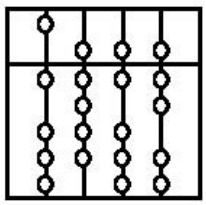
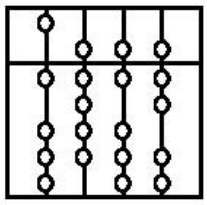


Fakultät für Informatik
der Technischen Universität München

Annotationen in der Lehre
-
Eine Annotationsarchitektur zur Erweiterung bestehender elektronischer Lehrsysteme

Frank Schütz



Angewandte Informatik und Kooperative Systeme
(Lehrstuhl XI)

Technische Universität München

Annotationen in der Lehre
-
Eine Annotationsarchitektur zur Erweiterung bestehender elektronischer Lehrsysteme

Frank Schütz

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Alfons Kemper, Ph.D.

Prüfer der Dissertation:
1. Univ.-Prof. Dr. Johann Schlichter
2. Univ.-Prof. Dr. Peter Hubwieser

Die Dissertation wurde am 1. August 2005 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 10. November 2005 angenommen.

Kurzfassung

Die heutige Gesellschaft erfordert lebenslanges Lernen. Als ein viel versprechender Ansatz dies zu ermöglichen, wird elektronisch unterstützte Lehre gesehen. In den letzten Jahren entstand auf diesem Gebiet eine Unmenge an Systemen. Leider sind einige darunter mehr technisch als didaktisch motiviert.

Die Ergebnisse und Erfahrungen klassischer Lehre sollten bei der Konzipierung neuer Systeme zur elektronischen Unterstützung der Lehre nicht außer Acht gelassen werden. Dazu müssen die Wirkungsweisen erfolgreicher klassischer Konzepte studiert und zumindest verlustfrei - besser noch gewinnbringend - in der elektronischen Lehre adaptiert werden.

In diesem Sinne beschäftigt sich die vorliegende Arbeit mit dem Konzept Annotationen in der Lehre. Nach einer Rechtfertigung für Annotationen als erfolgreichem Konzept der Lehre, wird eine allgemein verwendbare Annotationsarchitektur für die Umsetzung des Annotationskonzepts in die elektronische Lehre vorgestellt. Die Tragfähigkeit dieses Vorschlags wird anhand zweier unterschiedlicher Umsetzungen aufgezeigt.

Schlussfolgernd auf den Ergebnissen der Arbeit wird sich zeigen, dass zum Einen ein solches methodisches Vorgehen über die Betrachtung der klassischen Lehre hin zur Umsetzung in der elektronischen Lehre erfolgreich ist, weil die didaktischen Zusammenhänge verstanden sind und damit vorteilbringend umgesetzt werden können. Zum Anderen liefert diese Arbeit konkret für Annotationen in der Lehre eine Architektur, welche ihrer Flexibilität wegen speziell geeignet ist, bestehende Systeme zur elektronischen Unterstützung der Lehre um umfassende Annotationsfunktionalität zu erweitern.

Danksagung

Diese Arbeit widme ich meiner Frau Susanne und den Söhnen Jakob und Maximilian, welche die Arbeit über ihren Zeitverzicht in erheblichem Maße mitfinanziert haben.

Des Weiteren danke ich meiner Mutter, als guter Seele im Hintergrund und meinem Vater, der mir die Freiheit gibt, meinen Weg zu gehen. Ohne beide wäre ich nicht, wo ich nun bin.

Für die konstruktiven Gespräche zu dieser Arbeit danke ich meinem Doktorvater, Herrn Prof. Dr. Johann Schlichter. Die regelmäßig anschiebenden Worte von Frau Evelyn Gemkow bleiben mir wohl ebenfalls noch einige Zeit in Erinnerung. Insgesamt hat es viel Spaß gemacht, im Umfeld des Lehrstuhls für Angewandte Software und Kooperative Systeme an der Fakultät für Informatik der Technischen Hochschule München zu promovieren.

