momentanen Sicht zurückkehren zu können, wertet dieser z. B. wiederholte Mausklicks auf den Referenzindikator einer Chatnachricht aus und modifiziert das `RefSelectModel`, welches die `Views` der Zieldokumente informiert. Durch diesen Mechanismus kann auch zwischen den möglicherweise mehreren Referenzen einer Nachricht, die auf unterschiedliche Versionen zeigen können, gewechselt werden.

An der Anzeige der Referenzen sind drei zentrale Modelle beteiligt: das `ShownRefsModel` enthält die momentan angezeigten Referenzen, das `RefsDisplay-`

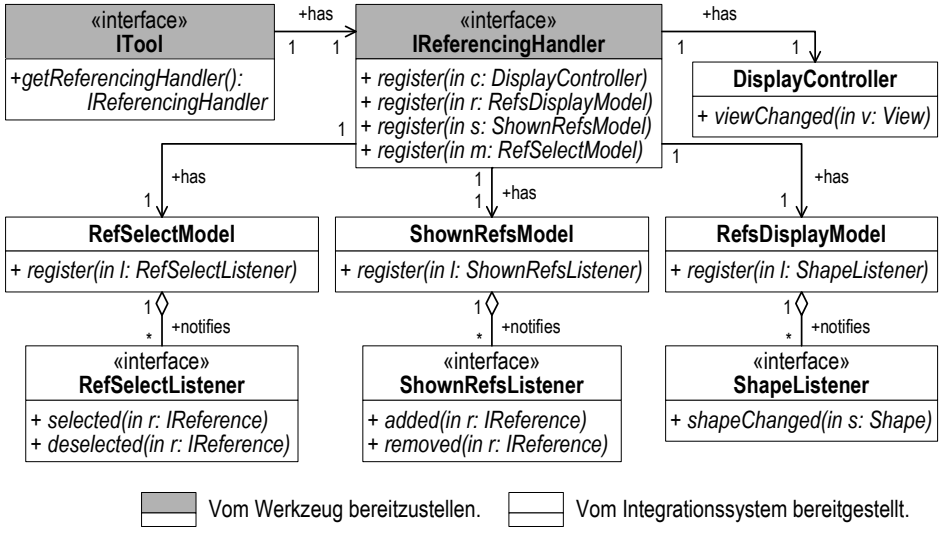


Abbildung 4.26: Klassendiagramm der für die Schnittstelle zwischen den Werkzeugen und dem Referenzanzeigesystem relevanten Klassen

`Model` die Beschreibung ihrer komponentenübergreifenden Darstellungen und das `RefSelectModel` die aktuell selektierte Referenz.

Über diese Modelle zusammen mit dem `DisplayController` erfolgt die Kommunikation zwischen den Werkzeugen und dem Referenzierungssystem. Dies wird deutlich in Abbildung 4.26. Dort sind die für die Schnittstelle relevanten Klassen und *Interfaces* zu finden. Werkzeugseitig wird ein `IReferencingHandler` erwartet, welcher die dortigen Referenzierungsfunktionalitäten bündelt. Für die Verknüpfung mit dem Referenzierungssystem enthält er vier Methoden zur Registrierung der genannten drei Modelle und des `DisplayControllers`.

Für die im Chatwerkzeug stattfindende Selektion von Startdokumenten und Referenzen bedarf es noch einer Ergänzung im `IReferencingHandler` des Chatwerkzeugs. Diese zeigt Abbildung 4.27: Das vom `IReferencingHandler` erbende *Interface* `IChatReferencingHandler` enthält zwei weitere Methoden, mit denen `Source-` und `ReferenceSelector` registriert werden können.

Erzeugung von Referenzen

Nachdem im vorangegangenen Abschnitt dargelegt wurde, wie die Referenzen angezeigt werden, soll nun das System zur Erzeugung von Referenzen vorgestellt werden (siehe Abb. 4.28).

Referenzen werden während der Eingabe eines Chatbeitrags durch die Benutzer erzeugt. Das referenzierende Objekt ist somit immer die gerade im Entstehen begriffene Nachricht. Für die Erzeugung muss durch die Nutzer das Referenzziel ausgewählt werden. Dies können Chatbeiträge im Chattranskript oder Objekte im

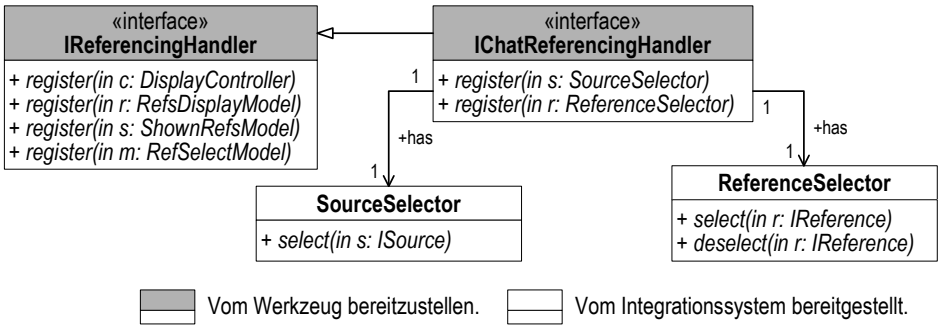


Abbildung 4.27: Klassendiagramm der zusätzlichen Schnittstellen zwischen Chatwerkzeug und dem Referenzanzeigesystem

gemeinsamen Arbeitsbereich bzw. jeweils Ausschnitte daraus sein. Wie in der Einleitung zu diesem Kapitel auf Seite 95 ausgeführt, muss der Erzeugungsaufwand für die Nutzer möglichst gering gehalten werden. Dies heißt, dass die Auswahl des Referenzziels mit kurzen Interaktionssequenzen verbunden sein muss. Hierfür bieten sich die etablierten Auswahlmechanismen an: die Selektion über die Tastatur, meist mittels der Richtungstasten, sowie die Selektion mit der Maus.

Für die Selektion über die Tastatur ist der `KeyTargetLocator` zuständig. Dieser übersetzt die Tastatureingaben in Aktionen, mit denen die im `TargetsModel` verfügbaren Zielobjekte traversiert werden können. Das `TargetsModel` enthält alle referenzierbaren Objekte in einer sinnvollen Reihenfolge, die sich an der Wahrscheinlichkeit der Referenzierung orientiert. So ist es wahrscheinlicher, dass auf

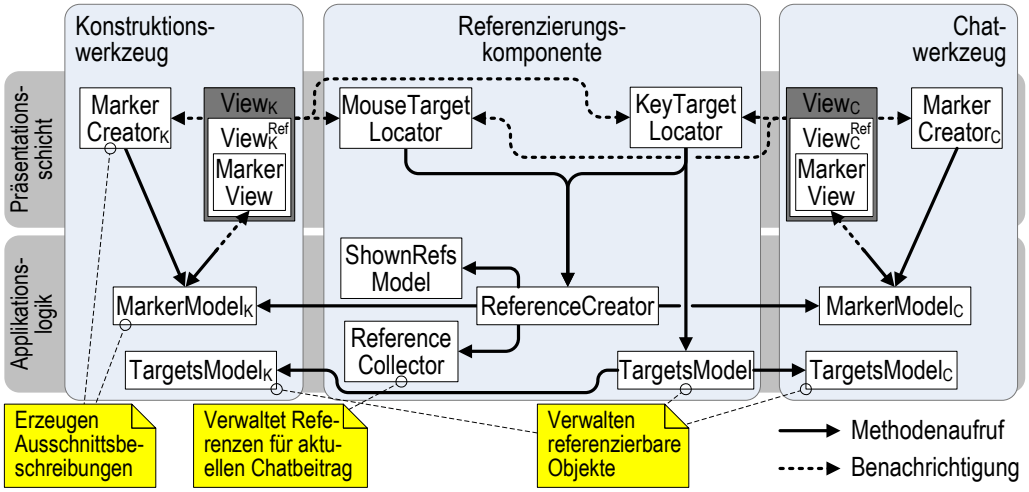


Abbildung 4.28: Schematische Darstellung des Erzeugungssystems für die Referenzierung

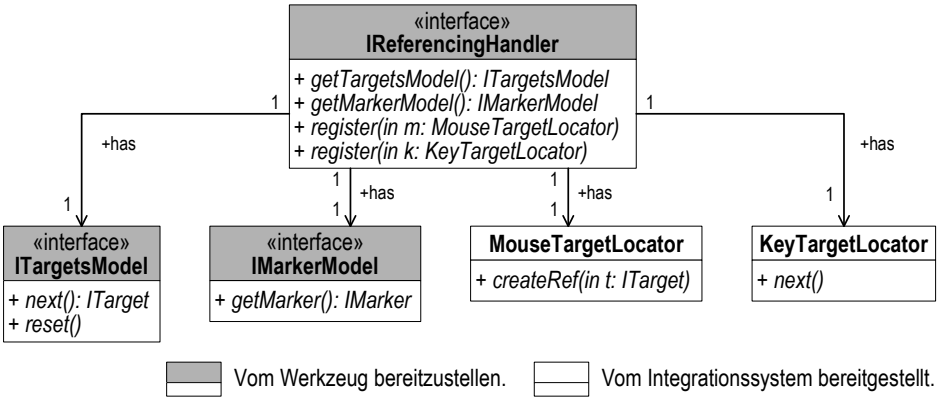


Abbildung 4.29: Klassendiagramm der für die Schnittstelle zwischen den Werkzeugen und dem Referenzerzeugungssystem relevanten Klassen

gerade eingetroffene Nachrichten referenziert werden wird, als auf ältere Nachrichten. Diese Sortierung wird durch die werkzeugspezifischen $\texttt{TargetsModel}_{\texttt{C}}$ und $\texttt{TargetsModel}_{\texttt{K}}$ realisiert, die vom globalen `TargetsModel` für die Traversierung genutzt werden. Hat der `KeyTargetLocator` das nächste referenzierbare Objekt ausgewählt, so wird dieses an den `ReferenceCreator` weitergereicht, welcher abhängig vom Typ des Zielobjektes das entsprechende `MarkerModel` aufruft, um für diese Referenz einen geeigneten `IMarker` erzeugen zu lassen. So muss der `IMarker` für eine Referenz auf einen manipulierbaren Arbeitsbereich die momentan angezeigte Version enthalten. Anschließend wird diese Referenz dem `ReferenceCollector` übergeben, welcher alle Referenzen für den aktuell vom Nutzer erstellten Chatbeitrag sammelt.

Der `ReferenceCollector` ist auch das Bindeglied zum Anzeigesystem. Ändern sich dessen Referenzen, so werden diese dem `ShownRefsModel` (siehe S. 102) direkt übergeben und somit zur Anzeige gebracht.

Für die Auswahl eines zu referenzierenden Objekts und insbesondere für die Selektion bestimmter Details bzw. Ausschnitte werden die Mausbewegungen ausgewertet. Da je nach Art des Objekts die Mausgesten verschieden interpretiert werden müssen (so wird üblicherweise ein Text anders selektiert als eine rechteckige Fläche), enthalten die Werkzeuge jeweils eigene `MarkerCreator`. Durch den `MarkerCreator` werden die Mausgesten in `IMarker` übersetzt, die im `MarkerModel` gespeichert werden. Dort stehen sie dann zur Verfügung, wenn durch den `MouseTargetLocator` der `ReferenceCreator` mit der Erzeugung einer neuen Referenz auf das korrespondierende Zielobjekt betraut wird (üblicherweise nach Beendigung der Mausgeste und dem Loslassen der Maustaste).

Für die Integration der Referenzerzeugung müssen die Werkzeuge über ihren `IReferencingHandler` sowohl ihr `TargetsModel` wie auch ihr `MarkerModel` nach außen hin anbieten. Dies ist in Abbildung 4.29 zu sehen. Darüber hinaus bedarf es zweier Methoden, mit denen die zentralen `Mouse-` und `KeyTarget-`

`Locators` an den relevanten `Views` der Werkzeuge registriert werden können. Auch dies erfolgt sinnvollerweise über den `IReferencingHandler`, um vom werkzeugspezifischen GUI-Aufbau zu abstrahieren.

Übertragung von Referenzen

Referenzen werden gleichzeitig mit den referenzierenden Chatbeiträgen übertragen. Erst wenn der Beitrag vom Ersteller abgesendet wird, müssen auch die gesetzten Referenzen verschickt werden. Trifft ein Chatbeitrag bei den Teilnehmern ein, so müssen auch die zugehörigen Referenzen eintreffen. Hierfür bietet es sich an, den vom Chatwerkzeug genutzten Übertragungskanal zu nutzen. So erlaubt zum Beispiel das zunehmend genutzte XML-basierte Protokoll XMPP IM (*Extensible Messaging and Presence Protocol – Instant Messaging and Presence*, Saint-Andre, 2004), ein Nachrichtenelement um eigene Elemente zu erweitern. Alternativ können die Referenzen auch direkt als Bestandteil des Nachrichtentexts vor dem Versenden kodiert und beim Empfang wieder extrahiert werden.

Im Datenmodell ist eine Referenz als eine Verknüpfung zwischen zwei Objekten, der referenzierenden Chatnachricht und dem referenzierten Zielobjekt, repräsentiert (siehe S. 99). Sowohl die lokalen Referenz- wie auch die Start- und Zielobjekte sind lokale Replikate. Wird nun eine Referenz an die Laufzeitumgebungen der anderen Teilnehmer übertragen, müssen die replizierten Referenzobjekte mit den dortigen Replikaten von Start- und Zielobjekt verknüpft werden. Die Verknüpfung mit dem Startobjekt ist bei der gemeinsamen Übertragung leicht herzustellen, denn die Chatnachricht und die Referenz werden gemeinsam empfangen.

Etwas mehr Aufwand ist nötig, um das Zielobjekt zu identifizieren. Hierzu werden die referenzierbaren Objekte mit Identifikatoren (kurz: IDs) versehen, die in allen Laufzeitumgebungen identisch und innerhalb des Kontextes eines Kollaborationsraums eindeutig sein müssen. Bei der Übertragung einer Referenz wird diese ID übermittelt, mit der dann auf Empfängerseite das zugehörige lokale Objekt ermittelt werden kann.

Diese IDs sind in (teil)replizierten Applikationen meist bereits vorhanden. Erlauben die Konstruktionswerkzeuge die gemeinsame Manipulation der Artefakte, müssen die betroffenen Artefaktelemente in allen Laufzeitumgebungen der Teilnehmer bestimmt werden können. Sind die Artefakte statische Dokumente, so sind diese meist über einen *Uniform Resource Identifier* (URI) beschrieben. Desgleichen müssen auch alle Chatbeiträge mit einer ID versehen sein. Um wieder XMPP als Beispiel heranzuziehen, kann dort jedes Nachrichtenelement mit einer ID versehen werden.[20]

Damit über die IDs die entsprechenden Zielobjekte vom Referenzierungssystem ermittelt werden können, besitzt dieses einen `ObjectManager` (siehe Abb. 4.30). Der `ObjectManager` wird vom `ChatModel`, welches die empfangenen Chatbeiträge verwaltet, und dem `ObjectModel` des Konstruktionswerkzeugs über neu

[20] Siehe die Spezifikation unter `http://www.xmpp.org/rfcs/rfc3920.html#rfc.section.9.1.3`.

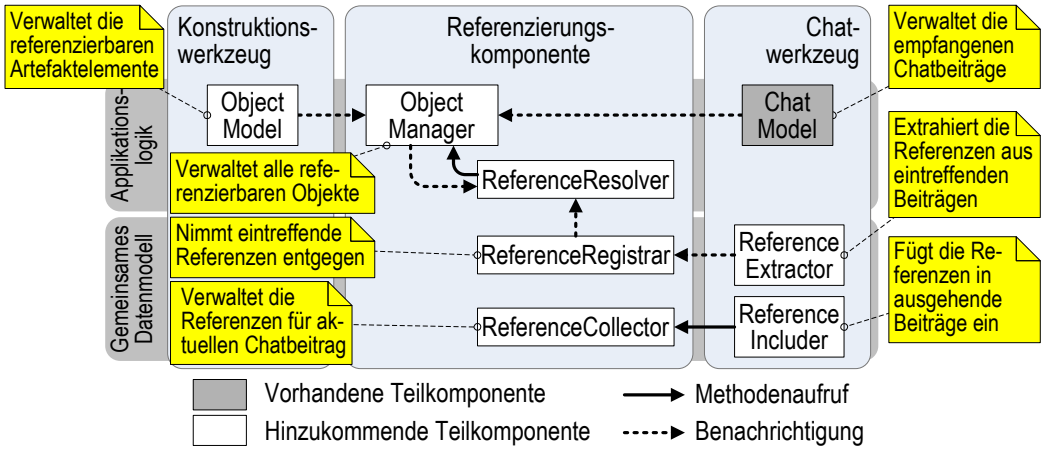


Abbildung 4.30: Schematische Darstellung des Übertragungssystems für die Referenzierung

erzeugte referenzierbare Objekte (die entsprechend vom Typ `ITarget` sind, siehe Abb. 4.23 auf S. 99) informiert. Jedes `ITarget`-Objekt besitzt eine eindeutige ID, so dass der `ObjektManager` die Abbildung ID → `ITarget` realisieren kann. Da der `ObjectManager` nur die innerhalb des Kollaborationsraums verfügbaren Objekte verwaltet, müssen die IDs entsprechend auch nur innerhalb des Raums eindeutig sein.

Wird nun im Chatwerkzeug ein Chatbeitrag verschickt, so wird durch den `ReferenceIncluder` im Chatwerkzeug vor der Übertragung die vom Erzeugungssystem im `ReferenceCollector` gesammelten Referenzen zum Beitrag (siehe den vorangegangenen Abschnitt zur Referenzerzeugung) in der Nachricht kodiert. Entsprechend werden auf Empfängerseite durch den `ReferenceExtractor` die Referenzen aus der eintreffenden Nachricht extrahiert und dem `ReferenceRegistrar` übergeben. Dieser reicht die Referenzen weiter an den `ReferenceResolver`, der mithilfe des `ObjectManagers` zu den Ziel-IDs die entsprechenden Zielobjekte ermittelt.

Hierbei ist ein Sonderfall zu beachten: Unter Umständen ist die Auflösung der Ziel-ID noch nicht möglich, weil das entsprechende Zielobjekt noch nicht in der Laufzeitumgebung des Empfängers bekannt ist. In diesem Falle muss der `ReferenceResolver` die noch nicht auflösbaren Referenzen zwischenspeichern. Treffen neue Objekte beim `ObjectManager` ein, benachrichtigt dieser den `ReferenceResolver`, so dass dieser die noch nicht aufgelösten Referenzen überprüfen kann.

Ein Zielobjekt ist in der Laufzeitumgebung eines Empfängers noch unbekannt, wenn es noch nicht aus der Historie geladen wurde (siehe Kap. 4.3). Insofern ist es naheliegend, in diesem Falle ein Nachladen der Historie auszulösen. Dies setzt jedoch voraus, dass anhand der Ziel-ID (denn mehr ist bisher nicht bekannt)

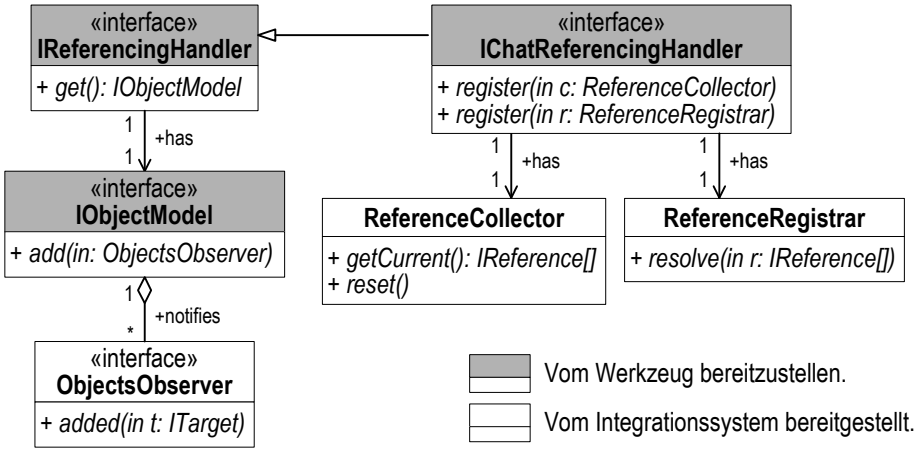


Abbildung 4.31: Diagramm der für die Schnittstelle zwischen den Werkzeugen und dem Referenzübertragungssystem relevanten Klassen

bestimmt werden kann, a) in welchem Werkzeug das Zielobjekt zu finden ist, und b) welche Informationen aus der Historie geladen werden müssen. Für Ersteres kann das Integrationssystem die vom Werkzeug gelieferte Ziel-ID um eine Werkzeug-ID anreichern. Für Zweiteres muss das Werkzeug selbst ausreichende Informationen in der Ziel-ID kodieren.

Abbildung 4.31 zeigt noch einmal die Schnittstelle zwischen den Werkzeugen und dem Referenzierungssystem beschränkt auf die Funktionalitäten zur Referenzübertragung. Der `IReferencingHandler` stellt nach außen hin das `IObjectModel` bereit, mit dem die Ziel-IDs aufgelöst werden können. An dieses Modell kann das Referenzierungssystem einen `ObjectsObserver` registrieren, um über die neu eintreffenden Objekte informiert zu werden. Der `IReferencing-Handler` des Chatwerkzeugs muss darüber hinaus noch zwei Methoden anbieten, über die der `ReferenceCollector` und der `ReferenceRegistrar` registriert werden können.

Schnittstelle des Gesamtsystems zur Referenzierung

Nachdem in den vorangegangenen Abschnitten die Subsysteme zur Referenzierung und die jeweils notwendigen Schnittstellen zwischen den Werkzeugen und dem Referenzierungssystem separat vorgestellt wurden, soll abschließend eine Gesamtsicht auf die notwendigen Schnittstellen geboten werden. Abbildung 4.32 zeigt, welche Schnittstellen die Werkzeuge für die Integration bezüglich der expliziten Referenzierung bereitstellen müssen:

Die Werkzeuge müssen über ihr `ITool`-*Interface* den Zugriff auf ihren jeweiligen `IReferencingHandler` erlauben. Der `IReferencingHandler` bündelt die für die Referenzierung notwendigen werkzeugseitigen Funktionalitäten. Da dem Chatwerkzeug für die Referenzerzeugung und -übertragung weitere Aufgaben

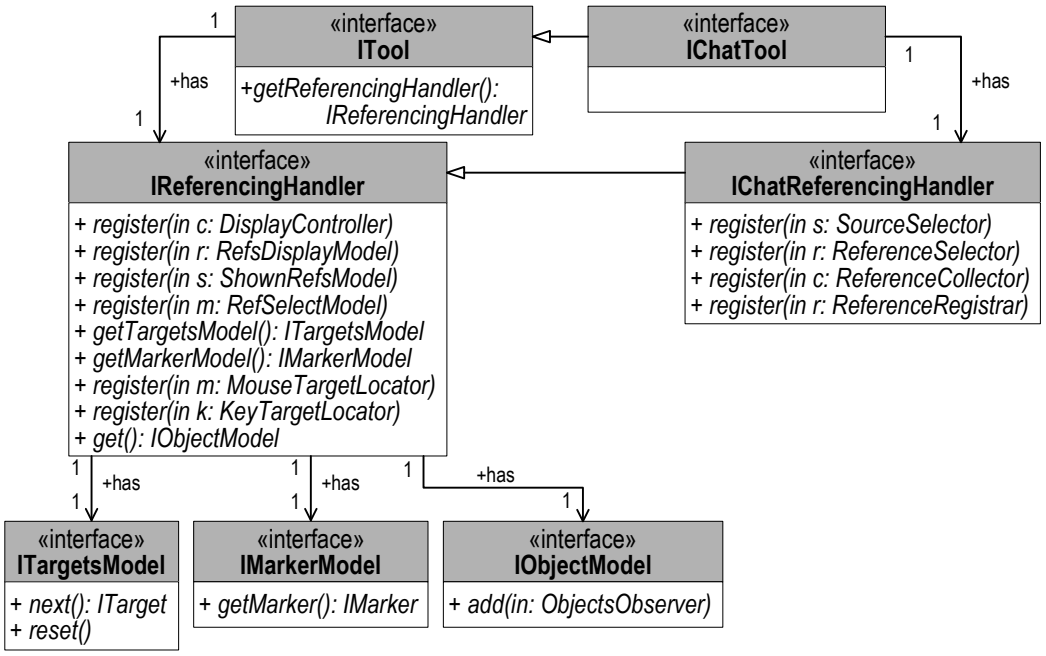


Abbildung 4.32: Diagramm der für die Schnittstelle zwischen den Werkzeugen und dem gesamten Referenzierungssystem relevanten Klassen

zukommen, muss es einen erweiterten `IReferencingHandler` bereitstellen, den `IChatReferencingHandler`.