Zu guter Letzt möchte ich nicht vergessen, die vielen Freunde und Nachbarn zu erwähnen, die mir mit ihrer Zuversicht und positiven Einstellung die nötige Kraft für meine Ideen geben.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

CAL	Call A Lecture, Plattform für die Aufzeichnung und Bereitstellung einer Vorlesungreihe
CBT	Computer Based Training
CORBA	Common Object Request Broker Architecture, Architektur für Verteilte Anwendungen
CSS	Cascading Style Sheets
DOM	Document Object Model, Hierarchie zur eindeutigen Navigation und Bezeichnung von Teilen eines Dokuments
GUID	Globally Unique Identifier, global eindeutiger Schlüssel
HTML	Hyper Text Markup Language, Sprache für Dokumente für das WWW
JNI	Java Native Interface
JSP	Java Server Pages
LAN	Local Area Network, lokales Netzwerk
OCLI	Open Client Lecture Interaction
OLE	Object Linking and Embedding, Microsoft Technologie zur engen Kopplung verschiedener Produkte
OMG	Object Management Group, Gemeinschaft zur Systematisierung objektorientierter Technologien
PDF	Portable Document Format, ein Format, um Dokumente abzulegen
RDF	Resource Description Format
SDK	Software Development Kit
TANGOW	Task-based Adaptive learNer Guidance On the WWW
UCAT	Projekts (Ubiquitous Collaborative Adaptive Training)
UI	User Interface, Benutzungsoberfläche
USB	Universal Serial Bus, Schnittstelle zur seriellen Kommunikation
UUID	Universal Unique Identifier, vgl. GUID
VHB	Virtuelle Hochschule Bayern
VNC	Virtual Network Computing, eine Möglichkeit, die Bildschirmanzeige auf einen anderen Computer zu übertragen
W3C	World Wide Web Consortium, Gremium zur Weiterentwicklung des WWW
WBT	Web Based Training
WLAN	Lokales Funknetzwerk
WWW	World Wide Web, ein Dienst im Internet
XML	Extensible Markup Language

1. Einleitung

1.1. Einordnung

„Du lernst für's Leben." Diese Standardantwort, die Schüler oft auf die Frage „Wozu brauche ich das?" erhalten, muss in heutiger Zeit erweitert werden. Man lernt nicht mehr nur für ein paar Jahre in der Schule und hat dann seine Ruhe. Abgesehen davon, dass es letztendlich nie so war, ist es heutzutage mit der rasend schnellen Entwicklung auf fast allen Gebieten unumgänglich, laufend aktiv zu lernen (vgl. Kommission der Europäischen Gemeinschaft, 1993). So wurde schon 1996 auf Vorschlag der Europäischen Kommission hin zum europäischen Jahr des lebensbegleitenden Lernens erklärt (vgl. Europäisches Parlament und Rat, 1995). Im Jahr 2000 wurde Lebenslanges Lernen in einem Memorandum sogar zu einem Ziel europäischer Politik erhoben (vgl. Kommission der Europäischen Gemeinschaft, 2000).

Für viele Berufe ist schon heute das wichtigste Werkzeug Wissen (vgl. Kuwan & Waschbüsch, 1998, Leonard-Barton, 1995, Nonaka & Takeuchi, 1995, u.a.). Ob in der Verwaltung oder am Band: Ohne das entsprechende „Know-How" kann das geforderte, sehr hohe Arbeitspensum kaum bezwungen werden. Somit ist es auch nicht verwunderlich, dass sich ganze Forschungszweige mit dem Problem Wissen und Information auseinander setzen. *Knowledge Management* und *Information Retrieval* sind nur zwei Schlagworte, die in diesem Zusammenhang fallen.

Zu Beginn konzentrierten sich konkrete Softwareanwendungen zum Wissensmanagement hauptsächlich darauf, Information zu sammeln und bereitzustellen. Dafür wurden in großen Firmen - schon alleine der riesigen Datenmengen wegen - große Softwareprojekte aufgesetzt. Es wurde bald erkannt, dass eine bloße Informationsbereitstellung aber noch kein Wissen vermittelt. Unternehmensweite Lernprogramme wurden ins Leben gerufen. Das Schlagwort „Training on the job" bekam eine neue Bedeutung.

Da heute sehr viele Menschen an einem Computerarbeitsplatz sitzen oder zumindest Zugang zu einem Computer haben, sollten aus Sicht der Vorgesetzten Fortbildungskurse eigenständig am Computer durchgearbeitet werden. Der Begriff *Computer Based Training* (CBT) wurde wieder aufgegriffen (vgl. z.B. frühes System PLATO: Bitzer , Braunfeld & Lichtenberger,1961).

Da die Kursangebote des administrativen Aufwandes wegen nicht auf dem Computer eines jeden einzelnen Schulungsteilnehmers extra eingerichtet werden sollen, werden sie meist zentral über Rechnernetze zur Verfügung gestellt. Mit steigender Verbreitung des World Wide Webs (WWW) wurde offensichtlich, dass für Kurse, die komplett in einer Webbrowserumgebung ausführbar sind, keinerlei weitere Installation auf Seiten der Kursteilnehmer erforderlich war. Zudem bieten in Webbrowsern darstellbare Dokumente weitgehende Plattformunabhängigkeit. Auf Webtechnologie basierende Kursangebote sind somit breit einsetzbar. Sie werden unter dem Begriff *Web based training* (WBT) zusammengefasst.

Alle diese Varianten elektronisch unterstützter Lehre haben zum Einen einen hohen Ausbildungsstand der Mitarbeiter, zum Anderen sinkende Kosten für Weiterbildungsmaßnahmen zum Ziel (vgl. Driscoll & Thomson, 1997).

Eine Frage, die in diesem Zusammenhang heute des Öfteren speziell von Pädagogen aufgeworfen wird, ist: „Ist der Computer überhaupt ein adäquates Mittel für die Lehre?“ Zum Einen mag das an einer gewissen konservativen Einstellung neuen Medien gegenüber liegen. Zum Anderen ist es aber sicher auch eine berechtigte Frage. Viele Lösungen zur elektronischen Unterstützung der Lehre sind technikzentriert. Das Vorgehen in der Entwicklung war oft: „Was bietet die Technik?“ So entstanden Lösungen, die technisch anspruchsvoll waren, mit der pädagogischen Wirklichkeit aber kaum in Einklang zu bringen sind. Solche Lösungen bieten oft wünschenswerte Innovationen in einigen Aspekten, vernachlässigen aber andere, aus pädagogischer Sicht unverzichtbare Dinge (vgl. Argumentation in Maurer & Dietinger, 1997).

„Es kann nicht alles schlecht sein, was wir seit hundert Jahren machen“, sagte ein vortragender Pädagoge auf der ICTE 2002 in Badajoz selbstironisch. In diesem Sinn ergab sich auf dieser Konferenz, die nicht nur von Informatikern und Technikern besucht war, das folgende Leitbild: „Vergessen wir für den Moment das technisch Machbare und setzen nicht Technik um der Technik willen ein. Überdenken wir die erprobten Konzepte und Techniken der Pädagogik und Didaktik auf die Möglichkeiten der *Unterstützung durch* Technik hin und prüfen anschließend, was auf dieser neuen Basis zusätzlich möglich ist. Dann ergibt sich sicher ein Gewinn für alle Seiten“.

Diesem Prinzip folgend ist diese Arbeit entstanden. Konkret werden im Laufe dieser Arbeit *Annotationen* im Lehrumfeld aus verschiedenen Perspektiven betrachtet. Schließlich wird als Ergebnis ein komponentenbasiertes Design zur Umsetzung von Annotationen in elektronisch unterstützter Lehre präsentiert. Erklärtes Ziel ist dabei primär kein weiteres *Komplett*lehrsystem mit Annotationsunterstützung, nach dem Motto „*Yet Another Teaching System with Annotation Support*“ zu entwickeln. Vielmehr soll es anderen Entwicklern - aufbauend auf dem vorgestellten Design - auf einfache Weise möglich sein, eine umfassende Annotationsunterstützung in ihre bestehenden Systeme zur elektronischen Unterstützung der Lehre zu integrieren. Das Design erhebt den Anspruch, aufgrund der intensiven Beschäftigung mit dem Konzept Annotation in der Lehre, ein sicheres Fundament für eine Umsetzung zu bieten. „Sicher“ bedeutet hier, zum Einen weitestgehend die Möglichkeiten der klassischen Lehre zu bieten. Zum Anderen beinhaltet eine Anwendung, die dieses Design zu Grunde legt, schon alle im weiteren Verlauf der Arbeit erkannten zusätzlichen Möglichkeiten einer elektronischen Umsetzung von Annotationen.