Über die Methoden vom `IReferencingHandler` bzw. `IChatReferencing-Handler` können die Werkzeuge über das *Dependency-Injection*-Entwurfsmuster mit den entsprechenden zentralen Modellen und *Controller*-Instanzen versehen und in das Referenzierungssystem integriert werden.

4.7 Integrierte *Awareness*-Anzeige

4.7.1 Einleitung

Ein wichtiges Mittel, mit dem die Kollaborierenden bei der Koordination ihrer simultanen Aktivitäten in virtuellen Lernumgebungen unterstützt werden können, ist die Präsentation von *Awareness*-Informationen zum aktuellen Geschehen. So wird in Chatwerkzeugen üblicherweise angezeigt, wer gerade einen Beitrag erstellt (siehe Kap. 2.3.3), und in Konstruktionswerkzeugen wird z. B. kenntlich gemacht, welche Teilnehmer gerade welches Objekt manipulieren (siehe Kap. 2.4.2). Nach Anforderung **I6** bedarf es nun in dualen Kollaborationsräumen einer integrierten Präsentation der *Awareness*-Informationen. Wie dies erreicht werden kann, ist Gegenstand der folgenden Ausführungen. Zuerst wird das Konzept der integrierten *Awareness* genauer beleuchtet, bevor dann die notwendigen Funktio-

nalitäten separiert und das System zur Integration der *Awareness*-Informationen dargestellt wird. Abschließend wird auf Varianten der Realisierung eingegangen.

4.7.2 Konzeption integrierter *Awareness*

Ziel der integrierten *Awareness*-Anzeige ist es, die Informationen über die simultanen Aktivitäten der anderen Teilnehmer, welche sich über die beiden Interaktionsbereiche erstrecken können, leichter erfassbar zu machen. Hierbei sollen die von den beiden Werkzeugen bereitgestellten *Awareness*-Informationen visuell integriert werden. Dies impliziert, dass die *Awareness*-Informationen nicht nur im jeweiligen Interaktionsbereich an den Orten der Aktivitäten der anderen Teilnehmer, sondern z. B. an der Stelle der Aktivität des Nutzers präsentiert werden.

Welche *Awareness*-Informationen das Konstruktionswerkzeug anbietet, hängt dabei stark von der Art des Werkzeuges ab, Anhaltspunkte liefert das auf Seite 33 vorgestellte Konzept von Gutwin & Greenberg (2002). An dieser Stelle soll von den konkreten Informationsarten wie auch der konkreten Darstellung abstrahiert werden und die prinzipiellen Fragen der Integration im Vordergrund stehen.

4.7.3 Funktionale Dekomposition

Welche Funktionalitäten sind nun für die Integration der *Awareness*-Informationen notwendig? Erstens müssen die *Awareness*-Informationen aus den beiden Interaktionsbereichen in ein gemeinsames Datenmodell überführt werden. Hierzu müssen diese aus dem werkzeuginternen Format in das des gemeinsamen Datenmodells übersetzt werden.

Zweitens bedarf es Visualisierungskomponenten, welche die im gemeinsamen Datenmodell enthaltenen Informationen in den Werkzeugen geeignet darstellen. Erstellt ein Nutzer z. B. gerade einen Chatbeitrag, dann sollten die Informationen in der Nähe des Eingabefensters beispielsweise in textueller Form präsentiert werden.

4.7.4 Modellierung des *Awareness*-Integrationssystems

Auf Basis dieser Funktionalitäten lässt sich nun das System zur Integration der *Awareness*-Informationen modellieren (siehe Abb. 4.33). Beim Entwurf des Systems zur Integration der *Awareness*-Informationen wird davon ausgegangen, dass jedes Werkzeug bereits *Awareness*-Informationen bereitstellt. Diese werden durch den oder die werkzeuginternen `AwarenessController` in das ebenfalls werkzeuginterne `AwarenessModel` geschrieben. Dieses befindet sich auf Ebene des gemeinsamen Datenmodells, da die Informationen im Modell zwischen den Laufzeitumgebungen aller Teilnehmer im Kollaborationsraum konsistent gehalten werden müssen.

Um nun die Informationen aus den einzelnen `AwarenessModels` der Werkzeuge im bereichsübergreifenden `GlobalAwarenessModel` gebündelt verfügbar

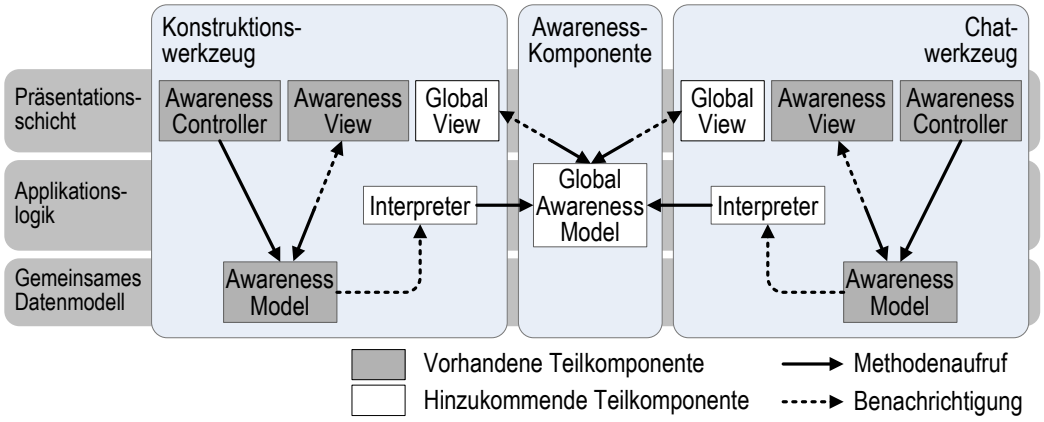


Abbildung 4.33: Schematische Darstellung des *Awareness*-Systems

zu machen, werden diese über das *Observer*-Entwurfsmuster von werkzeugspezifischen `Interpreters` beobachtet. Bei jeder Änderung im `AwarenessModel` werden diese benachrichtigt und aktualisieren daraufhin das `GlobalAwarenessModel`. Das `GlobalAwarenessModel` liefert damit eine integrierte Sicht auf die *Awareness*-Informationen. Änderungen im Modell werden an die registrierten werkzeugspezifischen `GlobalViews` propagiert und führen dort zu einer Aktualisierung der *Awareness*-Anzeige.

Zum Abschluss zeigt Abbildung 4.34 die für die Einbindung der Werkzeuge bzgl. des *Awareness*-Systems benötigten Schnittstellen: Im werkzeugseitigen `IToolModel` kann wiederum entsprechend des *Dependency-Injection*-Entwurfsmusters das von beiden Werkzeugen genutzte `GlobalAwarenessModel` gesetzt werden.

4.7.5 Integrationsvarianten

In der oben beschriebenen Variante wird das bereichsübergreifende `GlobalAwarenessModel` lokal auf Ebene der Applikationslogik gebildet und aktualisiert und stellt insofern nur eine integrierte Sicht der beiden unabhängigen (replizier-

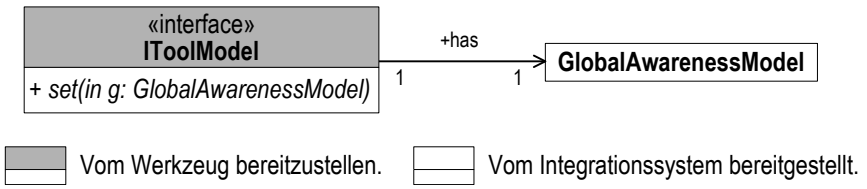


Abbildung 4.34: Klassendiagramm der für die Schnittstelle zwischen den Werkzeugen und dem *Awareness*-System relevanten Klassen

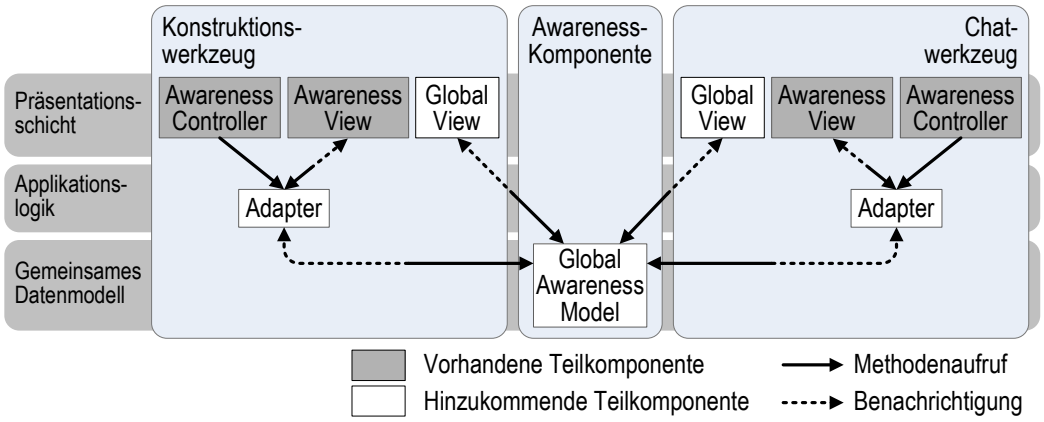


Abbildung 4.35: Vereinfachung des *Awareness*-Systems durch Verwendung eines gemeinsamen `GlobalAwarenessModels`

ten) `AwarenessModels` der Werkzeuge dar. Diese Variante bietet sich an, wenn die Werkzeuge unabhängige *Groupware*-Infrastrukturen nutzen.

Verwenden beide Werkzeuge jedoch dieselbe *Groupware*-Infrastruktur, so kann die Integration der *Awareness*-Informationen darüber erfolgen, dass beide Werkzeuge ein und dasselbe `GlobalAwarenessModel` anstelle ihrer separaten `AwarenessModels` verwenden (siehe Abb. 4.35).

Damit die Werkzeuge mit dem `GlobalAwarenessModel` interagieren können, wird entsprechend des *Adapter*-Entwurfsmusters in jedem Werkzeug ein spezifischer `Adapter` verwendet, der das *Interface* des jeweiligen `AwarenessModels` implementiert und die Aufrufe an das `GlobalAwarenessModel` delegiert.

4.8 Zusammenfassung

Die Kombination eines kollaborativen Konstruktionswerkzeugs und eines Chatwerkzeugs in einem dualen Interaktionsraum bringt eine Reihe von Barrieren mit sich. Zum einen wird die Interaktion zwischen den Gruppenmitgliedern, zum anderen die Reflexion über den Kollaborationsprozess erschwert. Abhilfe schafft eine verbesserte Integration der beiden Werkzeuge – der beiden Interaktionsbereiche, für die in Abschnitt 2.5 neun Anforderungen aufgestellt wurden. In diesem Kapitel wurde ein Integrationssystem konzipiert, welches die Anforderungen mit sechs verschiedenen Integrationsmaßnahmen adressiert:

Konsistente Aktionsabfolgen: In der synchronen computervermittelten Kommunikation enthält die Empfangsreihenfolge der kommunikativen Beiträge – die sowohl im Chat- wie auch im Konstruktionswerkzeug abgegeben werden können – wichtige Informationen für die Interpretation ihrer Inhalte. Werden durch das Medium Beiträge in ihrer Reihenfolge vertauscht, so wird

das *Grounding* erschwert. Entsprechend Anforderung **I 5** sollen deshalb alle Teilnehmer die Aktionen, die (u. U. simultan) in den beiden Interaktionsbereichen durch Gruppenmitglieder ausgeführt werden, in derselben Reihenfolge empfangen. Dies wird durch ein Nachrichtenaustauschmodell umgesetzt, welches die totale Ordnung aller Aktionen zusichert (siehe Kap. 4.2).

Kombinierte Interaktionshistorien: Für die Unterstützung der Reflexion bedarf es einer integrierten Interaktionshistorie. Der Kollaborationsverlauf setzt sich aus dem Diskurs und dem Konstruktionsverlauf zusammen. Die einzelnen Aktionen in beiden Interaktionsbereichen können aufeinander bezogen sein und eine Aktion in dem einem Bereich kann eine Reaktion in dem anderen nach sich ziehen. Für die Reflexion ist es notwendig, diese verschränkte Abfolge der Aktionen nachvollziehbar zu machen, insbesondere auch für ältere Teile des Kollaborationsprozesses, die vor dem Betreten des Raumes z. B. in vergangenen Sitzungen stattfanden (Anforderung **I 7**). Dies erfolgt durch das *History*-System (siehe Kap. 4.3).

Kontextrekonstruktion: Die Bereitstellung einer integrierten Historie allein reicht jedoch nicht aus. Vielmehr müssen die Nutzer dabei unterstützt werden, den über die beiden Interaktionsbereiche verteilten Interaktionsverlauf zu erkunden. In Anforderung **I 8** wird deshalb eine Funktionalität gefordert, mit der die Nutzer zu einem Chatbeitrag den zugehörigen Artefaktkontext ermitteln können. Analog fordert Anforderung **I 9** eine Funktionalität, um den in einem Artefaktzustand erfolgten Diskurs im Chat leicht identifizieren zu können. Beide werden durch das System zur Kontextrekonstruktion realisiert (siehe Kap. 4.4).

Aktivitätsdokumentation: Das textuelle Kommunikationsmedium Chat zeichnet sich dadurch aus, dass die Teilnehmer mit ihrer Aufmerksamkeit zwischen der Beitragsproduktion und der Rezeption der eintreffenden Chatbeiträge wechseln. Die textuelle Repräsentation des Diskurses ermöglicht es, die eingetroffenen Beiträge verzögert zu lesen. Manipulieren nun andere Teilnehmer im Konstruktionswerkzeug die Artefakte, so können diese Manipulationen Teil des Diskurses sein. Sie sollten deshalb analog zu den Chatbeiträgen im Chattranskript dokumentiert werden (Anforderung **I 4**). Dies wird durch das in Kapitel 4.5 konzipierte System zur bereichsübergreifenden Aktivitätsdokumentation umgesetzt.

Deiktische Bezugnahme: Befinden sich die Kollaborierenden im selben (realen) Raum, so nutzen sie vielfältige Gesten, um Objekte der gemeinsamen Umgebung zu identifizieren. Im virtuellen Kollaborationsraum, in dem die Kommunikation über Chat und Aktionen an den gemeinsamen Artefakten erfolgt, ist dies nicht möglich. Die Anforderungen **I 1**, **I 2** sowie **I 3** zielen darauf, das Medium Chat so zu erweitern, dass die Kommunizierenden leichter auf gemeinsame Objekte Bezug nehmen können. Umgesetzt wird dies

durch die explizite Referenzierung (siehe Kap. 4.6), die es den Teilnehmern ermöglicht, ihre Chatbeiträge mit anderen Chatbeiträgen und Objekten oder Ausschnitten der gemeinsamen Arbeitsumgebung zu verknüpfen.

Integrierte *Awareness*-Anzeige: Die Produktion von Beiträgen im Chat ist durch die anderen Teilnehmer nicht zu beobachten. Dasselbe gilt im Prinzip auch für die Aktionen im Konstruktionswerkzeug: Erst die Resultate elementarer Operationen sind durch die anderen zu beobachten, das eigentliche Agieren jedoch nicht. Damit die Kollaborierenden sich jedoch bei ihrem simultanen Tun koordinieren können, zeigen Konstruktions- und Chatwerkzeuge *Awareness*-Informationen über das momentane Tun der Kollaborationspartner an (Minimalanforderungen **K 2** und **C 4**). Diese müssen jedoch visuell integriert präsentiert werden (Anforderung **I 6**). Das in Kapitel 4.7 vorgestellte *Awareness*-Integrationssystem erlaubt den bereichsübergreifenden Zugriff auf die *Awareness*-Informationen beider Werkzeuge und damit eine Präsentation, die die Informationen aus beiden Bereichen integriert.

Für jede der Integrationsmaßnahmen ist eine separate Integrationskomponente zuständig, die mit den Werkzeugen verknüpft werden muss. In Abbildung 4.36 sind die hierzu notwendigen Schnittstellen noch einmal zusammengefasst und entsprechend der sechs Funktionsbereiche gruppiert.

Die Konzeption der sechs Integrationskomponenten zielte jeweils auf einen der sechs Funktionsbereiche. Die von den einzelnen Komponenten bereitgestellten Funktionalitäten können jedoch auch durch andere Integrationskomponenten genutzt werden:

- Die sich aus der totalen Ordnung ergebende globale Indizierung aller Aktionen wird zur Herstellung der kombinierten Interaktionshistorie, für die Kontextrekonstruktion wie auch für die bereichsübergreifende Aktivitätsdokumentation genutzt.

- Bei der Kontextrekonstruktion kann die Anforderung eines Kontextes das Nachladen der den Kontext enthaltenden Historienteile auslösen.

- Desgleichen kann die Anzeige einer Referenz, welche auf noch nicht aus der Historie geladene Chatsequenzen oder Artefaktversionen verweist, ein Nachladen der das Referenzziel enthaltenden Historienteile auslösen.

- Umgekehrt erlaubt das Nachladen von Historienteilen die Auflösung bisher unbekannter Referenzziele.

- Zur Darstellung der Aktionsmarkierungen im Chat kann die Referenzierung genutzt werden. Dies erlaubt die Hervorhebung der durch eine Aktion betroffenen Artefaktbestandteile und ein leichtes Navigieren zur entsprechenden Artefaktversion.

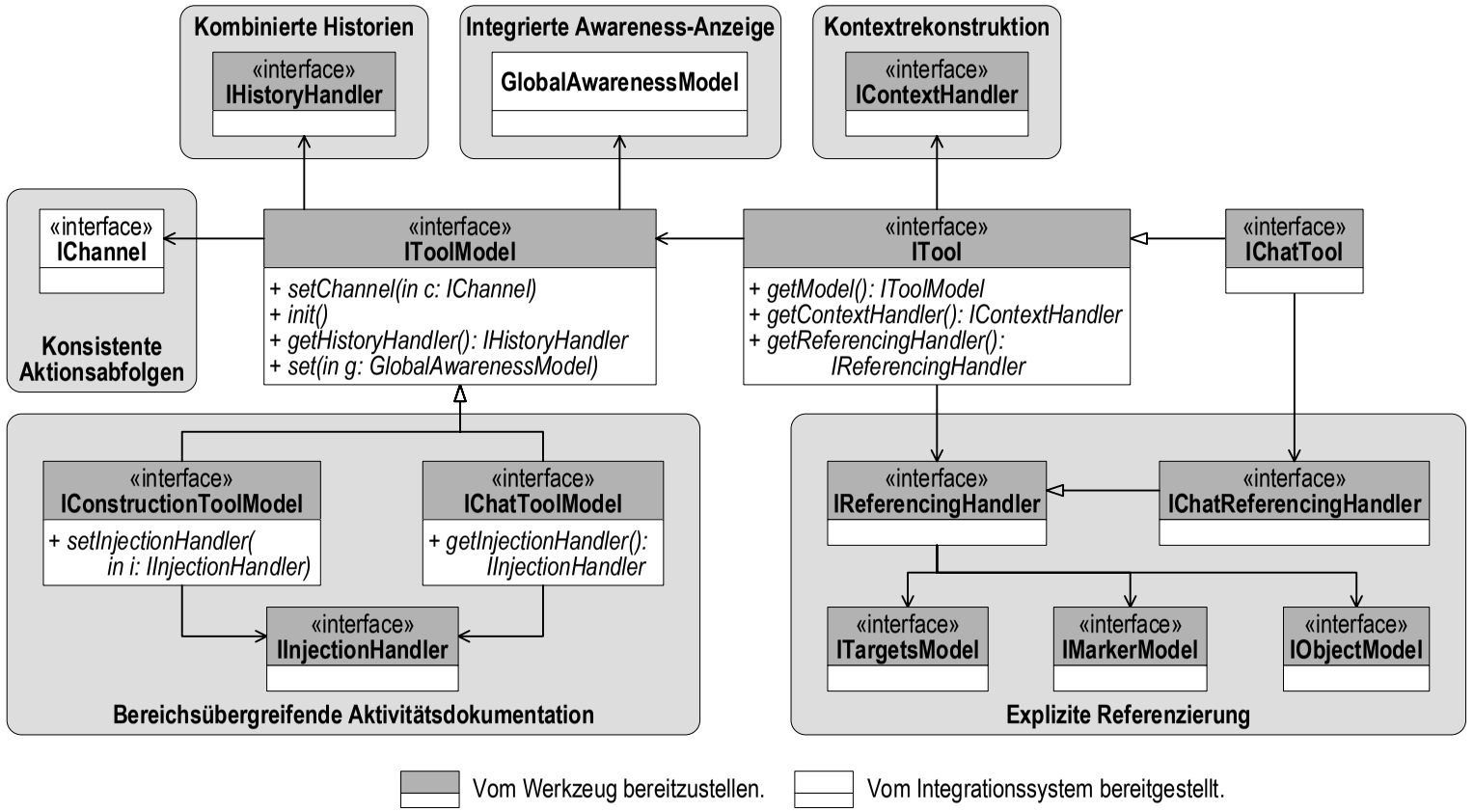


Abbildung 4.36: Übersicht über alle für die Schnittstelle zwischen den Werkzeugen und dem Integrationssystem relevanten Klassen – gruppiert nach den sechs Funktionsbereichen

Im nächsten Kapitel wird eine exemplarische Implementierung des Integrationssystems vorgestellt. Daran anschließend wird in Kapitel 6 das Konzept evaluiert.

5 *ConcertChat*: eine Beispielimplementierung

Die in Kapitel 4 beschriebenen Integrationsmaßnahmen sind exemplarisch in dem Kollaborationssystem *ConcertChat* umgesetzt. *ConcertChat* entstand am Fraunhofer IPSI in Darmstadt über einen Zeitraum von sechs Jahren[1] und profitierte von dem Einsatz in einer Reihe von Forschungsprojekten mit unterschiedlichen Anwendungskontexten. Erste Funktionalitäten zur expliziten Referenzierung wurden für die Experimentalsoftware *ExpChat* zur Untersuchung der Wirksamkeit von Lernskripten entwickelt (z. B. Pfister & Mühlpfordt, 2002). Die Referenzierungsfunktionalität wurde anschließend als eigenständige Bibliothek extrahiert, um sie in weiteren Applikationen wiederzuverwenden, wie z. B. in der Kollaborationsanwendung *RolePlay* für das als Rollenspiel organisierte Sprachenlernen in Kleingruppen (Balzer & Mühlpfordt, 2004; Jödick, 2004) oder dem im VMT-Projekt verwendeten *WhiteboardChat* (z. B. Stahl et al., 2006b), der einen Chat mit einem *Shared Whiteboard* verbindet.

In der ersten Version des *WhiteboardChats* waren die beiden Interaktionsbereiche noch unverbunden, es war ein Beispiel eines nicht-integrierten Interaktionsraumes. Beim Einsatz wurden die Defizite jedoch schnell deutlich und es begann eine kontinuierliche Weiterentwicklung hin zu den in dieser Arbeit vorgestellten Integrationsmaßnahmen.

Dieses Kapitel gibt einen Einblick, wie die Integrationsmaßnahmen in *ConcertChat* umgesetzt sind.

[1] Der Quelltext von *ConcertChat* ist unter `http://sourceforge.net/projects/concertchat/` frei verfügbar.

5.1 Vorbemerkung

Die in dieser Arbeit beschriebenen Integrationsmaßnahmen sind in dem Kollaborationssystem *ConcertChat* umgesetzt. *ConcertChat* zielt auf die Bereitstellung aufgabenspezifischer Kollaborationsräume. Die in *ConcertChat* entwickelten Kollaborationsanwendungen kombinieren hierfür einen Chat mit verschiedenen Arten von (Lern-) Materialien bzw. Konstruktionswerkzeugen.

Im *WhiteboardChat* sind alle hier beschriebenen Integrationsmaßnahmen umgesetzt. Abbildung 5.1 zeigt die Grundelemente der Benutzungsschnittstelle: Links befindet sich das *Shared Whiteboard* mit der gemeinsamen Arbeitsfläche, rechts oben die Liste der momentan im Raum anwesenden Personen und darunter das Chatwerkzeug mit der Transkriptanzeige und dem Eingabefeld für eigene Chatbeiträge.

ConcertChat stellt den Entwicklern eine Reihe von Grundfunktionalitäten für kollaborative Anwendungen bereit. Diese umfassen neben den beschriebenen Integrationsmaßnahmen das Nutzermanagement mit Authentifizierung, Mechanismen zur Erzeugung und Verwaltung von Kollaborationsräumen, Werkzeuge zum Export von Sitzungstranskripten und einen *Session Player*, mit dem vollständige Kollaborationshistorien im Nachhinein abgespielt werden können. Für die leichte Entwicklung neuer aufgaben- und lernmaterialspezifischer Kollaborationswerkzeuge bietet *ConcertChat* einen Standard-Container für Anwendungen, in denen allein die materialspezifische linke Seite der Benutzungsschnittstelle (im *WhiteboardChat* ist dies das *Shared Whiteboard*) gefüllt werden muss.

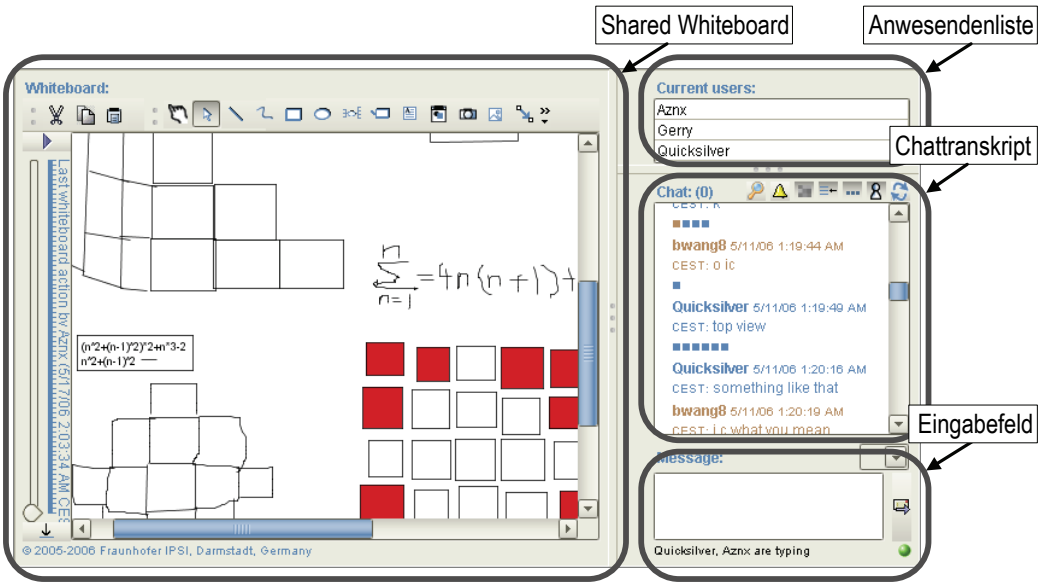


Abbildung 5.1: Benutzungsschnittstelle der *WhiteboardChat*-Anwendung

ConcertChat ist in Java implementiert und basiert auf einer Client-Server-Architektur. Anfangs kam das *Groupware-Framework* DyCE (Tietze, 2001) zum Einsatz; die Werkzeuge waren eng mit diesem *Framework* verknüpft. Um die Kollaborationsanwendungen jedoch auch mit anderen Infrastrukturen einsetzen zu können, wurde als minimale Schnittstelle das *Channel*-Konzept entwickelt. Dies reduziert die Anforderung an die *Groupware*-Infrastruktur darauf, Nachrichtenkanäle bereitzustellen. Hiermit wurde es möglich, die Anwendungen ohne jede Änderung der Applikationslogik auf andere Infrastrukturen, wie z. B. Sametime (Attardo et al., 2007) oder auch Agilo (Guicking et al., 2005), zu portieren.

Die Umsetzung der Integrationsmaßnahmen in *ConcertChat* entspricht den in Kapitel 4 beschriebenen Modellierungen. Im Folgenden werden wesentliche Aspekte der Implementierung vorgestellt. Kapitel 5.2 gibt einen Überblick über die Gesamtarchitektur. Anschließend wird genauer auf die Umsetzungen der Integrationsmaßnahmen eingegangen.

5.2 Architekturüberblick

In *ConcertChat* stehen alle Grundfunktionalitäten für aufgabenspezifische Kollaborationsräume bereit. Diese Grundfunktionalitäten sind clientseitig unabhängig von einem konkreten *Groupware-Framework* implementiert. Eine Schnittstelle (`IClientConnection` in Abb. 5.2) beschreibt die vom *Groupware-Framework* bereitzustellenden Funktionen. Unterschiedliche Implementierungen dieser Schnittstelle erlauben es, die *ConcertChat*-Clients[2] ohne jede Änderung mit unterschiedlichen *Groupware-Frameworks* zu betreiben. Seitens der Serverfunktionalität ist die Abgrenzung von den konkreten *Frameworks* schwieriger, da sie Groupwarefunktionalitäten wie Nutzerverwaltung, Persistenz und *Messaging* im unterschiedlichen Maße und mit unterschiedlichen Konzepten bereitstellen.

Standardmäßig kommt das erwähnte Agilo[3] als *Messaging*-Infrastruktur zum Einsatz. In Agilo erfolgt jeglicher Austausch zwischen Client und Server über (typisierte) Nachrichten. Die Nachrichten werden in Abhängigkeit von ihrem Typ server- und clientseitig von verschiedenen Modulen verschickt und empfangen, wobei jedoch die totale Ordnung der Nachrichten zugesichert wird.

Abbildung 5.2 zeigt die Basiselemente des *ConcertChat*-Systems. Diese sind zuständig für:

Anmeldung/Nutzerverwaltung: Bevor ein Nutzer einen Kollaborationsraum betreten kann, muss sein Client im System angemeldet und gegebenenfalls der Nutzer authentifiziert werden. Clientseitig ist hierfür der `LoginHandler` zuständig, der mit dem serverseitigen `UserManager` kommuniziert.

[2] Ein *ConcertChat*-Client zum Betreten eines Raumes wird üblicherweise über Java-Webstart (siehe `http://java.sun.com/products/javawebstart/`) bereitgestellt und gestartet.

[3] Siehe `http://www.agilo.org`.

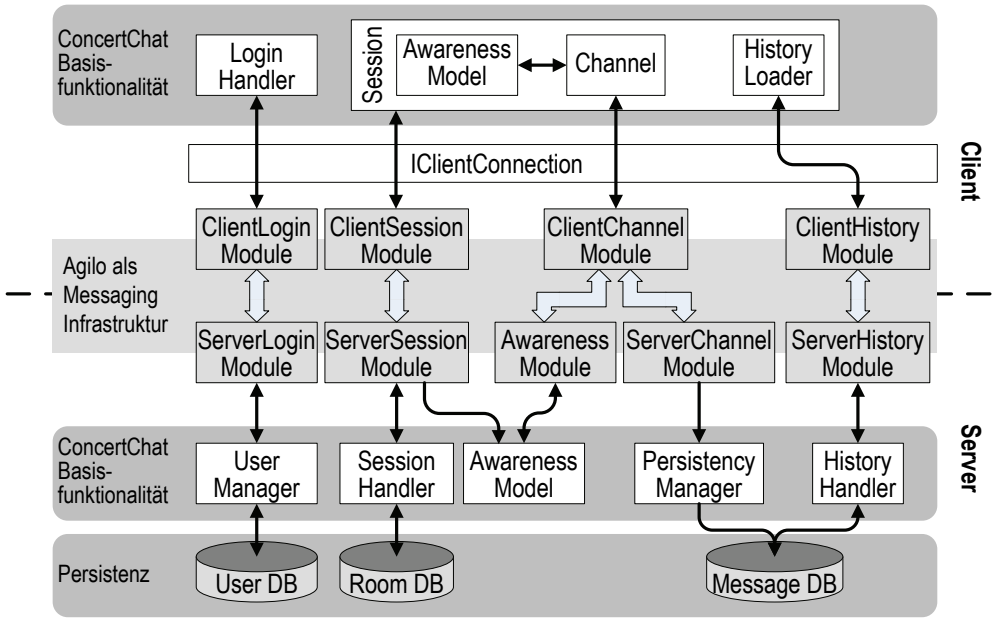


Abbildung 5.2: Basiselemente der *ConcertChat*-Systemarchitektur

Betreten/Verlassen eines Raumes: Beim Betreten eines Raumes muss im Client des Nutzers das gemeinsame Basis-Datenmodell initialisiert werden. Hierfür wird ein `Session`-Objekt erzeugt, welches einen `Channel` (siehe Kap. 4.2.3), ein `AwarenessModel` (siehe Kap. 4.7) und einen `HistoryLoader` (siehe Kap. 4.3.5) für diesen Raum umfasst.

Serverseitig werden die Räume vom `SessionHandler` verwaltet. Betritt ein Nutzer einen Raum, so muss dies auch den anderen Nutzern mitgeteilt werden, wofür einerseits eine spezielle Systemnachricht über den rauminternen `Channel` verschickt und andererseits das `AwarenessModel` des Raumes modifiziert wird.

Im *ConcertChat*-System können Räume bei der Erzeugung konfiguriert werden, z. B. mit einer Liste von Arbeitsmaterialien oder Instruktionen. Diese Konfigurationen werden zusammen mit einem Identifikator für den raumspezifischen Kanal in der Raum-Datenbank abgelegt.[4]

Nachrichtenaustausch im Raum: Der Nachrichtenkanal für einen Raum wird vom `Client`- zusammen mit dem `ServerChannelModule` realisiert. Auf *ConcertChat*-Ebene wird jedes über den `Channel` verschickte Objekt (z. B. ein Chatbeitrag oder eine Aktion im gemeinsamen Arbeitsbereich) zusammen mit dessen Typ und einer clientseitig erzeugten ID in ein Transferobjekt eingepackt (dem sogenannten `MessageContainer`, siehe Abb. 5.3). Serverseitig werden diese Transferobjekte um den für die Kontextrekonstruktion

[4] Die Konfiguration wird in der Webstart-Konfiguration (siehe Fußnote 2) als Parameter übergeben.

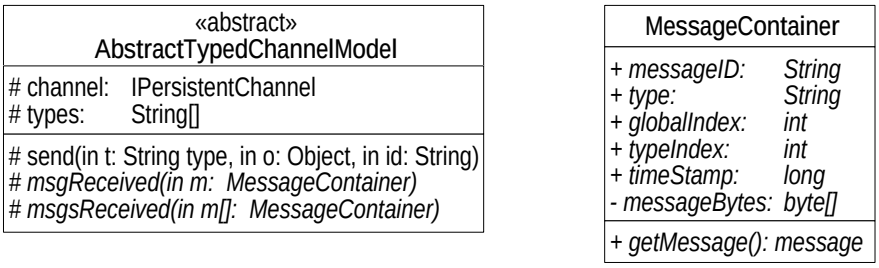


Abbildung 5.3: Diagramm der Basisklassen für die Implementierung gemeinsamer Datenmodelle

notwendigen globalen Index angereichert, in der *Message*-Datenbank gespeichert und an alle im Raum befindlichen Clients zugestellt.

Bereitstellen der Historie: Der im `Session`-Objekt verfügbare `HistoryLoader` erlaubt das selektive Nachladen der Nachrichtenhistorie in Form von Transferobjektlisten.

Bereitstellen der *Awareness*-Informationen: Im *ConcertChat*-System kommt ein globales Modell für die *Awareness*-Informationen zum Einsatz (siehe Abb. 4.35 auf S. 113). Jede Änderung im *Awareness*-Modell wird als eine entsprechende Operation über den Nachrichtenkanal des Raumes verschickt. Dies hat den Vorteil, dass diese Änderungen in der Nachrichtenhistorie des Raumes gespeichert und später z. B. für die Analyse genutzt werden können. Für die effiziente Initialisierung einer Session steht serverseitig immer der aktuelle Modellzustand zur Verfügung. Hierzu werden die Transferobjekte für die Änderungen im *Awareness*-Modell vom Server ausgepackt und ausgewertet. Betritt ein Client einen Raum, so wird diesem der aktuelle Modellzustand übermittelt.

Als Bindeglied zwischen dem *ConcertChat* zugrunde liegenden Konzept von Nachrichtenkanälen und der gemeinsamen Datenmodelle für Werkzeuge, die üblicherweise als über Nutzeraktionen veränderliche Objekte aufgefasst werden, steht den Anwendungsentwicklern die abstrakte Klasse `AbstractTypedChannelModel` zur Verfügung (siehe Abb. 5.3). Mit Hilfe dieser Klasse lassen sich Datenmodelle wie z. B. für das *Whiteboard* erstellen, die beim Betreten des Raumes mittels der vollständigen Aktionshistorie den aktuellen Modellzustand rekonstruieren. Das `AbstractTypedChannelModel` kapselt hierzu die dafür notwendige Initialisierung. Bei dieser muss sichergestellt werden, dass die in der Historie gespeicherten Aktionen und die während der Initialisierungsphase eventuell simultan von anderen Teilnehmern im Raum getätigten Veränderungen in eine vollständige und konsistente Reihenfolge gebracht werden. Die Initialisierung wird abgeschlossen durch die modellspezifische Verarbeitung aller bisheri-

gen Aktionen.[5] Hierzu muss die abstrakte Methode `msgsReceived()` implementiert werden. Diese erhält als Parameter die Liste aller bisherigen entsprechend der Aktionsabfolge geordneten `MessageContainer`.

Ein `MessageContainer` ist das in *ConcertChat* genutzte Transferobjekt zum Versenden beliebiger Objekte. Hierbei wird der Java-Mechanismus der Serialisierung zweifach genutzt. Erstens wird clientseitig das zu versendende Objekt in Form eines serialisierten Byte-Arrays im `MessageContainer` gespeichert. Zweitens wird der `MessageContainer` selbst zum Verschicken serialisiert. Der Vorteil der doppelten Serialisierung liegt darin, dass serverseitig die eigentlichen Datenobjekte nicht zeitaufwendig deserialisiert werden müssen, sondern dort im Allgemeinen nur mit den Byte-Arrays gearbeitet wird.

Nach erfolgter Initialisierung wird beim Eintreffen neuer Nutzeraktionen (wiederum transportiert in einem `MessageContainer`) die zu implementierende Methode `msgReceived()` aufgerufen. Verschickt werden Aktionen über die verfügbare `send()`-Methode.

Aufbauend auf diesen Basiselementen sind die Integrationsmaßnahmen in *ConcertChat* umgesetzt. Auf diese soll nun detailliert eingegangen werden.

5.3 Konsistente Aktionsabfolgen

Wie beschrieben, werden in *ConcertChat* alle Beiträge im Chat und alle Aktionen im gemeinsamen Arbeitsbereich als eigenständige Nachrichten eingepackt in ein Transferobjekt an den Server verschickt. Die Konsistenz der Aktionsabfolgen im Sinne einer totalen Ordnung ist entsprechend der Ausführungen in Kapitel 4.2.3 eine Anforderung an den zugrunde liegenden Nachrichtenkanal. In *ConcertChat* wird diese Anforderung an die genutzte *Messaging*-Infrastruktur delegiert. Mit Agilo steht eine Infrastruktur bereit, welche dieser Anforderung genügt.

5.4 Integrierte Historien

Die Bereitstellung integrierter Historien erfolgt in *ConcertChat* durch die in Abbildung 4.16.b (siehe S. 87) dargestellte Variante: Der Chat und der gemeinsame Arbeitsbereich nutzen dieselbe *Messaging*-Infrastruktur und ein gemeinsames *Persistency-Backend*. Im Server wird in jedes über den Kanal verschickte Transferobjekt (in welches die Chatbeiträge und Aktionen eingepackt sind) vor dem Abspeichern und Weitersenden der für die Kontextrekonstruktion benötigte pro Kanal fortlaufende globale Index eingetragen.

[5] Eine denkbare Erweiterung liegt in der Abspeicherung vollständiger Zwischenzustände, wie es auch in Lukosch (2003) vorgeschlagen und für das `GlobalAwarenessModel` umgesetzt wird. Dies hätte den Vorteil, dass die Zeit zum Herstellen des aktuellen Zustands nach dem Betreten eines Raumes insbesondere für fortgesetzte längere Sitzungen verringert werden kann.

Wie in Kapitel 4.3.5 auf Seite 86 ausgeführt, müssen verschiedene Varianten zum Laden der Historie unterstützt werden. Für die bisher im *ConcertChat*-System entwickelten Anwendungen kommen zum Einsatz:

Vollständiges Laden: Betritt ein Teilnehmer einen Raum, so wird z. B. für das *Whiteboard* zunächst die vollständige Historie aller Aktionen geladen. Mit dieser wird der aktuelle Zustand über das Ausführen aller bisherigen Aktionen berechnet.

Sukzessives Laden von n Nachrichten: Im Gegensatz dazu werden vom Chattranskript initial nur die letzten 100 Chatbeiträge geladen. Durch Klick auf den mit ↻ gekennzeichneten Button (siehe Abb. 5.1) können die Nutzer jeweils weitere 100 Beiträge nachladen.

Sukzessives Laden bis zu einer bestimmten Nachricht: Soll eine Referenz auf einen Chatbeitrag angezeigt werden, der bisher noch nicht geladen wurde (und demzufolge weiter zurück in der Chathistorie liegt), so werden alle Beiträge angefordert, die zwischen dem ältesten lokal bekannten und dem in der Referenz kodierten liegen.

Laden einzelner Nachrichten: Bei bestimmten Nutzeraktionen – wie z. B. dem Einfügen eines Bildes in die gemeinsame Arbeitsfläche – können die für die Aktion benötigten Daten recht umfangreich sein. Hier ist es sinnvoll, den Aktionsinhalt (z. B. die Bilddaten) von der Aktionsbeschreibung (z. B. Akteur, Bildmaße und Bildposition) zu trennen und als separate Nachricht mit einem eigenen Typ zu verschicken. Hierzu wird in der Aktionsbeschreibung die ID der den Aktionsinhalt tragenden Nachricht eingetragen. In späteren Sitzungen kann dann der Aktionsinhalt bei Bedarf asynchron geladen werden, indem mit der kodierten ID selektiv die zugehörige Nachricht abgerufen wird.

5.5 Kontextrekonstruktion

Die Kontextrekonstruktion folgt der in Kapitel 4.4 vorgestellten Modellierung. Sie basiert auf den serverseitig in die Transferobjekte eingetragenen globalen Indizes der einzelnen Nachrichten.

In Abbildung 5.4 ist dargestellt, wie ein Kontext in der Benutzungsschnittstelle abgerufen werden kann:

Rekonstruktion des Diskurskontextes zu einem Artefaktzustand: Wie auf dem Screenshot im Vordergrund zu sehen ist, erlaubt im *Shared Whiteboard* ein *History-Slider*, jeden beliebigen alten Zustand des gemeinsamen Arbeitsbereichs wiederherzustellen. Mit der Kontextrekonstruktion kann zum angezeigten Artefaktzustand der dazugehörige Diskurs ermittelt werden. Hierzu

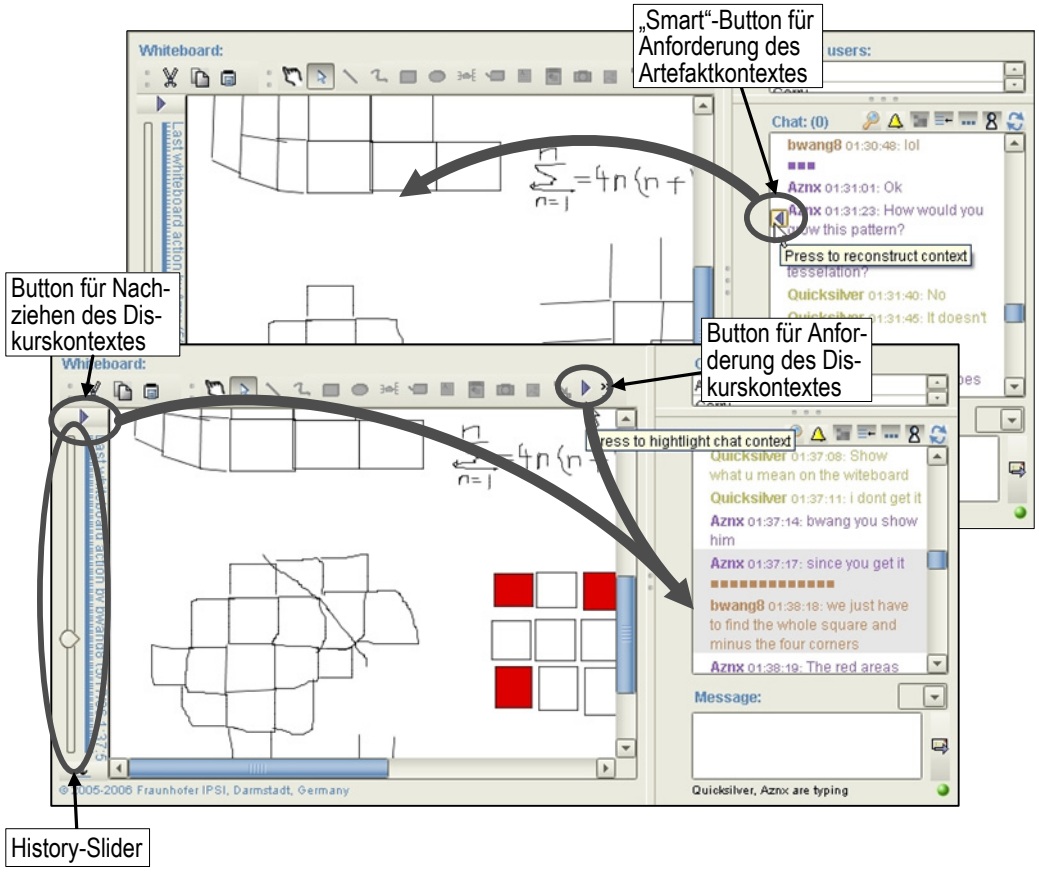


Abbildung 5.4: Kontextrekonstruktion in *ConcertChat*

werden die beiden Chatbeiträge, die direkt vor und nach der zu dem angezeigten Artefaktzustand führenden Aktion eintrafen, in den sichtbaren Bereich gebracht und kurzzeitig farblich hinterlegt. Angefordert werden kann dies durch Druck auf den markierten Button in der Tool-Palette. Eine weitere Möglichkeit bietet der Button oberhalb des *History-Sliders*. Mit diesem kann ein automatisches „Nachziehen" des Chatdiskurses aktiviert werden. Dabei wird beim Navigieren durch den Konstruktionsverlauf mittels des *History-Sliders* automatisch der zugehörige Chatdiskurs sichtbar gemacht und hervorgehoben.

Rekonstruktion des Artefaktkontextes zu einem Chatbeitrag: Umgekehrt erlaubt die Bildlaufleiste für das Chattranskript, zu älteren Chatbeiträgen zu springen. Wird der Cursor in den Bereich links neben einen Beitrag geführt, so erscheint (wie im hinteren Screenshot zu sehen) kurzzeitig ein Button. Beim Druck auf diesen Button wird im gemeinsamen Arbeitsbereich diejenige Version wiederhergestellt, welche beim Empfang des Beitrags zu sehen war.

Weitere Möglichkeiten zur Kontextrekonstruktion werden durch die Realisierungen der bereichsübergreifenden Aktivitätsdokumentation sowie der expliziten Referenzierung bereitgestellt. Auf diese gehen die folgenden Kapitel ein.

5.6 Bereichsübergreifende Aktivitätsdokumentation

Die bereichsübergreifende Aktivitätsdokumentation zielt darauf, die Aktionen im gemeinsamen Arbeitsbereich innerhalb des Chattranskripts darzustellen. Hierzu wird in *ConcertChat* (den Ausführungen in Kapitel 4.5 folgend) zu jeder Aktion im gemeinsamen Arbeitsbereich jeweils lokal eine spezielle Chatnachricht erzeugt und im `ChatModel` in die Liste aller Chatbeiträge an der durch den globalen Index der Aktion definierten Stelle eingefügt.

Abbildung 5.5 zeigt die Umsetzung der Aktivitätsdokumentation in der Benutzungsschnittstelle: Jede Aktion wird als ein Rechteck in der je Akteur spezifischen Farbe (die auch für die Anzeige der Chatbeiträge verwendet wird) dargestellt. Wie im Vordergrund der Abbildung zu sehen ist, werden zu einer Aktionsmarkierung weitere Informationen angezeigt, sobald der Mauszeiger über der Markierung positioniert ist. Diese umfassen den Akteur (in der Abbildung ist dies „bwang8“) und eine textuelle Beschreibung der Aktion („created a rectangle“).

Mit einem Mausklick auf die Aktionsmarkierung wird diejenige Artefaktversion im gemeinsamen Arbeitsbereich rekonstruiert, die mit der Aktion erzeugt wurde. Des Weiteren zeigt ein Pfeil kurzzeitig auf die in der Aktion manipulierten Ob-

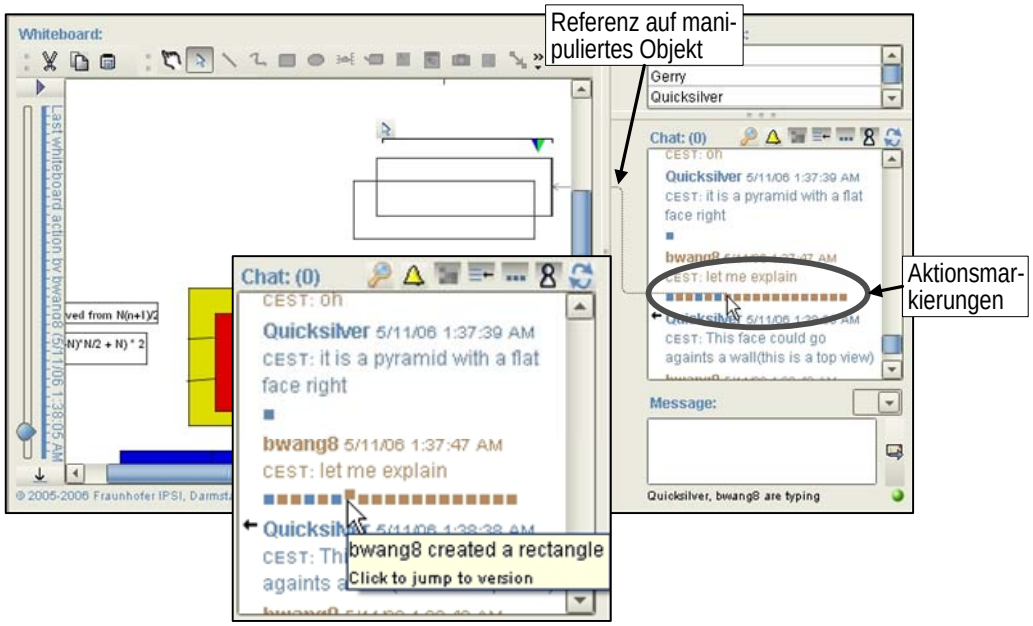


Abbildung 5.5: Darstellung der Aktionsmarkierungen im Chat

jekte. Dies stellt eine weitere Möglichkeit dar, den Artefaktkontext zum Chatdiskurs zu identifizieren, insbesondere für Kollaborationsphasen, in denen simultan im Chat diskutiert und im gemeinsamen Arbeitsbereich konstruiert wurde (wie es beispielsweise in der Abbildung zu sehen ist).

5.7 Explizite Referenzen

Explizite Referenzen sollen die deiktische Bezugnahme auf einen oder mehrere Chatbeiträge oder Teile von diesen sowie auf Objekte oder Ausschnitte des gemeinsamen Arbeitsbereichs erleichtern. *ConcertChat* setzt hierzu die in Kapitel 4.6 dargelegten Maßnahmen zur Referenzerzeugung, -übertragung und -anzeige um.

Hierfür stellt es den Anwendungsentwicklern einen Standard-Container für Kollaborationsanwendungen bereit. Dieser umfasst nicht nur das Chatwerkzeug und die Anwesendenliste, sondern nutzt Mechanismen des Java-Swing-*Frameworks*[6] zur Verwaltung und Darstellung der Referenzen. Wird dem Standard-Container eine Komponente (z. B. eine spezielle Materialanzeige oder ein Konstruktionswerkzeug wie das *Shared Whiteboard*) hinzugefügt, so wird automatisch geprüft, ob die Komponente oder ihre Subkomponenten referenzierbare und referenzierende Objekte anzeigen können (bzw. entsprechende Schnittstellen implementieren). In diesem Falle werden die referenzierbaren bzw. referenzierenden Komponenten im Referenzierungssystem registriert, womit sie an der Referenzauflösung und -darstellung teilnehmen.[7] Die Übertragung der Referenzen erfolgt zusammen mit den referenzierenden Chatbeiträgen.

Abbildung 5.6 zeigt die Darstellung expliziter Referenzen in der Benutzungsschnittstelle von *ConcertChat*. Jede referenzierende Nachricht wird links mit einem Indikator versehen, welcher anzeigt, ob die Nachricht auf den Arbeitsbereich (↞), auf andere Chatnachrichten (↥) oder beides (↨) verweist.

Trifft ein referenzierender Chatbeitrag ein oder wird er selektiert, so werden die ausgehenden Referenzen als Linien hin zu den referenzierten Objekten dargestellt. Eine Schwierigkeit ergibt sich bei der Anzeige von Referenzen eintreffender Nachrichten, die auf den gemeinsamen Arbeitsbereich eines *Shared Whiteboards* verweisen. Da der Arbeitsbereich naturgemäß seinen Inhalt ändert, müsste die Referenzanzeige korrekterweise jeweils den u. U. zeitlich zurückliegenden referenzierten Zustand des Arbeitsbereichs rekonstruieren. Dies birgt jedoch die Gefahr, dass die Teilnehmer in ihrer aktuellen Tätigkeit unterbrochen werden (wenn zum Beispiel im Arbeitsbereich gemalt wird). In *ConcertChat* wird deshalb der in Abbildung 5.6 als Vergrößerung gezeigte Ansatz verwendet. Zu einem Referenzziel im gemeinsamen Arbeitsbereich wird eine Zeitskala angezeigt, an der durch zwei Markierungen die referenzierte und die momentan dargestellte Version zu

[6] Siehe `http://java.sun.com/docs/books/tutorial/uiswing/`.

[7] Die Programmbibliothek für die Referenzierung umfasst referenzierbare Anzeigen für übliche Dokumentformate (Text, HTML, Bilder) sowie Basisklassen für die leichte Entwicklung neuer Komponenten.

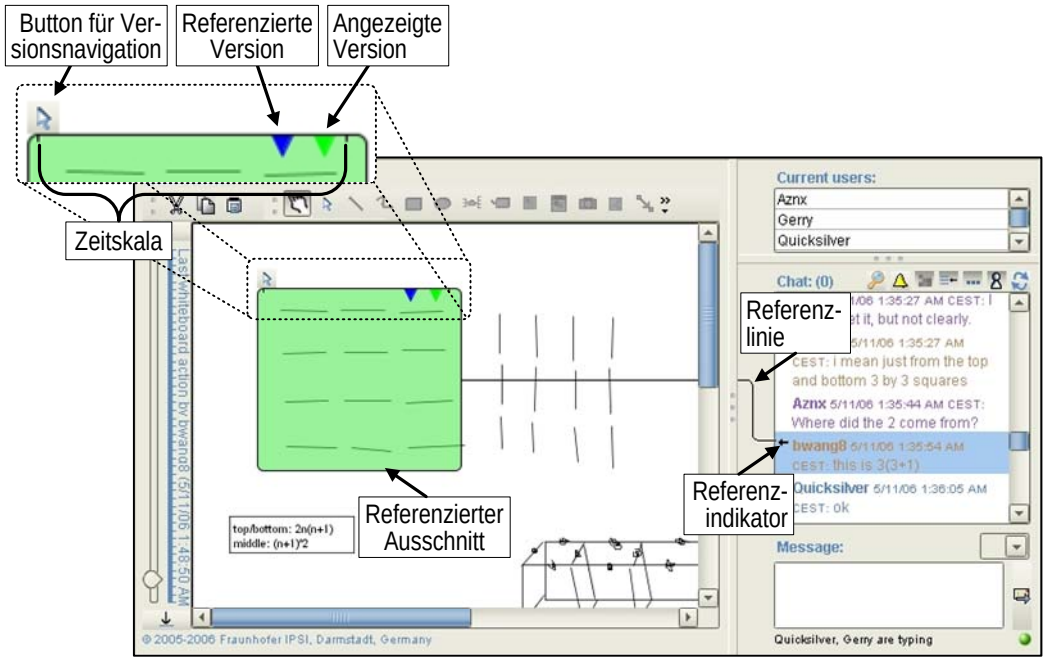


Abbildung 5.6: Darstellung expliziter Referenzen

erkennen sind. Per Knopfdruck kann in der Anzeige zwischen diesen beiden Versionen gewechselt werden.[8]

Die Mechanismen zur Referenzdarstellung kommen auch bei der bereichsübergreifenden Aktivitätsdokumentation durch Aktionsmarkierungen zum Einsatz (siehe Abb. 5.5). Die für die Aktionsmarkierung erzeugten „Aktions"beiträge enthalten hierfür Referenzen auf die in der Aktion manipulierten Objekte und die entsprechende Version.

5.8 Integration flüchtiger *Awareness*-Informationen

Die Integration flüchtiger *Awareness*-Informationen zielt darauf, die Informationen über aktuelle Aktivitäten aller Teilnehmer im Kollaborationsraum kombiniert am Ort der eigenen Aktivität zu ermöglichen. Entsprechend der Modellierung in Kapitel 4.7 kommt in *ConcertChat* ein von allen Interaktionsbereichen gemeinsam genutztes `GlobalAwarenessModel` zum Einsatz.

Im *WhiteboardChat* werden *Awareness*-Informationen darüber bereitgestellt, ob

- ein Teilnehmer gerade einen Beitrag erstellt (Tipp-*Awareness*),
- ein Teilnehmer gerade eine Textbox im *Shared Whiteboard* editiert.

[8] Bei Referenzen auf Objekte im Arbeitsbereich gibt es noch einen weiteren Fall: Die momentan angezeigte Version enthält diese Objekt nicht, weil sie entweder noch nicht oder nicht mehr existieren. Hier wird die zukünftige bzw. letzte Version der Objekte durchscheinend dargestellt.

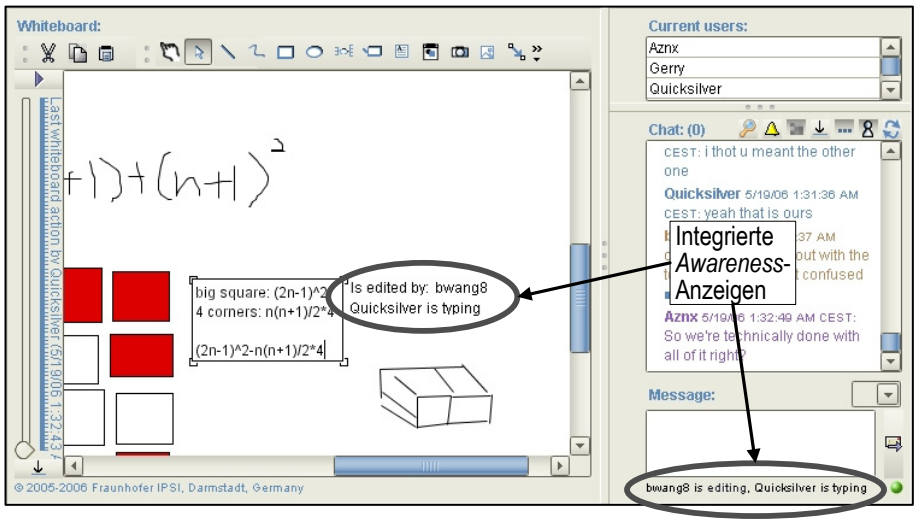


Abbildung 5.7: Integrierte Darstellung der *Awareness*-Informationen

Jede Änderung am `GlobalAwarenessModel` wird als eine Nachricht über den Kanal des Kollaborationsraumes übermittelt. Diese Nachricht wird somit von *ConcertChat* in ein Transferobjekt eingepackt und auch in der Historie persistent gespeichert. Dies ermöglicht es, die *Awareness*-Informationen im Nachhinein wieder vollständig zu rekonstruieren, wie es z. B. im erwähnten *Session Player* geschieht.

Wie eingangs in Kapitel 5.2 erwähnt, wird in *ConcertChat* serverseitig zu jedem Kollaborationsraum der aktuelle Zustand des `GlobalAwarenessModels` gehalten. Die Gründe hierfür sind, dass 1. das Datenmodell im Laufe einer Kollaboration sehr häufig verändert wird und 2. es jedoch im Allgemeinen nur wenige Einträge (maximal einen pro Anwesenden) enthält. Würde der aktuelle Zustand clientseitig aus der vollständigen *Awareness*-Historie rekonstruiert, müssten im Extremfall sehr viele Operationen geladen und ausgeführt werden, um den leeren Modellzustand („Niemand ist aktiv.“) herzustellen. Deshalb wird in *ConcertChat* jede Nachricht mit dem Typ der *Awareness*-Operationen vom Server entpackt und die entsprechende Operation auf der serverseitigen Instanz des Modells ausgeführt. Beim Betreten des Kollaborationsraums wird dem Client die aktuelle Version des *Awareness*-Modells zugestellt.

Die integrierte Anzeige ist in Abbildung 5.7 aus Sicht von „bwang8“, der gerade die Textbox in der Mitte des gemeinsamen Arbeitsbereiches editiert, dargestellt.[9] Hier ist neben bwang8 auch Quicksilver aktiv, der gerade einen Chatbeitrag erstellt. Die Informationen darüber werden sowohl neben der von bwang8 genutzten Textbox (im Konstruktionswerkzeug) wie auch unterhalb des Eingabefeldes für Chatbeiträge (im Chatwerkzeug) angezeigt.

[9] Die Abbildung entstand unter Zuhilfenahme der Sitzungsdaten einer realen Kollaboration, die jedoch mit einer älteren *ConcertChat*-Version erfolgte, in welcher die *Awareness*-Informationen noch nicht integriert dargestellt wurden.

6 Bewertung und Erfahrungen

In Kapitel 2 wurden drei Problemfelder dualer Interaktionsräume identifiziert: Die medial bedingten Schwierigkeiten bei der deiktischen Bezugnahme auf Artefakte im gemeinsamen Arbeitsbereich, bei der Koordination der Aktivitäten in den beiden Interaktionsbereichen und bei der Reflexion über den sich über beide Interaktionsbereiche erstreckenden Kollaborationsverlauf. Um die Probleme der Interaktion in dualen Interaktionsräumen zu überwinden, wurden neun Anforderungen an die Integration von Chat- und Konstruktionswerkzeugen abgeleitet. Kapitel 4 stellte eine Konzeption von Integrationsmaßnahmen entsprechend dieser Anforderungen vor. Auf deren konkreten Umsetzung in dem Kollaborationssystem *ConcertChat* ging Kapitel 5 ein.

Im Folgenden werden die Integrationsmaßnahmen bewertet und über die Erfahrungen mit der Umsetzung und Anwendung berichtet. In Kapitel 6.1 wird die Wirksamkeit der Integrationsmaßnahmen evaluiert. Hierzu werden Auszüge einer kollaborativen Sitzung präsentiert und mit dem Ziel analysiert, ob und wie Lernende von den Integrationsmaßnahmen nutzbringend Gebrauch machen. Ergänzt werden die Befunde durch Ergebnisse von Experteninterviews. Anschließend werden in Kapitel 6.2 die Umsetzbarkeit und Generalisierbarkeit der softwaretechnischen Modellierung diskutiert.

6.1 Evaluation der Integrationsmaßnahmen

6.1.1 Evaluationsziele und -methoden

Gegenstand dieser Arbeit sind softwaretechnische Maßnahmen, mit denen das kollaborative Lernen in dualen Interaktionsräumen erleichtert bzw. unterstützt wird. Hierfür wurden als drei Problemfelder die deiktische Bezugnahme, die Koordination und die Reflexion identifziert und neun Integrationsmaßnahmen entwickelt. Wie können diese Integrationsmaßnahmen nun auf ihre Wirksamkeit hin überprüft werden? Über die hierfür angemessenen Methoden wird seit längerem diskutiert (z. B. Salomon, 1992; Friedrich et al., 2001; Fjuk & Ludvigsen, 2001; Pfister, 2004; Stahl et al., 2006a; Reimann, 2007). Nicht unwesentlich wird dieser Methodenstreit von der jeweiligen fachlichen (Psychologie, Pädagogik, Informatik etc.) Herkunft[1] getragen.

Die inhärente Schwierigkeit bei der Evaluation von Kollaborationssoftware besteht darin, dass die Software nur eines von vielen Elementen im jeweiligen Lehr-Lernarrangement ist. Erfolg oder Misserfolg beim Lernen sind das Resultat vielfältiger, sich wechselseitig beeinflussender Faktoren. Ein kausaler Zusammenhang zwischen einzelnen Funktionen der Software und dem Lernerfolg ist deshalb nur schwer nachzuweisen. Zwar konnte für die explizite Referenzierung in einer experimentalpsychologischen Studie[2] ein positiver Effekt auf das kollaborative Lernen gezeigt werden (Pfister et al., 2004; Mühlpfordt & Wessner, 2005), ein Nachteil experimenteller Studien ist jedoch die Notwendigkeit, die Versuchsbedingungen stark kontrollieren und damit das Verhalten der Gruppe beim Lernen beschränken zu müssen. So wurden z. B. in der zitierten Studie die Interaktionsmöglichkeiten der Teilnehmer dahingehend beschränkt, dass sie vor dem Versenden eines Beitrags jeweils eine explizite Referenz setzen mussten.

Im Folgenden wird deshalb für die Bewertung der Integrationsmaßnahmen ein anderer Weg verfolgt. Ziel ist es zu prüfen, ob die Integrationsmaßnahmen in „alltagsnahen“ Kollaborationen die Interaktion erleichtern. Hierzu werden Auszüge von Kollaborationstranskripten daraufhin analysiert, ob und in welchen Situationen die Maßnahmen für die Kollaborierenden relevant sind. Die Transkripte entstanden bei der kollaborativen Bearbeitung mathematischer Probleme durch Gruppen Jugendlicher unter Verwendung des *ConcertChat*-Systems. Für die Jugendlichen war die Software das alleinige Medium für die gemeinsame Aktivität, sie waren frei im Gebrauch oder Nichtgebrauch der Integrationsmaßnahmen. Ergänzt werden die Analysen durch Experteninterviews. In diesen wurden regelmäßige Anwender des Systems nach ihren Erfahrungen befragt.

[1] Neben der fachlichen scheint auch die geographische Herkunft entscheidend zu sein. Während z. B. die deutsche CSCL-Forschung stark von experimentalpsychologischen, relevante Faktoren identifizierenden Ansätzen geprägt ist, neigt die nordamerikanische Forschung zur Verwendung qualitativer Untersuchungsmethoden.

[2] Diese Studie wurde von der Deutschen Forschungsgemeinschaft (DFG), Projekt PF330/1-1, im Rahmen des DFG-Schwerpunktprogramms „Netzbasierte Wissenskommunikation in Gruppen“ gefördert.

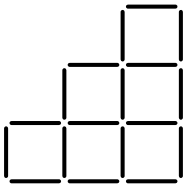


Abbildung 6.1: Ausgangsmuster für das Hölzchenproblem

6.1.2 Kollaborationsanalysen[3]

Hintergrund

Die im Folgenden vorgestellten Episoden entstammen kollaborativen Sitzungen, die im Rahmen des Projektes „*Virtual Math Teams @ the Math Forum*“ (VMT) stattfanden. Das *Math Forum* ist eine seit 1992 existierende offene Online-Community[4], die den Austausch zwischen Mathematiklehrern, Schülern, Mathematikern und anderen an der Mathematiklehre und -weiterbildung interessierten Personen zum Ziel hat (Renninger & Shumar, 2002).

Im VMT-Projekt wird der Nutzen neuer, innovativer Kollaborationswerkzeuge für die Mathematikausbildung untersucht, wobei durch die Werkzeuge kleinen Gruppen örtlich verteilter Lernender ermöglicht werden soll, kollaborativ mathematische Probleme zu bearbeiten. Hierbei kam auch der *WhiteboardChat* zum Einsatz. Basierend auf *ConcertChat* wurde die Lernumgebung *VMTChat* entwickelt, mit der Aufgaben, Kollaborationsräume und Gruppen organisiert und kollaborative Sitzungen durchgeführt werden können (Wessner et al., 2006).

In den Jahren 2005 und 2006 wurden Pilotstudien durchgeführt. Es nahmen daran Schüler verschiedener Schulen aus den USA teil, die durch im *Math Forum* aktive Lehrer angeworben wurden. Zwischen den Gruppen wurde ein Wettbewerb um die am meisten kollaborative Gruppe mit einem Preis für die Gewinner ausgelobt.

Durchführung

Als Kollaborationsinhalt wurde den Gruppen vorgegeben, die Eigenschaften und Gesetzmäßigkeiten geometrischer Figuren, wie sie aus Streichhölzern gebildet werden können, zu erkunden. Hierfür wurde ihnen als Ausgangspunkt ein aus Quadraten geformtes Muster (siehe Abb. 6.1) mit der Anregung präsentiert, Formeln für die Anzahl der Quadrate und der verwendeten Hölzer in Abhängigkeit von der Größe des Musters zu suchen. Davon ausgehend sollten sie andere Muster untersuchen, die sie oder die anderen Gruppen gefunden haben.

[3] Dieses Kapitel profitiert von meinen Aufzeichnungen, die ich während der *Data Sessions* des VMT-Teams an der *Drexel University* in Philadelphia, USA, machte: Einmal in der Woche trifft sich das Team und analysiert kollaborativ und sehr detailliert einzelne Episoden von Sitzungsmitschnitten.

[4] Siehe `http://www.mathforum.org`.

Transkript 6.1: Beginn der ersten Sitzung. Vor jeder Zeile ist die fortlaufende Nummer sowie (in Klammern) der zeitliche Abstand zur vorherigen Aktion in Sekunden angegeben. Die *kursiv* gesetzten Zeilen beschreiben Aktionen im gemeinsamen Arbeitsbereich.

22 (14) Mentor: Today's session is mainly to get to know the VMT system
23 (19) Aznx: So, how do you use the whiteboard?
24 (8) *bwang8 zeichnet Linie*
25 (4) *bwang8 löscht die Linie*
26 (22) Mentor: I will answer your questions as you start to do things
27 (8) bwang8: asdfadsf
28 (0) bwang8:
29 (8) *bwang8 zeichnet Linie*
30 (3) *bwang8 löscht die Linie*
31 (6) bwang8: ok
32 (12) Mentor: The whiteboard is a shared area for drawing and textboxes
33 (29) Mentor: What you put there is seen by everyone in this room, just like the chat messages
34 (22) Aznx: It seems they only last for a while thoough/
35 (8) bwang8: i delete them
36 (21) Aznx: Does that mean in the real time, all the work would be there?
37 (0) bwang8: can you see main topic?
38 (5) Mentor: Since things are shared, you should be careful about deleting
39 (10) bwang8: no, my things
40 (8) bwang8: just testing
41 (4) bwang8: functions
42 (16) *bwang8 zeichnet Linie*
43 (2) *bwang8 löscht die Linie*
44 (2) *bwang8 zeichnet Linie*
45 (5) Mentor: remember, you are trying to collaborate
46 (2) *Quicksilver löscht die Linie*
47 (4) bwang8: ok

Jede Kleingruppe bestand aus drei bis vier Schülern und traf sich in vier einstündigen Sitzungen innerhalb von zwei Wochen. Im Anschluss an jede Sitzung wurde den Gruppen Rückmeldung vom VMT-Team gegeben, indem ihnen auf dem *Shared Whiteboard* ein Text hinterlassen wurde.

Während der Sitzungen hielt sich ein Mentor im virtuellen Kollaborationsraum auf. Die Aufgabe der Mentoren beschränkte sich im Wesentlichen darauf, im Falle technischer Probleme oder Fragen Hilfestellung zu geben; sie sollten ansonsten möglichst wenig in die Kollaboration eingreifen.

Als Kollaborationsanwendung kam ein leicht modifizierter[5] *WhiteboardChat* zum Einsatz. Die verwendete Version erlaubte die explizite Referenzierung und verfügte über eine Konstruktionshistorie des gemeinsamen Arbeitsbereiches, enthielt jedoch noch nicht die Funktionen zur Kontextrekonstruktion; die Aktivitätsdokumentation wurde während der Studie hinzugefügt.

Die Schüler waren mit dem *WhiteboardChat* nicht vertraut. Sie erhielten keine separate Einführung in den Gebrauch, sondern Fragen wurden während der Sitzungen durch die Mentoren beantwortet. Dies illustrierend sind im Transkript 6.1 die Aktivitäten vom Beginn der ersten Sitzung wiedergegeben. Sie erstrecken sich über einen Zeitraum von vier Minuten und geschahen unmittelbar nach der Begrüßung. Aznx fragt in Zeile 23 direkt danach, wie das Whiteboard zu benutzen

[5] In der Benutzungsschnittstelle wurden zwei zusätzliche Buttons untergebracht. Der erste erlaubte es, eine Email an das VMT-Team bzw. die Gruppenmitglieder zu verschicken. Der zweite zeigte die Arbeitsanweisung in einem neuen Fenster.

Tabelle 6.1: Anzahl der Chatbeiträge (Spalte C), der Chatbeiträge mit Referenz (Spalte R) und der Aktionen im gemeinsamen Arbeitsbereich (Spalte A) pro Sitzung und Teilnehmer.

	1. Sitzung			2. Sitzung			3. Sitzung			4. Sitzung			Gesamt		
Teilnehmer	C	R	A	C	R	A	C	R	A	C	R	A	C	R	A
Aznx	91	0	35	112	1	17	159	5	74	279	3	214	641	9	340
bwang8	72	3	78	123	10	148	67	1	89	191	7	70	453	21	385
Quicksilver	51	7	12	180	11	140	161	8	133	194	16	27	586	42	312
Mentor	34	15	11	16	9	25	17	6	4	33	4	0	100	34	40
Summe	248	25	136	431	31	330	404	20	300	697	30	311	1780	106	1077

sei, worauf durch den Mentor in den Zeilen 32, 33 und 38 geantwortet wird. Unterdessen probiert bwang8 das *Whiteboard* aus, indem sie[6] verschiedene Linien zeichnet und wieder löscht (Zeilen 24 & 25, 29 & 30, 42 & 43). Die in Zeile 44 durch bwang8 gemalte Linie wird durch Quicksilver in Zeile 46 entfernt. Das scheinbar sofortige Verschwinden der Linien wird von Aznx anfangs falsch gedeutet: „It seems they only last for a while thoough/“ (Zeile 34); dies wird von bwang8 jedoch sofort korrigiert: „i delete them“ (Zeile 35).

Quantitative Analyse

Im Folgenden werden Episoden aus den Sitzungen dieser Gruppe präsentiert. Zwei Teilnehmer gingen in dieselbe Schule und kannten sich, der dritte kam aus einem anderen Bundesland der USA. Die gesamte Kollaboration über die vier Sitzungen umfasste 2857 Aktionen im Chat und im gemeinsamen Arbeitsbereich (siehe Tab. 6.1). Die drei Teilnehmer beteiligten sich in etwa gleichermaßen an der Kollaboration (Aznx: 34%, bwang8: 30%, Quicksilver: 31%), während entsprechend seiner Rolle der Moderator nur wenig eingriff (5% der Gesamtaktivität).

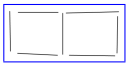
Von den 2857 Aktionen in der Kollaboration wurden 62% im Chat (1780 Beiträge) und 38% im gemeinsamen Arbeitsbereich (1077 Manipulationen an den gemeinsamen Artefakten) getätigt. Insofern ist dies ein gutes Beispiel für eine sich über beide Interationsbereiche erstreckende Kollaboration.

Die bereitgestellte Funktion zur expliziten Referenzierung wurde von den Jugendlichen für 4% (72) ihrer Beiträge genutzt, wobei 53% der Referenzen in den gemeinsamen Arbeitsbereich zielten.

Beachtlich hierbei ist, dass die Referenzierung – da bisher für die Teilnehmer unbekannt und in ihrem Gebrauch nur bei Nachfrage erklärt – erst als zusätzliche konversationale Ressource für die Kommunikation entdeckt und etabliert werden muss. Sie „konkurriert“ mit den in Kapitel 2.3.3 aufgeführten Konventionen in

[6] Das Geschlecht der Teilnehmer ist unbekannt. Im Folgenden wird der Einfachheit halber davon ausgegangen, das es sich um Mädchen handelt.

Transkript 6.2: Explizite Referenz zum Wiederaufgreifen aus Historie

287 (5) Quicksilver: Well, anyway, you can see a pattern that the amount of squares increases by the n. For the sticks, The bottom row's square on the right has 2 new sticks. All the squares in the new row to the left of it have 3 new sticks. So, If te row has 5 squares, 4 of the squares have 3 sticks, the last on only has two. For the enitre Figure, you would add the amount to the previous ammount
. . .
459 (17) bwang8: what was your pattern of growth, quicksiler
460 (72) Quicksilver: i think it was something about the amount of squares that increased with each row....and how one of the new squares had 3 new sticks while the other new ones had 2 new sticks
461 (17) *Quicksilver beginnt Zeichnung*
462 (4) bwang8: oh, ok
463 (0) *Quicksilver vollendet Zeichnung (6 Schritte, 19s):* 
469 (8) Quicksilver: i drew some sqaures
470 (11) Quicksilver: the left one had three new sticks
471 (14) Quicksilver: the right one has a new stick on the bottom and on the right
472 (6) Quicksilver: the top one is from an old square
473 (1) Mentor: It was at 7:00:39 – to get the old messages, click on the icon above here with the two circular arrows [Ref: 287]
474 (12) *bwang8 verschiebt eine Textbox*
475 (5) *Quicksilver zeichnet Linie*
476 (72) Quicksilver: yea it's at the very top. but i think there ae errors in that paragraph [Ref: 287]
477 (15) Quicksilver: yea that's wrong

der Chatkommunikation. Im nun folgenden Abschnitt werden Fälle analysiert, in denen die Teilnehmer die explizite Referenzierung einsetzen.

Anwendung der expliziten Referenzierung

Wiederaufgreifen älterer Beiträge Am Ende der ersten Sitzung wurde von Quicksilver eine entdeckte Gesetzmäßigkeit darüber zusammengefasst, wieviele neue Hölzchen für eine weitere Reihe an Quadraten benötigt werden (Zeile 287 im Transkript 6.2). Zu Beginn der zweiten Sitzung fragt bwang8 in Zeile 459 Quicksilver danach. Quicksilvers Antwort erstreckt sich über die Zeilen 460 bis 472 und beide Interaktionsbereiche: Nachdem sie im Chat über die erinnerte Gesetzmäßigkeit (Zeile 460) berichtet hat, beginnt sie diese im Whiteboard zu illustrieren (Sie zeichnet eine Reihe mit zwei Quadraten), um dann abschließend diese Illustration wiederum im Chat zu erläutern (Zeilen 469 bis 472). Hier greift der Mentor ein mit einem Chatbeitrag, der auf den gesuchten alten Beitrag von Quicksilver verweist und gleichzeitig Auskunft gibt, wie die Historie im Chat nachgeladen werden kann. Quicksilver greift dies mit ihrem durch ein fortsetzendes „yea" eingeleiteten Beitrag (Zeile 476) auf, weist aber gleichzeitig darauf hin, dass ihre alte Vermutung Fehler enthielte.[7]

An dieser kurzen Episode sind mehrere Dinge beachtenswert:

- Im Beitrag 469 weist Quicksilver auf den Wechsel der Interaktionsbereiche hin („I drew some sqaures"). Dieser Beitrag hat mehrere Funktionen; zum

[7] Jedoch ist auch die neue Hypothese Quicksilvers nicht korrekt.

einen weist er auf die Illustration im *Whiteboard* hin[8], zum anderen lenkt er über auf die nun folgende Erläuterung und etabliert hierfür die gezeichneten Quadrate als Bezugspunkt.

- Quicksilvers räumliche Bezugnahme in Beitrag 476 („it's at the very top") legt nahe, dass hier ein WYSIWIS zumindest in dem Sinne erwartet wird, dass die anderen den Beitrag 287 auch bei sich ganz am Anfang der Chathistorie finden werden. Dies ist zwar in diesem Falle wahrscheinlich (denn sofern noch nicht lokal vorhanden, werden in *ConcertChat* alle Beiträge bis zum referenzierten nachgeladen), trifft aber nicht immer zu, denn die anderen Teilnehmer könnten sich selbstständig bereits größere Teile der Chathistorie geladen haben.

- Beide expliziten Referenzen auf den sehr weit zurückliegenden Beitrag enthalten jeweils noch andere deiktische Elemente: Quicksilver verwendet den erwähnten räumlichen Hinweis; im Beitrag vom Mentor findet sich die Uhrzeit. Die expliziten Referenzen allein erscheinen für diese in der Historie weit zurück verweisenden Bezüge nicht ausreichend zu sein.

- Man beachte die beiden langen Pausen vor den Beiträgen 460 und 476, die jeweils 72 Sekunden lang sind. Im Gegensatz zu einem Gespräch, in dem eine Pause von über einer Minute sehr selten ist, wird dies von den Teilnehmern hier offenbar toleriert.

Einbeziehen von Objekten im *Whiteboard* Die im Transkript 6.3 abgebildete Interaktionssequenz zeigt weitere Funktionen expliziter Referenzen. Eingeleitet wird die Sequenz mit bwang8s Ankündigung, sie habe eine Formel gefunden (Zeile 893). Quicksilver reagiert mit einem abwartenden „lets see". bwang8 schreibt die Formel in eine Textbox des gemeinsamen Arbeitsbereichs. Mit ihrem nächsten Beitrag, der auf den Bereich des gemeinsamen Arbeitsbereichs, in dem die Textbox liegt, mit einer expliziten Referenz verweist, erläutert sie die Formel. Interessant hierbei ist wieder das lange Schweigen von 43s zwischen dem Veröffentlichen der Formel und dem nächsten Beitrag. Hier kann die Information zur Tipp-*Awareness* Aufschluss geben: Bereits 16 Sekunden nach dem Schreiben der Formel beginnt bwang8 mit dem Schreiben einer Chatnachricht. In den nächsten Beiträgen (Zeilen 897 bis 905) leitet bwang8 zur nächsten Formel über, die sie dann mit dem Freihandwerkzeug im *Whiteboard* schreibt. Im Gegensatz zur Textbox hat dies in *ConcertChat* den Vorteil, dass das Schreiben direkt beobachtet werden kann. Dies führt dazu, dass Quicksilver bereits nach dem Erscheinen des Summenzeichens in Zeile 932 reagieren kann. Hierbei nutzt sie jedoch – offenbar in Erwartung, dass diese Bezugnahme verstanden wird – eine rein sprachliche Referenz: „I forgot how to solve that thing". Gleichzeitig verdeutlicht dies aber auch, dass die neue Formel

[8] Hier sei daran erinnert, dass zu dem Zeitpunkt der *WhiteboardChat* noch keine Aktionsmarkierungen bereitstellte.

Transkript 6.3: Explizite und „implizite“ Referenzen

893 (5) bwang8: i think i got the equation for the middle sticks [Ref: Whiteboard]
894 (8) Quicksilver: All right...lets see
895 (5) *bwang8 erzeugt Textbox* top/bottom: 2n(n+1) middle: (n+1)^2
896 (43) bwang8: now we know the n by n blocks on the bottom [Ref: Ausschnitt mit Textbox]
897 (6) Aznx: Yeah it seems so.
898 (5) *bwang8 verändert Größe der Textbox*
899 (2) Quicksilver: yes.
900 (33) bwang8: so we can use recursive equation to figure out n 3d pryamind
901 (6) bwang8: stick number
902 (6) bwang8: lol
903 (6) bwang8: bad grammar
904 (7) Quicksilver: yea
905 (3) Quicksilver: w/e
906 (57) *bwang8 beginnt eine Formel zu schreiben (26 Schritte, 63s)*
932 (1) Quicksilver: Ahh... I forgot how to solve that thing...We learned it like two months ago
933 (4) *bwang8 setzt Formel fort (3 Schritte, 7s)*
936 (1) Quicksilver: Could someone remind me
937 (1) *bwang8 vollendet Formel (7 Schritte, 13s):* $\sum_{n=1}^{n} = 4n(n+1) + (n+1)^2$
944 (27) bwang8: i don't know if this equation is right
945 (14) Aznx: I don't exactly remember either. :P
946 (5) bwang8: the top n is the highest
947 (10) Quicksilver: It goes up by one every time because n=1 rite?
948 (19) bwang8: no, n start at 1
949 (8) Quicksilver: oh yeah!!!
950 (0) bwang8: this is a recursive function
951 (0) Quicksilver:
952 (39) bwang8: when n=1, plug 1 into the right equation\
953 (15) Quicksilver: 12?
954 (18) bwang8: when n=2, plug 2 into the right equation and add the equation when n=1
955 (14) Quicksilver: Oh...so u add the thing that came before
956 (4) bwang8: yes
957 (14) bwang8: 12 is right [Ref: 953, „12“]
958 (24) bwang8: since when there is only one cube the number of sticks is 122
959 (5) bwang8: i mean 12

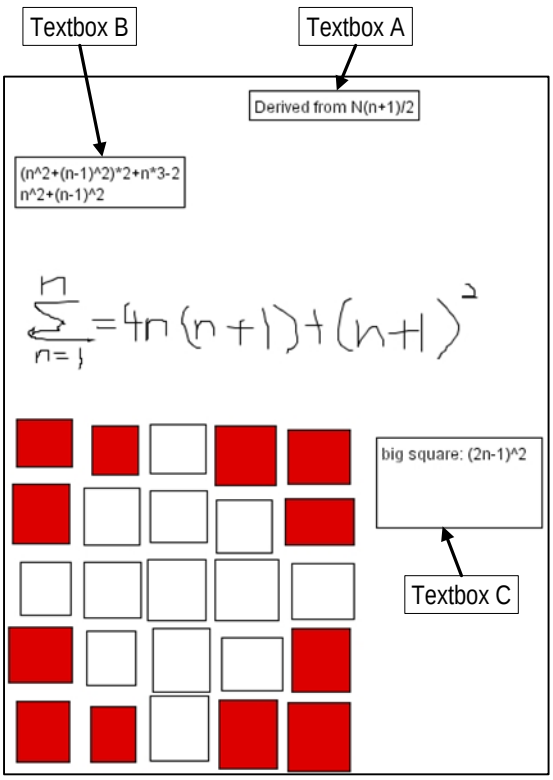
in ihrem gemeinsamen Bezugsrahmen liegt. Darauf deutet auch die Verwendung des unspezifischen „this equation“ in bwang8s folgendem Beitrag 944 hin.

In den Zeilen 946 beginnt bwang8 mit der Erklärung der (nicht ganz korrekt geschriebenen) Formel. Diese setzt sich fort in den Beiträgen 950, 952 und 954. Interessant ist nun das „yes“ in Beitrag 956. Dieses ist insofern mehrdeutig, als dass es sowohl eine Reaktion auf den adjazenten Kommentar von Quicksilver in Zeile 955 als jedoch auch eine Antwort auf die Frage von Quicksilver in Zeile 953 („12?“) sein kann. Geradezu im Kontrast zu dieser Ambiguität steht der nächste Beitrag von bwang8 („12 is right“), der seinen Bezug gleich zweifach herstellt: Es wird Quicksilvers „12“ sowohl sprachlich aufgenommen als auch explizit referenziert. Es liegt die Vermutung nahe, dass der Wechsel im Bezugsrahmen (vorher Beitrag 955, nun 953) besonders deutlich gemacht werden soll. Anhand der folgenden Interaktion kann dies jedoch nicht aufgeklärt werden.

Die flexible Nutzung der Referenzierungsfunktion zeigt sich auch im folgenden Beispiel (siehe Transkript 6.4). Hier diskutiert die Gruppe, über welche Formel sich die Anzahl der weißen Quadrate in dem abgebildeten Muster berechnen lässt. In den Zeilen 2210 bis 2220 führt bwang8 – geradezu didaktisch – eine Formel für die Anzahl der Quadrate in einer Ecke ein. In Zeile 2224 erweitert sie diese Formel auf die vier Ecken. Diese Anzahl muss von der Gesamtzahl aller Quadrate im Muster subtrahiert werden. Für die Gesamtzahl hat die Gruppe bereits eine Formel

Transkript 6.4: Nutzung der expliziten Referenz für Formelableitung

2210 (11) bwang8: i think the 4 corner is growing like this
2211 (9) bwang8: 0,1,3,6,10
2212 (5) bwang8: what is the pattern
2213 (8) Aznx: Triagnular numbers.
2214 (2) Quicksilver: triangular numbers!
2215 (2) bwang8: yep
2216 (3) Aznx: We had already figured that out.
2217 (7) bwang8: we can use the equation from session 1
2218 (0) Quicksilver: yes
2219 (9) Aznx: Yup.
2220 (16) bwang8: n(n+1)/2
2221 (3) *Quicksilver verschiebt Textbox B*
2222 (2) *Quicksilver verschiebt Textbox A*
2223 (13) *Aznx verschiebt Textbox B*
2224 (2) bwang8: 4*n(n+1)/2= the four corners
2225 (1) Quicksilver: this right? [Ref: Bereich mit Textbox A]
2226 (2) *Aznx verschiebt Textbox B*
2227 (2) *Aznx verschiebt Textbox B*
2228 (2) bwang8: yes
2229 (3) Aznx: Yeah
2230 (22) bwang8: (2n-1)^2-2n(n-1) [Ref: Textbox C]
2231 (20) bwang8: this is the equation for each level



gefunden und in Textbox C dokumentiert („big square: (2n-1)^2“). Bwang8s Beitrag in Zeile 2230 ist nun mit dieser Textbox verknüpft. Die explizite Referenz geht hier über ein reines Zeigen hinaus, mit ihr wird ein für die Ableitung notwendiger Zwischenschritt in die Argumentation aufgenommen.

In den Zeilen 2225, 2228 und 2229 zeigt sich der souveräne Umgang der Gruppe mit einer Phantomadjazenz (Garcia & Jacobs, 1998). Unmittelbar nach bwang8s Beitrag mit der Formel für die vier Ecken trifft Quicksilvers Nachfrage „this right?“ ein. Der Beitrag ist mit einer expliziten Referenz im Sinne einer Zeigegeste auf die Textbox mit der Formel für die Anzahl der Quadrate in einer Ecke versehen. Dieser Beitrag ist jedoch eine Reaktion auf Zeile 2220 und wird als solcher auch von den beiden anderen Teilnehmern verstanden.

Wahl des Interaktionsbereichs

Wie im folgenden Beispiel (siehe Transkript 6.5) zu sehen, ist es nicht immer einfach, den richtigen Interaktionsbereich auszuwählen. Die Gruppe ist seit einiger Zeit dabei, sich eine Formel, die von einer anderen Gruppe gefunden wurde, zu erklären. In Zeile 2409 schreibt Aznx begeistert („Omg“ = Oh my god), sie habe die

Transkript 6.5: Simultane Nutzung beider Interaktionsbereiche

2409 (8) Aznx: Omg, I just solved their formula... and it turns out to be n(4n-1)!
2410 (2) Quicksilver: so its symmetric?
2411 (4) Aznx: what in the world?
2412 (4) Aznx: am i going crazy?
2413 (0) bwang8: don't consider the 4 cornors
2414 (3) Aznx: someone check my work.
2415 (6) Aznx: simplify their formula
2416 (15) Quicksilver: k
2417 (4) bwang8: what do you mean
2418 (35) Aznx: 2(n^2+n^2-2n+1)+3n-2
2419 (4) bwang8: i don't see how you can simplify it
2420 (2) Aznx: simply the formula
2421 (4) Aznx: for the number of sticks
2422 (5) Aznx: so that simplifies to...
2423 (60) Aznx: I stil get the same.
2424 (35) bwang8: how did you simplify it
2425 (7) Aznx: um
2426 (5) Aznx: square the n-1
2427 (7) Aznx: then multiply the whole thing by 2
2428 (8) Aznx: then multiply the 3 and n
2429 (5) Aznx: and add it with that
2430 (5) Aznx: and subtract by 2
2431 (18) bwang8: quicksliver
2432 (5) Quicksilver: im lost
2433 (4) bwang8: did you get the same answer
2434 (6) Quicksilver: no
2435 (9) Aznx: i'll do it on the board
2436 (5) Quicksilver: yeah
2437 (3) *Aznx beginnt mit Formel (3 Schritte, 5s)*
▪▪▪
2440 (1) Quicksilver: i got something totally difrent
2441 (0) *Aznx setz Formel fort (24 Schritte, 42s)*
▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪▪
2465 (1) bwang8: so far i got $4*n^2+3*n$
2466 (1) ▪▪▪▪▪▪▪▪▪▪▪ (11 Schritte, 18s)
2477 (1) Quicksilver: indranil rite in the box
2478 (0) ▪▪▪▪▪▪▪▪▪▪▪▪▪ (13 Schritte, 21s)
2491 (1) bwang8: i mean 4n^2-n
2492 (0) ▪▪▪▪▪ (5 Schritte, 6s)
2497 (3) Aznx: EXactly
2498 (1) ▪▪▪▪▪ (5 Schritte, 11s)
2503 (2) Quicksilver: yea that waht azn x got eralier
2504 (1) ▪▪▪▪▪▪▪▪▪ (9 Schritte, 19s)
2513 (0) bwang8: holy
2514 (1) ▪▪▪ (3 Schritte, 2s)
2517 (0) bwang8: moley
2518 (1) *Aznx beendet Formel (2 Schritte, 1s)*
▪▪
2520 (0) Quicksilver: whyd u multiply by the two [Ref: „2“ in Formel]

Formel gelöst. Diese Begeisterung setzt sich in den folgenden vier Beiträgen fort mit der Aufforderung, die Formel zu vereinfachen. Bwang8s Reaktion („what do you mean“, Zeile 2417) führt zu einer Reihe von 11 Beiträgen seitens Aznx, in der sie die Schritte zur Vereinfachung der Formel beschreibt. Doch die beiden anderen scheinen nicht zu verstehen (2431 bis 2434): bwang8 adressiert Quicksilver, die nur erwidert „im lost“. Hierauf wechselt Aznx den Interaktionsbereich: „i'll do it on the board“. Während der folgenden zwei Minuten schreibt Aznx als Freihandzeichnung und somit für die anderen beobachtbar die Ableitung zu „4n^2-n“ auf die gemeinsame Arbeitsfläche (siehe Abb. 6.2). Erstaunlich ist hier ihr kurzzeitiger Wechsel zurück in den Chat: In Zeile 2491 schreibt bwang8 (offenbar ist sie selbstständig am Vereinfachen) ihr Ergebnis, worauf Aznx unmittelbar mit „EXactly“ antwortet. Quicksilver zeigt mit ihren Beiträgen 2503 und 2520, dass sie beiden Threads (der Ableitung von bwang8 im Chat und der von Aznx im *Whiteboard*) folgt: 2503 ist an bwang8 adressiert (Aznx ist im Beitrag in der dritten Person erwähnt), während 2503 mit einer expliziten Referenz in das *Whiteboard* zeigt und Aznx anspricht.

In dieser Sequenz sind die Aktivitäten der Gruppe über die beiden Interaktionsbereiche verteilt. Aznx ist im zweiten Teil vorrangig im *Whiteboard* und die

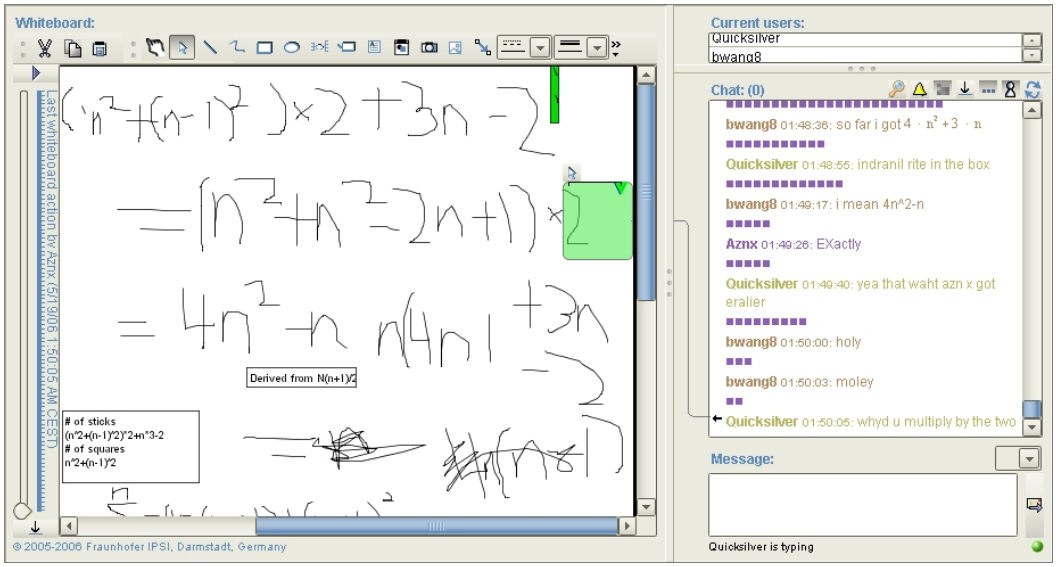


Abbildung 6.2: Simultane Nutzung beider Interaktionsbereiche

anderen beiden sind im Chat aktiv. Dennoch sind die Aktionen stark aufeinander bezogen und miteinander verwoben.

Aktivitätsdokumentation

Zwischen der zweiten und dritten Sitzung wurde eine neue Version des *Concert-Chat*-Systems eingespielt. Diese beinhaltete als Neuerung die Aktivitätsdokumentation. Weder die Teilnehmer noch der Mentor hatten hiervon Kenntnis. Transkript 6.6 zeigt, wie die Aktionsmarkierungen von der Gruppe aufgenommen werden.

Ausgangspunkt ist das Problem, dass der gemeinsame Arbeitsbereich mittlerweile gefüllt ist mit den Zeichnungen der ersten beiden Sitzungen und zweier großer Textboxen mit dem Feedback des VMT-Teams (siehe Abb. 6.3). Quicksilver schlägt deshalb in Zeile 1215 vor, einige Dinge zu löschen. Nach einer kurzen Diskussion (die im Transkript ausgelassen ist) beginnt Quicksilver mit dem Entfernen von Objekten. Ihre Aktionen werden nun im Chat dokumentiert (im Transkript 6.6 analog zu *ConcertChat* durch die farbigen Quadrate verdeutlicht).

Der Beitrag von Aznx in Zeile 1249 greift dies mit einem verwunderten „What's with the squares?“ auf und wird von Quicksilvers „Yeah“ in Zeile 1252 bestärkt. Nach weiteren Löschaktionen von Quicksilver bietet bwang8 in Zeile 1258 als eine Erklärung an, die Quadrate stünden für die Aktionen auf der Tafel. Der Beitrag vom Mentor (Zeile 1259) ist eher eine Fortsetzung von Quicksilvers Beitrag in Zeile 1252. Hierfür spricht, dass er auch mit einem „Yeah“ beginnt, wie auch, dass die Beitragserstellung entsprechend der im Transkript nicht wiedergegebenen Informationen zur Tipp-*Awareness* simultan zu bwang8 begonnen hat.

Transkript 6.6: Die Gruppe entdeckt die Aktionsmarkierungen

1215 (23) Quicksilver: Actually....lets erase some stuff from the whiteboard
. . .
1234 (0) *Quicksilver löscht 15 Objekte in 10s*
1249 (2) Aznx: What's with the squares?
1250 (0) *Quicksilver löscht zwei Objekte in 3s*
1252 (3) Quicksilver: yeah
1253 (7) *Quicksilver löscht 5 Objekte in 3s*
1258 (1) bwang8: i think that's for each black board action
1259 (8) Mentor: Yeah - - I don't know either
1260 (9) Quicksilver: every time i dleted a line a square came
1261 (1) Aznx: Let's just ignore it.
1262 (4) Quicksilver: yes
1263 (7) *bwang8 löscht 3 Objekte in 6 s*
1266 (4) Quicksilver: can i delete first session's comments?
1267 (0) bwang8: yep
1268 (8) *Quicksilver löscht 10 Objekte in 20s*
1269 (0) *Aznx verschiebt simultan 5 Objekte*
1283 (9) bwang8: the squares tell you who is doing each action
1284 (9) Quicksilver: oh

Quicksilver bekräftigt mit ihrem nächsten Beitrag (Zeile 1260) die Beobachtung von bwang8, während Aznx im simultan erstellten Beitrag pragmatisch vorschlägt, sie einfach zu ignorieren[9]. Dem stimmt Quicksilver in Zeile 1262 zu.

Nach einer Reihe weiterer Löschaktionen, die zum Teil simultan erfolgen, zeigt der Beitrag von bwang8 in Zeile 1283, dass die farbigen Quadrate nach wie vor ihre Aufmerksamkeit haben: „the squares tell you who is doing each action". Indem sie diese Erklärung der Gruppe mitteilt, wird deutlich, dass bwang8 ihnen Relevanz beimisst.

[9] Wenn schon der Mentor nicht weiß, was es damit auf sich hat . . .

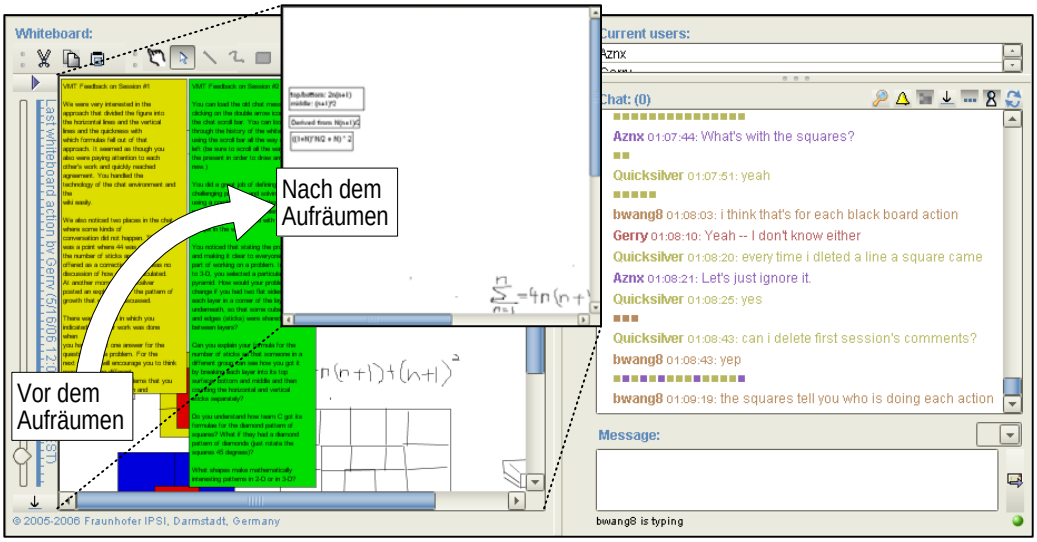


Abbildung 6.3: Die Gruppe entdeckt die Aktionsmarkierungen

Dieser die eigentliche Aufräumaktivität begleitende kurze Diskurs zeigt, dass die in *ConcertChat* gewählte Art der Aktivitätsdokumentation von der Gruppe in kurzer Zeit interpretiert werden kann.

Auf einen weiteren Nutzen der Aktionsmarkierungen für die Interpretation des „Schweigens“ im Chat weisen die Antworten in den Experteninterviews hin, auf die nun eingegangen werden soll.

6.1.3 Experteninterviews

Die Analyse der Kollaborationstranskripte liefert zwar wertvolle Hinweise, wie die Integrationsmaßnahmen in der Kollaboration als Ressourcen genutzt werden. Es sind jedoch nur die sich in konkreten (Inter)Aktionen manifestierenden Funktionen und Probleme erkennbar. Deshalb wurden vier der Mentoren des VMT-Teams in halb-standardisierten Interviews nach ihren Erfahrungen befragt. Hierbei ist zu berücksichtigen, dass sie entsprechend ihrer Rolle in den Gruppensitzungen wenig aktiv sind. Zum Teil setzten sie den *WhiteboardChat* aber auch zu vor- und nachbereitenden Sitzungen mit Lehrern ein.

Die Vorteile der Referenzierung wurden von den Befragten vor allem in Bezug auf das *Whiteboard* gesehen. Die Referenzierung auf andere Beiträge im Chat wird vor allem in größeren Gruppen wichtig. So sei sie in vorbereitenden „*Question & Answer*“-Sitzungen mit Lehrern für die Organisation der Antworten hilfreich gewesen. Von den Schülern wurde die Referenzierung mit Interesse aufgenommen, „*but in the heat of collaboration it goes down until it's really pressing*“.

Einer der Befragten berichtete aus einer eigenen Sitzung, in der das Team selbst ein mathematisches Problem bearbeitete, von einem interessanten Seiteneffekt der Referenzierung in das Whiteboard: „*I was working, doing my thing in the whiteboard. Then people asked me. How I realized it? You know, the stuff was highlighted by the reference.*“ Der Teilnehmer konzentrierte sich auf das Whiteboard und ignorierte den Chat, in dem die anderen nach ihm riefen. Erst eine Referenz, die seinen Bereich grün hervorhob, führte ihn wieder zurück in den Diskurs.

Den umgekehrten Effekt, nämlich die Aufmerksamkeit auf das Whiteboard zu verlagern, haben nach Aussage der Befragten die Aktionsmarkierungen. Dies ist umso wichtiger, da vom gemeinsamen Arbeitsbereich meist nur ein kleiner Ausschnitt sichtbar ist. Sind die anderen Teilnehmer im nicht sichtbaren Bereich aktiv, so ist dies ohne die Aktionsmarkierungen nur schwer erkennbar („*but with the dots I know that something happens*“). Für die Rolle der Mentoren erhält dies eine weitere Relevanz: Längere Zeiten der Inaktivität können nach ihrem Eingreifen verlangen. Durch die Aktionsmarkierungen kann leichter zwischen nicht beobachtbarer Aktivität und tatsächlicher Inaktivität unterschieden werden.

Einem der Befragten erschien die Markierung im ersten Moment redundant zur *Awareness*-Anzeige[10] zu sein („*I thought it's a waste of space*“), im Laufe der Zeit wurde jedoch deutlich, dass sie wesentlich salienter seien.

[10] Im VMTChat wird unterhalb des Whiteboards nach Ausführung einer Aktion kurzzeitig der Name des Akteurs eingeblendet.

Nach Einschätzung eines Befragten führten die Aktionsmarkierungen ähnlich wie die Tipp-*Awareness* dazu, dass das eigene Tippen unterbrochen und auf das Ergebnis der Aktion der Anderen (also den simultan erstellten Chatbeitrag bzw. die vollständige Manipulation, sofern sie sich aus mehreren Schritten zusammensetzt) gewartet wird.

6.1.4 Diskussion

Die Analyse der Kollaborationstranskripte wie auch die Befragung der Experten ergab, dass die explizite Referenzierung von den Teilnehmern als Ausdrucksmittel für deiktische Bezugnahmen situationsspezifisch eingesetzt wird. Die Aktionsmarkierungen werden in der intendierten Art und Weise interpretiert und nach Aussage der Nutzer zur Koordination der eigenen Aktivität genutzt. Über die Maßnahmen zur Kontextrekonstruktion konnte in den Studien keine Aussage getroffen werden, da sie in der eingesetzten Version noch nicht verfügbar waren.

Weitere Analysen kollaborativer Prozesse im VMTChat finden sich z. B. in Stahl et al. (2006b), Cakir et al. (2007), Sarmiento & Stahl (2007a) sowie Zhou et al. (2007). Vertiefende Analysen derselben Gruppe finden sich in Stahl (2007) sowie Zemel et al. (2007).

6.2 Erfahrungen bei der softwaretechnischen Umsetzung

Für die Bewertung der *softwaretechnischen* Konzeption stehen die folgenden Fragen im Vordergrund: Sind die Integrationsmaßnahmen entsprechend dem Konzept mit vertretbarem Aufwand umsetzbar? Und auf welche Art von Applikationen sind sie anwendbar?

Hierbei wird auf Erfahrungen zurückgegriffen, die bei der Entwicklung unterschiedlicher Kollaborationsanwendungen auf Basis von *ConcertChat* gemacht wurden. Das *ConcertChat*-System wurde in einer Reihe von Projekten[11] eingesetzt und weiterentwickelt. Die in den Projekten entstandenen Anwendungen kombinieren jeweils einen Chat mit unterschiedlichen Arbeitsmaterialien (siehe Abb. 6.4), auf die explizit referenziert werden kann:

WebPageChat: Damit können Gruppen Webseiten zusammentragen und über deren Inhalte diskutieren. Der Arbeitsbereich enthält ähnlich einem Webbrowser ein Adressfeld, mit dem neue URLs eingegeben bzw. bereits in der Sammlung befindliche ausgewählt werden können, und die Anzeige der gerade ausgewählten Webseite. Neue Adressen werden erst dann der gemeinsamen Sammlung hinzugefügt, wenn ein Chatbeitrag mit einer expliziten Referenz auf sie verschickt wird.

[11] Unter diesen Projekten waren: das DFG-Projekt „Lernprotokolle“, das BMBF-Projekt „ALBA“, das interne Forschungsprojekt „*ConcertChat* “, dem das System seinen Namen verdankt, das oben erwähnte VMT-Projekt, das EU-Projekt „Mapper“.

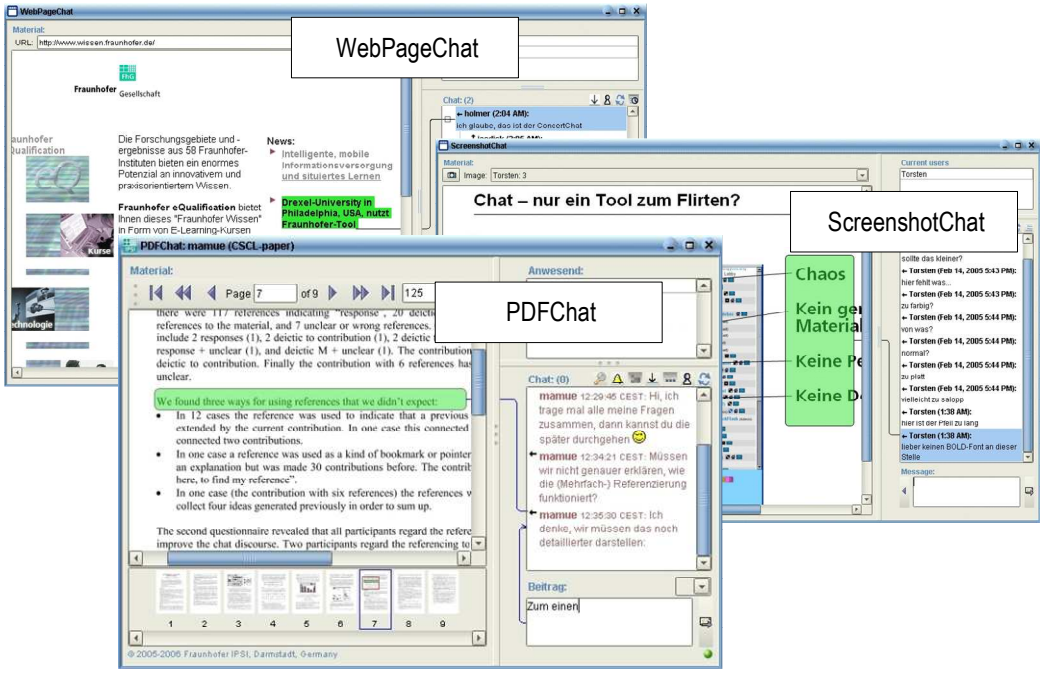


Abbildung 6.4: Die Kombination von Chat mit unterschiedlichen Artefakttypen

ScreenshotChat: Analog zum *WebPageChat* können mit diesem Screenshots bzw. Bilder gesammelt und über diese diskutiert werden. Ein importiertes Bild wird erst dann der gemeinsamen Sammlung hinzugefügt, wenn ein Chatbeitrag mit einer expliziten Referenz auf sie verschickt wird.

MeetingChat: Der *MeetingChat* zielt auf die Unterstützung von Meetings über ein Agendawerkzeug. Die Meetingagenda wird von den Teilnehmern asynchron vor der Sitzung über ein Wiki-artiges Webinterface zusammengestellt. Sie steht als referenzierbares Material im Kollaborationsraum zur Verfügung. Dort können die Teilnehmer den Diskussionsbedarf zu einem Agendapunkt signalisieren. Ein Moderator kann Agendapunkte als neues Thema auswählen; es wird daraufhin automatisch eine entsprechend lautende und auf den Agendapunkt referenzierende Chatnachricht erzeugt.

PDFChat: Dieser stellt im Arbeitsbereich ein PDF-Dokument bereit, auf das explizit referenziert werden kann.

WhiteboardChat: Der *WhiteboardChat* kombiniert einen Chat mit einem *Shared Whiteboard*, wobei dessen Werkzeugpalette und damit die erzeug- und manipulierbaren Objekttypen erweiterbar sind. Für das Hinzufügen weiterer Objekttypen bedarf es keiner Änderung an den Integrationsmaßnahmen.

Das auf der totalen Ordnung beruhende Konzept des in *ConcertChat* umgesetzten persistenten Nachrichtenkanals, der einen selektiven Zugriff auf die Kanal-

historie erlaubt, hat sich als sehr mächtig herausgestellt. Zum einen ist damit die **Kombination der Interaktionshistorien** über die globale Indizierung leicht realisierbar. Zum anderen erlaubt es, verschiedene Arten gemeinsamer Datenmodelle, die über als Nachrichten verschickbare Operationen manipuliert werden, zu erzeugen. Hierüber sind die gemeinsamen Datenmodelle für die genannten Anwendungen, wie z. B. die Datenmodelle für die Adress- bzw. Bildersammlungen sowie die Meetingagenda, umgesetzt.

Der Aufwand für die Umsetzung der **expliziten Referenzierung** liegt im Wesentlichen in der Darstellung. Seitens der *Groupware*-Infrastruktur muss allein die Übertragung sichergestellt werden. Davon ausgehend, dass eine Infrastrukur für die Chatkommunikation existiert, können die Referenzen geeignet in der Chatnachricht kodiert und damit auch übertragen werden. So war es zum Beispiel unter Verwendung der frei verfügbaren Bibliothek JClaim[12] möglich, einen einfachen *ConcertChat*-Client mit expliziter Referenzierung über die Infrastrukturen von Google Talk wie auch AIM zu betreiben.

In *ConcertChat* stehen die Basisfunktionalitäten zur Referenzierung als wiederverwendbare Module bereit. Damit ist der Aufwand für die Integration der Referenzierung in neuen Arten von Arbeitsmaterialien gering. Beispielsweise sind nur sechs Klassen mit ca. 360 Codezeilen notwendig, um unter Nutzung der JPedal-Bibliothek[13] eine referenzierbare Komponente zur Anzeige von PDF-Dokumenten zu erstellen: drei Klassen für die Modellierung und drei für die Erzeugung und Anzeige der Referenzen.

Der Aufwand zur Herstellung der **Konsistenz der Aktionsabfolgen** hängt stark von der genutzten Infrastruktur für den Nachrichtenaustausch ab. Die aus einer serverseitig garantierten Ordnung notwendigerweise resultierenden Latenzen scheinen jedoch im Allgemeinen akzeptabel zu sein.

Mit der **Aktivitätsdokumentation** wird die Chatanzeige zur Darstellung von Nutzeraktionen in anderen Interaktionsbereichen genutzt. Diese Maßnahme verlangt keine Änderung am gemeinsamen Datenmodell. Änderungen sind innerhalb des Chatwerkzeugs notwendig, um die verschiedenen Aktionsbeschreibungen darstellen zu können. In *ConcertChat* ist dies über ein generisches *Renderer*-Konzept realisiert. Das Standard-Chatwerkzeug erlaubt es, applikationsspezifische *Renderer* für die jeweiligen Aktionsbeschreibungen zu registrieren, die dann zu einer Aktionsbeschreibung die entsprechende Darstellung erzeugen. Im Falle vom *WhiteboardChat* werden die Aktionsbeschreibungen als farbige Rechtecke angezeigt. Derselbe Mechanismus erlaubte es jedoch auch, in einem anderen Werkzeug – wie z. B. dem *MeetingChat* – eine textuelle Beschreibung analog zu einer Chatnachricht zu erzeugen. Somit reduziert sich der Aufwand bei der Entwicklung einer Anwendung auf die Bereitstellung eines entsprechenden *Renderers*.

Da *ConcertChat* Gebrauch macht von einem global genutzten Datenmodell für die *Awareness*-Informationen, muss bei der Anwendungsentwicklung für die **inte-**

[12] Siehe `http://sourceforge.net/projects/jclaim/`.
[13] Siehe `http://www.jpedal.org/`.

grierte Awarenessanzeige allein die Anzeige-Komponente angepasst werden. Da die Darstellung jedoch üblicherweise in Textform erfolgt, ist der Aufwand hierbei minimal.

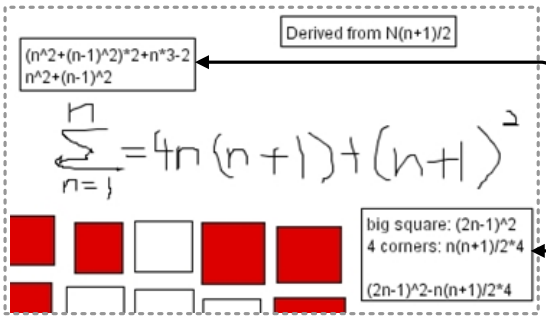
Wie bereits in Kapitel 5.2 beschrieben, erfolgt in *ConcertChat* die Anbindung an eine konkrete *Groupware*-Infrastruktur über die Instanziierung einer passenden `IClientConnection`-Implementierung. So wurde für *ConcertChat* eine Anbindung an IBM Sametime[14] entwickelt.[15]

[14] Siehe `http://www.ibm.com/sametime`.

[15] Einen für den Anwendungsentwickler hilfreichen Sonderfall stellt die ebenso mögliche „lokale Implementierung“ dar. Diese simuliert innerhalb der Client-JVM die notwendigen Grundfunktionalitäten wie Verschicken und Empfangen von Nachrichten. Damit kann ein *ConcertChat*-Client während der Softwareentwicklung *stand alone* gestartet und entsprechend leichter getestet werden.

7 Zusammenfassung und Ausblick

In dieser Arbeit wurden Maßnahmen entwickelt, die die Kollaboration in dualen Interaktionsbereichen erleichtern. Sie zielen auf eine Vereinfachung deiktischer Bezüge, die Erleichterung der Koordination und die Hilfe bei der Reflexion. Das kollaborative Lernen in synchronen, verteilten Lernszenarien setzt eine Reihe von Kompetenzen voraus bzw. beinhaltet deren Erwerb: Problemlösefähigkeiten zur Bearbeitung der eigentlichen Aufgabe, soziale Fertigkeiten zum Agieren als Gruppe und Kompetenzen zur Nutzung der medialen Beschränkungen und Möglichkeiten für die erfolgreiche Kommunikation. Die aus dieser Arbeit resultierenden Integrationsmaßnahmen erweitern die medialen Möglichkeiten dualer Interaktionsräume. Jedoch muss auch deren Anwendung als konversationale Ressource erst erlernt werden, wie die Fortsetzung des auf S. 1 begonnenen Wortwechsels zeigt:



Aznx: Suppose their second formula is our third.
Quicksilver: That was taem c's tho
Aznx: No.
Aznx: They didn't do.
Aznx: The nuumber of squares
Quicksilver: ohj!
Aznx: or the find the big square
Quicksilver: that formula
Quicksilver: i thot u meant the other one
Quicksilver: yeah that is ours
bwang8: point formula out with the tools so we don't get confused
Quicksilver: this is ours
Aznx: That is theirs.

Erst bwang8s Aufforderung „*point formula out with the tools so we don't get confused*" veranlasst Aznx und Quicksilver, durch explizite Referenzen Klarheit in ihre Bezüge zu bringen.

7.1 Zusammenfassung der Arbeit

Die zentrale Frage dieser Arbeit lautet, mit welchen Funktionalitäten duale Interaktionsräume – bestehend aus einem textuellen Kommunikations- und einem diskreten Konstruktionswerkzeug – für die örtlich verteilte synchrone Kollaboration angereichert werden müssen, damit sie als *ein* integriertes Medium besser nutzbar sind.

Für die Beantwortung dieser Frage wurden in der Problemanalyse (Kapitel 2) der Diskurs, die gemeinsame Konstruktion und die Reflexion als Basiselemente kollaborativen Lernens identifiziert. Aufbauend auf den Erkenntnissen zur textuellen Kommunikation und zur computervermittelten Ko-Konstruktion (und den daraus resultierenden Minimalanforderungen an Chat- bzw. Konstruktionswerkzeuge) wurden die Anforderungen an die Integration dualer Interaktionsräume abgeleitet. Diese beziehen sich auf drei Problemfelder:

Deiktische Bezugnahme: Wie kann die im direkten Gespräch mögliche gestische Bezugnahme auf Objekte in der gemeinsamen Umgebung in der computervermittelten textuellen Kommunikation ermöglicht werden?

Koordination der Aktivitäten: Wie kann den Teilnehmern die Koordination ihrer als *disembodied acts* (Zemel et al., 2007) wahrnehmbaren Aktionen, die sich über beide Interaktionsbereiche erstrecken, erleichtert werden?

Reflexion: Wie kann die Reflexion über den Kollaborationsverlauf, der sich aus dem Wechselspiel von Diskurs und der gemeinsamen Konstruktion ergibt, unterstützt werden?

Diese Probleme dualer Interaktionsräume werden bisher nur unzureichend adressiert. Wie die Analyse existierender Systeme und der alternativen Ansätze (Einbettung von Artefakten in den Diskursraum bzw. des Diskurses in den Artefaktraum) in Kapitel 3 zeigte, liegen hierfür bisher keine ausreichenden Lösungen vor.

Die in dieser Arbeit entwickelten Integrationsmaßnahmen (siehe Kap. 4) zielen auf die Unterstützung der Teilnehmer bezüglich der drei genannten Problemfelder:

Explizite Referenzierung: Die explizite Referenzierung erweitert die textuelle Kommunikation um graphische Ausdrucksmöglichkeiten, mit denen die Bezüge eines Chatbeitrages auf Objekte im gemeinsamen Arbeitsbereich oder auf andere Chatbeiträge explizit angegeben und visuell hervorgehoben werden können.

Integrierte *Awareness*-Anzeigen und Aktivitätsdokumentation: Zur Erleichterung der Koordination werden *Awareness*-Informationen über die Interaktionsbereiche hinweg verfügbar gemacht. Dies erlaubt einerseits die Anzeige von Informationen über das aktuelle Tun der anderen am Ort der ei-

genen Aktivität. Andererseits ermöglicht es die persistente Dokumentation der Konstruktionsaktivitäten im Chattranskript und begegnet damit der „Inkompatibilität“ flüchtiger (d. h. zeitlich befristet angezeigter) *Awareness*-Information mit dem Kommunikationsverhalten im quasi-synchronen Medium Chat.

Vollständige Kollaborationshistorie mit Kontextrekonstruktion: Zur Unterstützung der Reflexion über den Kollaborationsverlauf wird die vollständige Kollaborationshistorie verfügbar gemacht. Die Exploration der Kollaborationshistorie wird über die Mechanismen zur Rekonstruktion des Artefaktzustandes zu einem Diskursabschnitt bzw. des Diskurskontextes zu einem Artefaktzustand erleichtert.

Die Umsetzbarkeit der softwaretechnischen Konzeption der Integrationsmaßnahmen wurde anhand des *ConcertChat*-Systems gezeigt (siehe Kap. 5). Dessen Einsatz in realen Kollaborationssituationen zeigte, dass Kleingruppen von den Maßnahmen kreativ und nutzbringend Gebrauch machen (siehe Kap. 6).

7.2 Vergleich mit den Anforderungen

Wie wurden die in Kapitel 2 abgeleiteten Anforderungen an die Integration dualer Integrationsräume (siehe auch Tab. 2.2, S. 45) durch die Integrationsmaßnahmen erfüllt?

Im Folgenden wird überblicksartig die jeweilige Lösung erläutert:

I1, I2: Ermöglichung expliziter Bezugnahmen auf Objekte und Ausschnitte im Arbeitsbereich

Eine Referenz ist modelliert als eine Relation zwischen einem referenzierenden Chatbeitrag und dem arbeitsbereichspezifischen Referenzziel. Sie wird zusammen mit dem Chatbeitrag übertragen. Nach Empfang wird das kodierte Referenzziel im lokalen Datenbestand aufgelöst und damit die Referenz in das lokale Datenmodell integriert. Darauf basierend kann eine Referenz bei Bedarf entsprechend der jeweiligen Positionen von Chatbeitrag und Referenzziel in der Benutzungsschnittstelle graphisch dargestellt werden.

I3: Erfassung und Verfügbarmachen des referenzierten Artefaktzustands

In der Beschreibung des Referenzziels wird ein Verweis (z. B. eine fortlaufende Nummer) auf die entsprechende Version des Arbeitsbereichs gespeichert. Mit diesem kann die Version in der Artefakthistorie identifiziert und bei Bedarf rekonstruiert werden. In *ConcertChat* wird weiterführend das zeitliche Verhältnis der referenzierten zur dargestellten Version durch ein Zeitskala (siehe Abb. 5.6, S. 130) dargestellt.

I4: Nachlesbarkeit der Artefaktaktionen im Chattranskript

Jede im Konstruktionswerkzeug eintreffende Aktion wird an das Chatwerkzeug weitergereicht und dort entsprechend der zeitlichen Ordnung im Chattranskript eingefügt und dargestellt. Dies trifft auch für Aktionen zu, die vor dem Betreten des Kollaborationsraum getätigt wurden und aus der Kollaborationshistorie geladen werden.

I5: Konsistente Aktionsabfolgen für alle Teilnehmer

Die Anforderung, alle Aktionen (also alle Chatbeiträge und Aktionen im gemeinsamen Arbeitsbereich) für alle Teilnehmer in derselben Reihenfolge darzustellen, entspricht der Forderung nach einer totalen Ordnung der die Aktionen tragenden Nachrichten. Die totale Nachrichtenordnung kann über den zugrundeliegenden Nachrichtenkanal sichergestellt werden.

I6: Visuell integrierte Anzeige der *Awareness*-Informationen

Die visuell integrierte Anzeige von *Awareness*-Informationen beider Interaktionsbereiche wird über ein bereichsübergreifendes Datenmodell realisiert. Hierfür werden die *Awareness*-Informationen aus den bereichsspezifischen Modellen gelesen und in dem gemeinsamen gebündelt.

I7: Bereitstellen einer kombinierten Kollaborationshistorie

Durch die totale Ordnung aller Nachrichten ist es möglich, die Chatbeiträge und Aktionen im gemeinsamen Arbeitsbereich global zu indizieren. Hiermit können die gegebenenfalls separat gespeicherten Historien des Chats und des Arbeitsbereichs jederzeit wieder gemischt und die bereichsübergreifende Abfolge rekonstruiert werden.

I8: Rekonstruktion des Artefaktkontextes zu jedem Chatbeitrag

Der globale Index erlaubt es auch, zu jedem Chatbeitrag den entsprechenden Artefaktkontext zu bestimmen. Dies ist derjenige Artefaktzustand, in dem sich der Arbeitsbereich beim Empfang der Chatnachricht befand.

I9: Identifikation des Chatdiskurses zu einem Artefaktzustand

Gleichermaßen erlaubt die Indizierung auch, den zu einem Artefaktzustand geführten Chatdiskurs zu identifizieren. So werden in *ConcertChat* bei Anforderung durch den Nutzer diejenigen Chatbeiträge visuell hervorgehoben, die unmittelbar vor und nach Erzeugung des Artefaktzustands verschickt wurden.

7.3 Wesentliche Beiträge zum Stand der Technik

In dieser Arbeit wurden Maßnahmen zur Integration dualer Interaktionsräume entwickelt, deren softwaretechnische Umsetzung konzipiert, in einem konkreten System implementiert und ihre Wirksamkeit in der Kollaboration von Kleingruppen überprüft. Bisherige Systeme, welche die textuelle Kommunikation mit gemeinsam manipulierbaren Artefakten in dualen Interaktionsräumen kombinieren, stellen das Chat- und das Konstruktionswerkzeug unverbunden nebeneinander. Es fehlt an einer Unterstützung der bereichsübergreifenden Interaktion.

Der Stand der Technik wurde in dieser Arbeit deshalb in mehrfacher Hinsicht weiterentwickelt:

Problemanalyse: Diese Arbeit trägt erstmals umfassend die Probleme bei der Kollaboration in dualen Interaktionsräumen zusammen und analysiert diese aus der Perspektive der *Interaktions*unterstützung. Hierbei wurden drei Problemfelder identifiziert, die bei der Integration dualer Interaktionsräume adressiert werden müssen: die Deixis, die Koordination und die Reflexion.

Anforderungen an Integrationsmaßnahmen: Für die genannten drei Problemfelder wurden neun Anforderungen an Maßnahmen abgeleitet, mit denen über eine technische Integration dualer Interaktionsräume die Nutzer bei der Interaktion unterstützt werden können. Für die Ableitung der Anforderungen wurden Theorien und empirische Erkenntnisse zur computervermittelten Kommunikation und Ko-Konstruktion herangezogen.

Konzeption von Integrationsmaßnahmen: Es wurden in dieser Arbeit sechs Integrationsmaßnahmen entwickelt und softwaretechnisch konzipiert: die Sicherstellung konsistenter Aktionsabfolgen, die Kombination der bereichsspezifischen Interaktionshistorien, deren Nutzung zur Kontextrekonstruktion, die explizite Referenzierung, die Aktivitätsdokumentation sowie integrierte *Awareness*-Anzeigen. Die softwaretechnische Konzeption verdeutlicht, wie existierende duale Interaktionsräume um diese Integrationsmaßnahmen angereichert werden können.

Diese Arbeit zeigt auf, durch welche softwaretechnischen Maßnahmen die Interaktion in reichhaltigen Kollaborationsanwendungen für das synchrone, örtlich verteilte kollaborative Lernen unterstützt werden können. Die Konzeption erlaubt es, diese Maßnahmen in verschiedenartigsten dualen Interaktionsräumen, die einen textuellen Kommunikationskanal mit einem aufgabenspezifischen Konstruktionswerkzeug verbinden, umzusetzen.

7.4 Offene Forschungsfragen

Die im *ConcertChat*-System umgesetzten integrierten Interaktionsräume eröffnen neue Möglichkeiten zur Erforschung wichtiger Fragestellungen im Bereich CSCL:

Wie erfolgt in der Kollaboration die gemeinsame Wissenskonstruktion? Wie wird in der textuellen Kommunikation der *common ground* aufrechterhalten und erweitert? Wie wird von den Teilnehmern indexikalische Symmetrie hergestellt? Wie verändern sich die Kommunikationsmechanismen aufgrund der erweiterten medialen Möglichkeiten (Referenzierung, Aktivitätsdokumentation, Kontextrekonstruktion)?

Momentan wird *ConcertChat* von drei verschiedenen Forschergruppen genutzt, wobei die Funktion zur expliziten Referenzierung besondere Aufmerksamkeit erfährt: Die Gruppe um Hans-Rüdiger Pfister an der Universität Lüneburg untersucht die Nutzung der Referenzierung im Kontext von Lernprotokollen (z. B. Oehl & Pfister, 2007), die Gruppe um Pierre Dillenbourg an der Ecole Polytechnique Fédérale de Lausanne, Schweiz, untersucht den Nutzen bei der Bearbeitung geographischer Problemstellungen mithilfe von Stadtplänen (z. B. Cherubini & Dillenbourg, 2007) und die Gruppe um Gerry Stahl an der Drexel University Philadelphia, USA, setzt den auf dem *ConcertChat*-System basierenden VMTChat ein und untersucht umfassend das kollaborative Lernen in Kleingruppen bei der Bearbeitung offener mathematischer Problemstellungen (z. B. Cakir et al., 2007; Sarmiento & Stahl, 2007b; Stahl, 2007).

Neben der neuen Möglichkeiten zur Erforschung grundlegender Fragestellungen der CSCL-Forschung wirft diese Arbeit jedoch eine Reihe von Fragen bezüglich der Generalisierbarkeit des vorgestellten Integrationskonzeptes auf. Die Konzeption konzentrierte sich

1. auf **duale** Interaktionsräume bestehend aus
2. einem **textuellen, quasi-synchronen Kommunikationsmedium** und
3. einem **diskreten Medium** als Konstruktionswerkzeug
4. sowie auf die Unterstützung **einzelner** und **kleiner** Gruppen bei der
5. **synchronen** und **örtlich verteilten** Bearbeitung
6. **einzelner** Lernaufgaben.

Diese Einschränkungen beziehen sich auf die medialen Eigenschaften (Punkte 1 bis 3) sowie die Organisation des kollaborativen Lernens bzw. im engeren Sinne die verwendete Raummetapher (Punkte 4 bis 6). Inwieweit lässt sich nun das entwickelte Konzept auf Interaktionsräume mit anderen medialen Eigenschaften bzw. auf Lernszenarien, die jenseits der strengen Raumbindung organisiert sind, übertragen? Im Einzelnen heißt dies:

Von dualen zu multiplen Interaktionsräumen: Kann die Konzeption auf Interaktionsräume mit mehr als zwei Interaktionsbereichen übertragen werden? Aus technischer Perspektive steht dem nichts im Wege, die Verallgemeinerung auf mehrere Konstruktionswerkzeuge ist im Rahmen der vorgestellten

softwaretechnischen Konzeption leicht möglich. Im Prinzip steht zu Erwarten, dass die in Kapitel 2.2 diskutierten Probleme nochmals vergrößert werden und damit die Integrationsmaßnahmen für die Interaktion eine größere Bededeutung erhalten. Es droht die Gefahr des „*lost in interaction space*“: Wer ist in welchem Interaktionsbereich aktiv, wie bezieht sich diese Aktivität auf den Kollaborationsinhalt, wie können die Aktionen in den verschiedenen – aufgrund des beschränkten Bildschirmplatzes meist nicht sichtbaren – Interaktionsbereichen koordiniert werden? Möglicherweise bedarf es hier weitergehender technisch unterstützter bereichsübergreifender Koordinationsmechanismen.

Von diskreten zu kontinuierlichen Medien: Die technische Realisierung der Integrationsmaßnahmen basiert auf den Eigenschaften diskreter Medien, indem jegliche Änderungen an den Zuständen auf elementare Aktionen bzw. Nachrichten abgebildet werden können. Hiermit können die einzelnen Aktionen sehr einfach in eine globale Ordnung gebracht werden. Ein strenger zeitlicher Abgleich zwischen den jeweiligen Client-Rechnern ist hierbei nicht notwendig. Wie die Integrationsmaßnahmen für kontinuierliche Medien realisiert werden können, muss durch zukünftige Forschung geklärt werden. Jenseits der *technischen* Realisierung stellen sich jedoch auch prinzipielle Fragen der Nutzungskonzeptes: Wie kann z. B. die Videoübertragung eines physikalischen Experimentes durch eine Gruppe im textuellen, quasi-synchronen Kommunikationsmedium diskutiert werden? Welche Formen zur deiktischen Bezugnahme sind hierbei notwendig?

Von einzelnen zu vernetzten Kollaborationen: Die Konzeption geht von der Raummetapher mit im Prinzip festen Kleingruppen und isolierten Kollaborationen aus. Welcher Interaktionsbedarf besteht jedoch, wenn mehrere Gruppen in verschiedenen Räumen an verschiedenen Aufgaben kollaborativ arbeiten und Möglichkeiten zum Austausch zwischen ihnen geschaffen werden sollen? Wie können die Interaktionsinhalte anderer Räume aufgegriffen werden (Sarmiento & Stahl, 2007a)? Können die integrierten und durch die expliziten Referenzen angereicherten Interaktionshistorien nutzbringend für andere Gruppen aufbereitet werden?

Von „öffentlicher“ zu privater Kommunikation: Eine Annahme der Raummetapher ist, dass alle Aktionen im Kollaborationsraum für alle Anwesenden sichtbar sind. Die Interaktionen erfolgen immer in der Raum„öffentlichkeit“. Jedoch gibt es Hinweise aus der Forschung, dass die Möglichkeit zur privaten Kommunikation zwischen einzelnen Teilnehmern innerhalb eines Raumes wichtig ist z. B. für verschiedene soziale Aspekte räumlich verteilten Lernens (Bregman & Haythornthwaite, 2001) oder auch als *Backchannel* (Cogdill et al., 2001). Wie kann dies aber in integrierten dualen Interaktionsräumen ermöglicht werden? Ist der private Austausch nur im Kommunikationsmedium (Chat) oder auch im gemeinsamen Arbeitsbereich möglich? Wie kön-

nen die sich durch die private Kommunikation aufspannenden unterschiedlichen Diskurs- und Artefaktkontexte (z. B. die private Diskussion einer Formel des gemeinsamen Arbeitsbereichs) organisiert werden? Wie kann hierbei visuell und funktional die unterschiedliche Sichtbarkeit dieser Beiträge/Aktionen berücksichtigt werden?

Jenseits dieser Fragen im Kontext kollaborativen *Lernens* gilt es zu prüfen, ob die Konzeption integrierter dualer Interaktionsräume auf andere Anwendungsszenarien, wie z. B. des kollaborativen Arbeitens oder der Freizeitchats, nutzbringend übertragen werden können.

Literaturverzeichnis

Alterman, R., Feinman, A., Landsman, S. & Introne, J. (2001). Coordination of Talk: Coordination of Action. Technical Report CS-01-217, Department of Computer Science, Brandeis University.

Attardo, D., Barrow, J., Brooks, R., Cummins, J., Godby, P., Kantor, K., Martens, J., Smiles, E. & Womack, T. (2007). Extending Sametime 7.5. Building Plug-ins for Sametime. IBM Red Book, IBM, Poughkeepsie, NY.

Avgeriou, P. & Tandler, P. (2006). Architectural patterns for collaborative applications. *International Journal of Computer Applications in Technology (IJCAT)*, 25(2/3):86–101.

Baker, M. J. & Lund, K. (1997). Promoting reflective interactions in a computer-supported collaborative learning environment. *Journal of Computer Assisted Learning*, 13:175–193.

Balzer, E. & Mühlpfordt, M. (2004). Die ALBA Suite. In S. Münzer & U. Linder (Hg.), *Gemeinsam online lernen. Vom Design bis zur Evaluation kooperativer Online-Übungen*, (S. 74–90). Bielefeld: W. Bertelsmann Verlag.

Barab, S. A., Evans, M. A. & Baek, E.-O. (2004). Activity Theory as a Lens for Characterizing the Participatory Unit. In D. H. Jonassen (Hg.), *Handbook of Research for Educational Communications and Technology*, (S. 199–214). Mahwah, NJ: Lawrence Erlbaum, 2. Aufl.

Beißwenger, M. (2002). Getippte „Gespräche“ und ihre trägermediale Bedingtheit. Zum Einfluß technischer und prozeduraler Faktoren auf die kommunikative Grundhaltung beim Chatten. In I. W. Schröder & S. Voell (Hg.), *Moderne Oralität*. Marburg: Reihe Curupira.

Beißwenger, M. (2003). Sprachhandlungskoordination im Chat. *Zeitschrift für germanistische Linguistik*, 31(2):198–231.

Bekker, M. M., Olson, J. S. & Olson, G. M. (1995). Analysis of gestures in face-to-face design teams provides guidance for how to use groupware in design. In

DIS '95: Proceedings of the conference on Designing interactive systems, (S. 157–166). New York, NY, USA: ACM Press.

Bregman, A. & Haythornthwaite, C. (2001). Radicals of Presentation in Persistent Conversation. In *34th Annual Hawai'i International Conference on System Sciences (HICSS-34)*, Bd. 4. Washington, DC, USA: IEEE Computer Society.

Brinck, T. & Gomez, L. M. (1992). A collaborative medium for the support of conversational props. In *CSCW '92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, (S. 171–178). New York, NY, USA: ACM Press.

Bromme, R., Jucks, R. & Rambow, R. (2004). Experten-Laien-Kommunikation im Wissensmanagement. In G. Reinmann & H. Mandl (Hg.), *Der Mensch im Wissensmanagement: Psychologische Konzepte zum besseren Verständnis und Umgang mit Wissen*, (S. 176–188). Göttingen: Hogrefe.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. (1996). *Pattern-oriented software architecture: a system of patterns*. New York, NY, USA: John Wiley & Sons, Inc.

Cakir, M. P., Zemel, A. & Stahl, G. (2007). The organization of collaborative Math problem solving activities across dual interaction spaces. In C. Chinn, G. Erkens & S. Puntambekar (Hg.), *CSCL 2007: Mice, Minds, and Society*. Verfügbar über: `http://www.cis.drexel.edu/faculty/gerry/vmt/cakir.doc` [Zugriff am: 18.10.2007].

Chabert, A., Grossman, E., Jackson, L. S., Pietrowiz, S. R. & Seguin, C. (1998). Java object-sharing in Habanero. *Commun. ACM*, 41(6):69–76.

Cherubini, M. & Dillenbourg, P. (2007). The effects of explicit referencing in distance problem solving over shared maps. In *GROUP '07: Proceedings of the 2007 international ACM conference on Supporting group work*, (S. 331–340). New York, NY, USA: ACM.

Ching, C. C. (1999). It's not just programming: reflection and the nature of experience in learning through design. In *CSCL '99: Proceedings of the 1999 conference on Computer support for collaborative learning*, (S. 11). International Society of the Learning Sciences.

Chiu, D. (2004). Web-Based Mathematics Education with MathChat. In *International Conference on Information Technology: Coding and Computing (ITCC'04), Las Vegas, Nevada, USA*, Bd. 1, (S. 709–717). IEEE Computer Society.

Churchill, E. F., Trevor, J., Bly, S., Nelson, L. & Cubranic, D. (2000). Anchored Conversations: chatting in the context of a document. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, (S. 454–461). New York, NY, USA: ACM Press.

Clark, H. H. (1978). On inferring what is meant. In W. J. M. Levelt & G. B. F. d'Arcais (Hg.), *Studies in the perception of language*, (S. 295–322). New York: Wiley.

Clark, H. H. (1996). *Using Language*. Cambridge: Cambridge University Press.

Clark, H. H. & Brennan, S. E. (1991). Grounding in communication. In L. B. Resnick, J. M. Levine & S. D. Teasley (Hg.), *Perspectives on socially shared cognition*, (S. 127–149). Washington: APA.

Clark, H. H. & Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, 13(2):259–294.

Cogdill, S., Fanderclai, T. L., Kilborn, J. & Williams, M. G. (2001). Backchannel: Whispering in Digital Conversation. In *34th Annual Hawai'i International Conference on System Sciences (HICSS-34)*, Bd. 4. Washington, DC, USA: IEEE Computer Society.

Cohen, E. G. (1994). *Designing Groupwork: Strategies for heterogeneous classroom*. New York: Teachers College Press, 2. Aufl.

Convertino, G., Ganoe, C. H., Schafer, W. A., Yost, B. & Carroll, J. M. (2005). A Multiple View Approach to Support Common Ground in Distributed and Synchronous Geo-Collaboration. In J. C. Roberts (Hg.), *Proceedings of the Coordinated and Multiple Views in Exploratory Visualization (CMV'05)*, (S. 121–132). Washington, DC, USA: IEEE Computer Society.

Cristian, F., Aghali, H., Strong, R. & Dolev, D. (1985). Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement. In *Proc. 15th Int. Symp. on Fault-Tolerant Computing (FTCS-15)*, (S. 200–206). Ann Arbor, MI, USA: IEEE Computer Society Press.

De Jong, T., Ainsworth, S., Dobson, M., Van der Hulst, A., Levonen, J., Reimann, P., Sime, J.-A., van Someren, M. W., Spada, H. & Swaak, J. (1998). Acquiring knowledge in science and mathematics: the use of multiple representations in technology-based learning environments. In M. W. van Someren, P. Reimann, H. P. A. Boshuizen & T. de Jong (Hg.), *Learning with multiple representations*, (S. 9–40). Amsterdam: Pergamon.

Dewan, P. (1999). Architectures for Collaborative Applications. In M. Beaudouin-Lafon (Hg.), *Computer Supported Co-operative Work*, Bd. 7 von *Trends in Software*, (S. 169–193). John Wiley & Sons.

Dillenbourg, P. (1999). What do you mean by 'collaborative learning'? In P. Dillenbourg (Hg.), *Collaborative Learning. Cognitive and Computational Approaches*, Advances in Learning and instruction series, (S. 1–19). Oxford: Pergamon.

Dillenbourg, P. & CSCL SIG of Kaleidoscope (2005). Dual Interaction Spaces. CSCL 2005 Workshop Proposal. In T. Koschmann, D. D. Suthers & T.-W. Chan (Hg.), *CSCL 2005. Community Events*, (S. 23–24). Taipei, Taiwan.

Dillenbourg, P. & Traum, D. (2006). Sharing Solutions: Persistence and Grounding in Multimodal Collaborative Problem Solving. *Journal of the Learning Sciences*, 15(1):121–151.

Ellis, C. A., Gibbs, S. J. & Rein, G. L. (1991). Groupware – Some Issues and Experiences. *Communications of the ACM*, 34(1):39–58.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors. Special Issue: Situation Awareness*, 37(1):32–64.

Erickson, T. (1999). Persistent conversation: An introduction. *JCMC*, 4(4).

Feinman, A. (2006). *From Discourse Analysis to Groupware Design*. Dissertation, Brandeis University, Waltham, MA.

Fischer, F. (2001). Gemeinsame Wissenskonstruktion – Theoretische und methodologische Aspekte. Forschungsbericht 142, LMU München, Lehrstuhl für Empirische Pädagogik und Pädagogische Psychologie, München.

Fischer, F., Bruhn, J., Gräsel, C. & Mandl, H. (2000). Kooperatives Lernen mit Videokonferenzen: Gemeinsame Wissenskonstruktion und individueller Lernerfolg. *Kognitionswissenschaft*, 9(1):5–16.

Fjuk, A. & Ludvigsen, S. (2001). The complexity of distributed collaborative learning: Unit of analysis. In *Proceedings of the First European Conference on Computer-Supported Collaborative Learning*. Maastricht, The Netherlands.

Friedrich, H. F., Hron, A. & Hesse, F. W. (2001). A framework for designing and evaluating virtual seminars. *European Journal of Education*, 36:157–174.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley.

Garcia, A. & Jacobs, J. B. (1998). The Interactional Organization of Computer Mediated Communication in the College Classroom. *Qualitative Sociology*, 21(3):299–317.

Garcia, A. & Jacobs, J. B. (1999). The eyes of the beholder: Understanding the turn-taking system in quasisynchronous computer mediated communication. *Research on Language and Social Interaction*, 32(4):337–367.

Genau, A. & Kramer, A. (1995). Translucent history. In *CHI '95: Conference companion on Human factors in computing systems*, (S. 250–251). New York, NY, USA: ACM Press.

Gergle, D., Kraut, R. E. & Fussell, S. R. (2004a). Action as language in a shared visual space. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, (S. 487–496). New York, NY, USA: ACM Press.

Gergle, D., Millen, D. R., Kraut, R. E. & Fussell, S. R. (2004b). Persistence matters: making the most of chat in tightly-coupled work. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, (S. 431–438). New York, NY, USA: ACM Press.

Granville, K. & Hickey, T. J. (2005). The design, implementation, and application of the grewpEdit tool. In *TAPIA '05: Proceedings of the 2005 conference on Diversity in computing*, (S. 14–16). New York, NY, USA: ACM Press.

Grudin, J. (1994). Computer-Supported Cooperative Work: History and Focus. *Computer*, 27(5):19–26.

Guicking, A., Tandler, P. & Avgeriou, P. (2005). Agilo: A Highly Flexible Groupware Framework. In *Proceedings of the 11th International Workshop on Groupware (CRIWG '05)*, Bd. 3706 von *Lecture Notes in Computer Science*, (S. 49–56). Berlin, Heidelberg: Springer.

Gutwin, C. & Greenberg, S. (2002). A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Computer Supported Cooperative Work (CSCW)*, 11(3-4):411–446.

Haake, J. M., Schümmer, T. & Haake, A. (2003). Supporting Collaborative Exercises for Distance Education. In *36th Annual Hawai'i International Conference on System Sciences (HICSS'03)*, (S. 31). Washington, DC, USA: IEEE Computer Society.

Haake, J. M., Schwabe, G. & Wessner, M. (2004). Einleitung und Begriffe. In J. M. Haake, G. Schwabe & M. Wessner (Hg.), *CSCL-Kompendium. Lehr- und Handbuch zum computerunterstützten kooperativen Lernen*, (S. 1–4). Müchen, Wien: Oldenbourg.

Hamman, R. (1997). History of the Internet, WWW, IRC, and MUDs. Verfügbar über: `http://www.socio.demon.co.uk/history.html` [Zugriff: 27.02.2006].

Harnoncourt, M., Holzhauser, A., Seethaler, U. & Meinl, P. (2005). Referenzierbarkeit als Schlüssel zum effizienten Chat. In M. Beißwenger & A. Storrer (Hg.), *Chat-Kommunikation in Beruf, Bildung und Medien: Konzepte – Werkzeuge – Anwendungsfelder*, (S. 161–179). Stuttgart: ibidem-Verlag.

Herring, S. C. (1999). Interactional coherence in CMC. In *32th Annual Hawai'i International Conference on System Sciences (HICSS-32)*. Washington, DC, USA: IEEE Computer Society.

Hinsz, V. B., Tindale, R. S. & Vollrath, D. A. (1997). The emerging conceptualization of groups as information processors. *Psychological Bulletin*, 121(1):43–64.

Holmer, T. & Wessner, M. (2005). Gestaltung von Chat-Werkzeugen zur Verringerung der Inkohärenz. In M. Beißwenger & A. Storrer (Hg.), *Chat-Kommunikation in Beruf, Bildung und Medien: Konzepte – Werkzeuge – Anwendungsfelder*, (S. 181–199). Stuttgart: ibidem-Verlag.

Hoppe, U., Pinkwart, N., Lingnau, A., Hofmann, D. & Kuhn, M. (2002). Designing and Supporting Collaborative Modelling Activities in the Classroom. In A. Dimitracopoulou (Hg.), *Information and Communication Technologies in Education*, Bd. I. Proceedings of 3rd HICTE, (S. 185–190). Rhodes, Greece: Kastaniotis Editions.

Hutchins, E. (1990). The Technology of Team Navigation. In J. Galegher, R. Kraut & C. Egido (Hg.), *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, (S. 191–220). Hillsdale, NJ: Lawrence Erlbaum.

IJsselsteijn, W. & Riva, G. (2003). Being There: The experience of presence in mediated environments. In G. Riva, F. Davide & W. A. IJsselsteijn (Hg.), *Being There: Concepts, effects and measurement of user presence in synthetic environments*, (S. 3–16). Amsterdam: IOS Press.

Isaacs, E., Kamm, C., Schiano, D. J., Walendowski, A. & Whittaker, S. (2002). Characterizing instant messaging from recorded logs. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, (S. 720–721). New York, NY, USA: ACM Press.

Jermann, P. (2002). Task and Interaction Regulation in Controlling a Traffic Simulation. In G. Stahl (Hg.), *Proceedings of CSCL 2002*, (S. 601–602). Längere Version verfügbar über: `http://tecfa.unige.ch/tecfa/publicat/jermann-papers/cscl2002/267-long.PDF` [Zugriff: 17.10.2007].

Jermann, P. (2004). *Computer Support for Interaction Regulation in Collaborative Problem-Solving*. Dissertation, University of Geneva, Geneva.

Jödick, F. (2004). GUI Redesign für den RolePlay-Chat, ein Werkzeug für das computervermittelte Rollenspiel. *i-com, Zeitschrift für interaktive und kooperative Medien*, 33(2):47–49.

Kerres, M. & de Witt, C. (2004). Pragmatismus als theoretische Grundlage zur Konzeption von eLearning. In D. Treichel & H. O. Meyer (Hg.), *Handlungsorientiertes Lernen und eLearning. Grundlagen und Beispiele*, (S. 77–99). München: Oldenbourg Verlag.

Kienle, A. (2003). *Integration von Wissensmanagement und kollaborativem Lernen*, Bd. 1 von *Schriften zu Kooperations- und Mediensystemen*. Lohmar: Josef Eul Verlag.

Koch, M. & Schlichter, J. (2001). Verteilung von Daten und Kommunikation. In G. Schwabe, N. Streitz & R. Unland (Hg.), *CSCW-Kompendium: Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten*, (S. 117–123). Berlin, Heidelberg: Springer.

Koch, P. & Oesterreicher, W. (1994). Schriftlichkeit und Sprache. In H. Günther & O. Ludwig (Hg.), *Schrift und Schriftlichkeit: ein interdisziplinäres Handbuch internationaler Forschung*, Bd. 10 von *Handbücher zur Sprach- und Kommunikationswissenschaft*, (S. 587–604). Berlin: de Gruyter.

Kraut, R. E., Gergle, D. & Fussell, S. R. (2002). The use of visual information in shared visual spaces: informing the development of virtual co-presence. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, (S. 31–40). New York, NY, USA: ACM Press.

Landsman, S. (2006). *A Software Lifecycle for Building Groupware Applications: Building Groupware On THYME*. Phd thesis, Brandeis University, Waltham, MA.

Langton, J. T., Hickey, T. J. & Alterman, R. (2004). Integrating tools and resources: a case study in building educational groupware for collaborative programming. *Journal of Computing Sciences in Colleges*, 19(5):140–153.

Leponiemi, J. (2003). Visualizing Discussion History. *International Journal of Human-Computer Interaction*, 15(1):121–134.

Loch, B. & McDonald, C. (2007). Synchronous Chat and Electronic Ink for Distance Support in Mathematics. *innovate. Journal of online education*, 3(3).

Lukosch, S. (2003). *Transparent and Flexible Data Sharing for Synchronous Groupware*. Dissertation, FernUni Hagen, Hagen.

Mauve, M. (2000). *Distributed Interactive Media*. Dissertation, Universität Mannheim, Berlin.

McCarthy, J. C. & Monk, A. (1994). Channels, conversation, cooperation and relevance: All you wanted to know about communication but were afraid to ask. *Collaborative Computing*, 1:35–60.

Miles, V. C., McCarthy, J. C., Dix, A. J., Harrison, M. D. & Monk, A. F. (1993). Reviewing designs for a synchronous-asynchronous group editing environment. In M. Sharples (Hg.), *Computer Supported Collaborative Writing*, (S. 137–160). London: Springer.

Monk, A. (2003). Common Ground in Electronically Mediated Communication: Clark's Theory of Language Use. In J. M. Carroll (Hg.), *HCI Models, Theories and Frameworks: Toward a Multi-Disciplinary Science*, Kap. 10, (S. 265–290). San Francisco: Morgan Kaufmann.

Mühlpfordt, M. & Wessner, M. (2005). Explicit Referencing in Chat Supports Collaborative Learning. In T. Koschmann, D. D. Suthers & T.-W. Chan (Hg.), *Computer Supported Collaborative Learning 2005. The Next 10 Years!* CSCL, Mahwah, NJ: Lawrence Erlbaum Associates.

Murray, D. E. (2000). Protean Communication: The language of computer-mediated communication. *TESOL Quarterly*, 34(3):397–421.

Nash, C. M. (2005). Cohesion and Reference in English Chatroom Discourse. In *38th Annual Hawai'i International Conference on System Sciences (HICSS'05) - Track 4*, (S. 108.3). Washington, DC, USA: IEEE Computer Society.

Ngwenyama, O. & Lyytinen, K. (1997). Groupware Environments as Action Constitutive Resources: A Social Action Framework for Analysing Groupware Technologies. *Computer Supported Collaborative Work: The Journal of Collaborative Computing*, 6:71–93.

Nova, N. (2005). A Review of How Space Affords Socio-Cognitive Processes during Collaboration. *PsychNology Journal*, 3(2):118–148.

O'Donnell, A. N. & Dansereau, D. F. (1992). Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance. In R. Hertz-Lazarowitz & N. Miller (Hg.), *Interactions in cooperative groups. The theoretical anatomy of group learning*, (S. 120–141). Cambridge: Cambridge University Press.

Oehl, M. & Pfister, H.-R. (2007). Explicit Referencing and Shared Materials as Factors for Successful Chat-Based CSCL with Learning Protocols. In C. Montgomerie & J. Seale (Hg.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2007*, (S. 1897–1901). Chesapeake, VA: AACE.

Oerter, R. & Montada, L. (1987). *Entwicklungspsychologie. Ein Lehrbuch*. Weinheim: Psychologie Verlags Union, 2. Aufl.

Ogura, K., Masuda, T. & Ishizaki, M. (2003). Building a New Internet Chat System for Sharing Timing Information. In *4th SIGdial Workshop on Discourse and Dialogue*. Sapporo, Japan. Verfügbar über: `http://fife.speech.cs.cmu.edu/sigdial2003/proceedings/05_LONG_ogura_1_SIGdial03_final_Ogura.pdf` [Zugriff am: 17.10.2007].

Oikarinen, J. (1993). IRC Historie. Verfügbar über `www.efnet.org/?module=docs&doc=22&type=text` [Zugriff am 17.10.2007].

O'Malley, C. (1995). Designing computer support for collaborative learning. In C. O'Malley (Hg.), *Computer Supported Collaborative Learning*, (S. 283–297). Berlin, Heidelberg: Springer.

Palincsar, A. S. & Brown, A. (1984). Reciprocal teaching of comprehension monitoring activities. *Cognition and Instruction*, 1:117–175.

Pata, K. & Sarapuu, T. (2003). Meta-communicative regulation patterns of expressive modeling on whiteboard tool. In *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2003*, (S. 1126–1129). Phoenix, Arizona, USA: AACE.

Patterson, J. F. (1995). A taxonomy of architectures for synchronous groupware applications. *ACM SIGOIS Bulletin – Special Issue on Workshop Write-Ups and Position Papers from CSCW'94*, 15(3):27–29.

Pea, R. (1994). Seeing what we build together: distributed multimedia learning environments for transformative communications. *The Journal of the Learning Sciences*, 3:285–299.

Pfister, H.-R. (2004). Forschungsmethoden. In J. M. Haake, G. Schwabe & M. Wessner (Hg.), *CSCL-Kompendium. Lehr- und Handbuch zum computerunterstützten kooperativen Lernen*. Müchen, Wien: Oldenbourg.

Pfister, H.-R. & Mühlpfordt, M. (2002). Supporting Discourse in a Synchronous Learning Environment: The Learning Protocol Approach. In G. Stahl (Hg.), *Proceedings of the Conference on Computer Supported Cooperative Learning (CSCL) 2002*, (S. 581–589). Hillsdale: Erlbaum.

Pfister, H.-R., Mühlpfordt, M. & Müller, W. (2003). Lernprotokollunterstütztes Lernen – ein Vergleich zwischen unstrukturiertem und systemkontrolliertem diskursivem Lernen im Netz. *Zeitschrift für Psychologie*, 211(2):98–109.

Pfister, H.-R., Müller, W. & Holmer, T. (2004). Learning and Re-Learning from Net-Based Cooperative Learning Discourse. In P. Kommers & G. Richards (Hg.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2004*, (S. 2720–2724). Chesapeake, VA: AACE.

Phillips, W. G. (1999). Architectures for Synchronous Groupware. Techn. Ber. 1999-425, Queen's University, Kingston, Ontario, Canada.

Pimentel, M. G., Fuks, H. & Lucena, C. J. P. (2003). Co-text Loss in Textual Chat Tools. In *Modeling and Using Context. 4th International and Interdisciplinary Conference CONTEXT 2003*, Lecture Notes in Computer Science, (S. 483–490). Berlin, Heidelberg: Springer.

Pürer, H. (2003). *Publizistik- und Kommunikationswissenschaft. Ein Handbuch*. Konstanz: UVK Verlagsgesellschaft.

Reimann, P. (2007). Time is Precious: Why Process Analysis is Essential for CSCL (and Can Also Help To Bridge Between Experimental and Descriptive Methods). In C. Chinn, G. Erkens & S. Puntambekar (Hg.), *CSCL 2007: Mice, Minds, and Society*.

Reimann, P. & Zumbach, J. (2001). Design, Diskurs und Reflexion als zentrale Elemente virtueller Seminare. In F. Hesse & F. Friedrich (Hg.), *Partizipation und Interaktion im virtuellen Seminar*, (S. 135–163). München: Waxmann.

Reinmann-Rothmeier, G. & Mandl, H. (1999). Teamlüge oder Individualisierungsfalle? Eine Analyse kollaborativen Lernens und deren Bedeutung für die Förderung von Lernprozessen in virtuellen Gruppen. Forschungsbericht 115, LMU München, Institut für Pädagogische Psychologie und Empirische Pädagogik, München.

Renninger, A. & Shumar, W. (2002). Community building with and for teachers: The Math Forum as a resource for teacher professional development. In A. Renninger & W. Shumar (Hg.), *Building virtual communities: Learning and change in cyberspace*, (S. 60–95). New York: Cambridge University Press.

Rivera, K., Cooke, N. J. & Bauhs, J. A. (1996). The effects of emotional icons on remote communication. In *CHI '96: Conference companion on Human factors in computing systems*, (S. 99–100). New York, NY, USA: ACM Press.

Rolf, A. (1992). Sichtwechsel – Informatik als (gezähmte) Gestaltungswissenschaft. In W. Coy, F. Nake, J.-M. Pflüger, A. Rolf, J. Seetzen, D. Siefkes & R. Stransfeld (Hg.), *Sichtweisen der Informatik*, (S. 33–48). Braunschweig, Wiesbaden: Vieweg.

Romiszowski, A. & Mason, R. (2004). Computer-Mediated Communication. In D. H. Jonassen (Hg.), *Handbook of Research in Educational Communications and Technology*, (S. 397–431). Mahwah, NJ: Lawrence Erlbaum Associates, 2. Aufl.

Roseman, M. & Greenberg, S. (1996). Building real-time groupware with GroupKit, a groupware toolkit. *ACM Trans. Comput.-Hum. Interact.*, 3(1):66–106.

Roth, J. (2000). *Entwicklungs- und Laufzeitunterstützung für synchrone Groupware*. Dissertation, FernUniversität Hagen.

Roth, J. & Unger, C. (1998). DreamTeam – A Platform for Synchronous Collaborative Applications. In T. Herrmann & K. Just-Hahn (Hg.), *Groupware und organisatorische Innovation (D-CSCW'98)*, (S. 153–165). Stuttgart, Leipzig: B. G. Teubner.

Roth, J. & Unger, C. (2000). An extensible classification model for distribution architectures of synchronous groupware. In R. Dieng, A. Giboin, L. Karsenty & G. D. Michelis (Hg.), *Designing Cooperative Systems. The Use of Theories and Models – Proceedings of COOP'2000*, Bd. 58 von *Frontiers in Artificial Intelligence and Applications*, (S. 113–127). Amsterdam: IOS Press.

Rubart, J. (2005). *THINK SHARED. Modellbasierte und komponentenorientierte Unterstützung wissensintensiver Kooperationen*. Dissertation, FernUniversität Hagen, Hagen.

Saint-Andre, P. (2004). Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence, IETF proposed standard. RFC 3921, Internet Engineering Task Force.

Salomon, G. (1992). What Does the Design of Effective CSCL Require and How Do We Study Its Effects? *ACM SIGCUE Outlook*, 21(3):62–68.

Sarmiento, J. W. & Stahl, G. (2007a). Bridging and Persistence in Sustained, Collaborative Problem Solving Online. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS '07)*, (S. 78). Washington, DC, USA: IEEE Computer Society.

Sarmiento, J. W. & Stahl, G. (2007b). Group creativity in virtual math teams: interactional mechanisms for referencing, remembering and bridging. In *C&C '07: Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*, (S. 37–44). New York, NY, USA: ACM Press.

Schuckmann, C., Kirchner, L., Schümmer, J. & Haake, J. M. (1996). Designing object-oriented synchronous groupware with COAST. In *CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work*, (S. 30–38). New York, NY, USA: ACM Press.

Schulmeister, R. (1999). Virtuelles Lernen aus didaktischer Sicht. *Zeitschrift für Hochschuldidaktik*, 3:1–27.

Schümmer, T. & Schuckmann, C. (2001). Synchrone Softwarearchitekturen. In G. Schwabe, N. Streitz & R. Unland (Hg.), *CSCW-Kompendium: Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten*, (S. 297–309). Berlin, Heidelberg: Springer.

Slavin, R. (1996). Research on Cooperative Learning and Achievement: What We Know, What We Need To Know. *Contemporary Educational Psychology*, 21(1):43–69.

Slavin, R. E. P. (1997). *Educational Psychology: Theory and Practice*. Needham Heights, MA: Allyn & Bacon, 5. Aufl.

Stahl, G. (2006). Sustaining group cognition in a Math chat environment. *Research and Practice in Technology Enhanced Learning*, 1(2):85–113.

Stahl, G. (2007). Meaning making in CSCL: Conditions and preconditions for cognitive processes by groups. In *CSCL 2007*.

Stahl, G., Koschmann, T. & Suthers, D. (2006a). CSCL: An Historical Perspective. In R. K. Sawyer (Hg.), *Cambridge Handbook of the Learning Sciences*, (S. 409–426). Cambridge, UK: Cambridge University Press.

Stahl, G., Zemel, A., Sarmiento, J., Cakir, M., Weimar, S., Wessner, M. & Mühlpfordt, M. (2006b). Shared Referencing of Mathematical Objects in Online Chat. In *International Conference of the Learning Sciences (ICLS 2006)*. Bloomington, Indiana.

Stefik, M., Bobrow, D. G., Foster, G., Lanning, S. & Tatar, D. (1987a). WYSIWIS revised: early experiences with multiuser interfaces. *ACM Transactions on Office Information Systems*, 5(2):147–167.

Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S. & Suchman, L. (1987b). Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *ACM Communications*, 30(1):32–47.

Suthers, D. D. (1999a). Representational Bias as Guidance for Learning Interactions: A Research Agenda. In S. Lajoie & M. Vivet (Hg.), *Artificial Intelligence in Education. Open Learning Environments: New Computational Technologies to Support Learning, Exploration and Collaboration*, Bd. 50 von *Frontiers in Artificial Intelligence and Applications*. Amsterdam: IOS Press.

Suthers, D. D. (1999b). Representational Support for Collaborative Inquiry. In *32th Annual Hawai'i International Conference on System Sciences (HICSS-32)*. Washington, DC, USA: IEEE Computer Society.

Suthers, D. D., Girardeau, L. & Hundhausen, C. D. (2003). Deictic Roles of External Representations in Face-to-face and Online Collaboration. In B. Wasson, S. Ludvigsen & U. Hoppe (Hg.), *Designing for Change in Networked Learning Environments – Proceedings of CSCL 2003*, (S. 173–182). Dordrecht: Kluwer Academic Publishers.

Tang, C. (2003). *Capturing and Visualizing Histories of Multimedia-based Casual Interactions*. M.sc. thesis, Department of Computer Science, University of Calgary, Calgary, Alberta CANADA.

Tatar, D., Foster, G. & Bobrow, D. (1991). Design for Conversation: lessons from Cognoter. *International Journal of Man-Machine Studies*, 34(2):185–209.

ter Hofte, G. H. (1998). *Working apart together: Foundations for component groupware*. Dissertation, Telematica Instituut, Enschede, Niederlande.

Thalemann, S. (2004). *Die Rolle geteilten Wissens beim netzbasierten kollaborativen Problemlösen*. Dissertation, Albert-Ludwigs-Universität, Freiburg i. Brsg.

Tietze, D. (2001). *A Framework for Developing Component-based Cooperative Applications*. Dissertation, Technische Universität Darmstadt, Darmstadt.

Tran, M. H., Yang, Y. & Raikundalia, G. K. (2005). Supporting awareness in instant messaging: an empirical study and mechanism design. In *OZCHI '05: Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction*, (S. 1–10). Narrabundah, Australia, Australia: Computer-Human Interaction Special Interest Group (CHISIG) of Australia.

van Bruggen, J. M. (2003). *Explorations in graphical argumentation. The use of external representations of argumentation in collaborative problem solving*. Dissertation, Open Universiteit Nederland, Heerlen.

van Bruggen, J. M., Kirschner, P. A. & Jochems, W. (2002). External representation of argumentation in CSCL and the management of cognitive load. *Learning and Instruction*, 12:121–138.

Werry, C. (1997). Linguistic and Interactional Features of Internet Relay Chat. In S. Herrings (Hg.), *Computer-mediated Communication. Linguistic, Social and cross-cultural perspectives*, Pragmatics & Beyond, (S. 47–63). Amsterdam: John Benjamins.

Wertheimer, M. (1923). Untersuchungen zur Lehre von der Gestalt. II. *Psychologische Forschung*, 4(1):301–350.

Wessner, M. (2005). *Kontextuelle Kooperation in virtuellen Lernumgebungen*, Bd. 8 von *Schriften zu Kooperations- und Mediensystemen*. Lohmar: Eul-Verlag.

Wessner, M., Shumar, W., Stahl, G., Sarmiento, J., Mühlpfordt, M. & Weimar, S. (2006). Designing an Online Service for a Math Community. In *Proceedings of International Conference of the Learning Sciences (ICLS)*.

Whittaker, S. (2003). Things to Talk About When Talking About Things. *Human-Computer Interaction*, 18(1 & 2):149–170.

Whittaker, S., Brennan, S. E. & Clark, H. H. (1991). Co-ordinating activity: an analysis of interaction in computer-supported co-operative work. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, (S. 361–367). New York, NY, USA: ACM Press.

Zemel, A., Shumar, W. & Cakir, M. (2007). The disembodied act: Copresence and indexical symmetry in computer-mediated communication. In C. Chinn, G. Erkens & S. Puntambekar (Hg.), *CSCL 2007: Mice, Minds, and Society*. Verfügbar über: `http://www.cis.drexel.edu/faculty/gerry/vmt/zemel.doc` [Zugriff am: 18.10.2007].

Zhang, J. (1997). The nature of external representations in problem solvng. *Cognitive Science*, 21(2):179–217.

Zhang, J. & Norman, D. A. (1994). Representations in distributed cognitive tasks. *Cognitive Science*, 18:87–122.

Zhou, N., Zemel, A. & Stahl, G. (2007). Information as a social achievement: Collaborative information behavior in CSCL. In C. Chinn, G. Erkens & S. Puntambekar (Hg.), *CSCL 2007: Mice, Minds, and Society*. Verfügbar über: `http://www.cis.drexel.edu/faculty/gerry/vmt/zhou.doc` [Zugriff am: 18.10.2007].

Curriculum Vitae

Name: Martin Mühlpfordt
Geboren: 27. 10. 1971 in Bleicherode

BILDUNGSWEG / AKADEMISCHE ABSCHLÜSSE

1990 Abitur an der Spezialschule math.-naturwiss.-technischer Richtung in Erfurt

1992–1999 Studium der Psychologie an der Technischen Universität Berlin
Abschluss: Diplom-Psychologe (Note „Sehr gut“)
Schwerpunkte: Kognitionspsychologie, Pädagogische Psychologie
Diplomarbeit: „Repräsentationsprozesse beim analogen Schließen“

1993–2000 Studium der Informatik an der Technischen Universität Berlin
Abschluss: Diplom-Informatiker (Note „Mit Auszeichnung“)
Schwerpunkte: Künstliche Intelligenz, Theoretische Informatik
Diplomarbeit: „Syntaktische Inferenz Rekursiver Programmschemata“

BERUFSPRAXIS

2000 Wissenschaftlicher Mitarbeiter an der Technischen Universität Berlin, Laboratory for Distributed Artificial Intelligence

2000–2007 Wissenschaftlicher Mitarbeiter am Fraunhofer IPSI, Darmstadt

2007– Senior Software Developer, Lycos Europe, Gütersloh