Wie kann ein so hoher Anspruch an ein solches Design erfüllt werden?

Zuerst werden dazu Annotationen im klassischen Lehrprozess untersucht. Nachdem die Konzepte, die in der klassischen Lehre mit Hilfe von Annotationen umgesetzt werden, durch Analysieren der Prozesse identifiziert und verstanden sind, wird weiter betrachtet, welche innovativen Erweiterungen bei einer elektronischen Umsetzung zusätzlich möglich sind. Solche innovativen Erweiterungen werden auf dreierlei Wegen gefunden:

- Es werden theoretische Überlegungen darüber angestellt, was aufgrund der elektronischen Verfügbarkeit von Annotationen machbar ist.
- Es werden bestehende elektronische Umsetzungen von Annotationen in der Lehre betrachtet.
- Es werden allgemeine Systeme mit Annotationsunterstützung außerhalb der Lehre betrachtet.

Außerdem wird für die Entwicklung eines zukunftsicheren Designs konsequent auf eine gute Erweiterbarkeit und Anpassbarkeit geachtet.

Zum Beleg der Tragfähigkeit der Architektur werden schließlich zwei konkrete Erweiterungen bestehender elektronischer Systems zur Unterstützung der Lehre zum Einen auf Basis der Architektur, zum Anderen auf Basis der erstellten Referenzimplementierung durchgeführt. Aufgrund der strikten Trennung zwischen Vorstellung der Architektur und dem zur Verfügung stellen einer konkreten Referenzimplementierung ist eine breite Nutzung der Ergebnisse dieser Arbeit möglich.

Nun zur Abgrenzung gegenüber anderen Arbeiten im Umfeld von Annotationen in elektronisch unterstützter Lehre.

1.2. Verwandte Arbeiten

Der im Bereich der elektronischen Lehre beheimatete Leser kennt sicher schon einige Arbeiten, die sich in unterschiedlicher Tiefe mit der Thematik Annotationen in der elektronischen Lehre beschäftigen. Diese Arbeiten betrachten Annotationen meist in einem recht speziellen Kontext, wie beispielsweise Gruppenlernen (vgl. z.B. Iles, Glaser, Kam & Canny, 2002), und Aufzeichnung des Kursgeschehens (vgl. z.B. Bacher, Müller, Ottmann & Will, 1997). Aus dem Bereich Wissensmanagement - speziell Semantic Web - kommen mittlerweile weitere Arbeiten hinzu, die von ihrem Standpunkt aus Annotationen behandeln (vgl. z.B. Yang, Chen & Shao, 2004).

Alle diese Arbeiten haben gemeinsam, dass sie Annotationen als gegeben ansehen und benutzen. Eine tiefergehende Auseinandersetzung mit dem Konzept Annotation und hier speziell Annotation in der Lehre findet kaum statt. Falls doch, handelt es sich meist um Teilaspekte von Annotationen. So rückt im Umfeld des Semantic Web's der maschinenlesbare beziehungsweise -interpretierbare Inhalt der Annotation in den Vordergrund. Handschriftliche Bemerkungen und grafische Veranschaulichungen finden hier kaum Beachtung, obwohl diese - wie gezeigt wird - speziell im Lehrumfeld große Bedeutung haben.

Die vorliegende Arbeit schließt mit seinen ersten Kapiteln zu Annotationen diese Lücke. Durch die intensive Beschäftigung mit dem Konzept Annotation in der Lehre und der dafür notwendigen Aufarbeitung unterschiedlicher Blickwinkel ergibt sich ein tiefergehendes Verständnis für Annotationen. Dies wiederum kommt den oben genannten Arbeiten als geschaffene Grundlage zu Gute.

Im weiteren Verlauf der Arbeit wird dann ein allgemeines Design zur Umsetzung von Annotationen in elektronisch unterstützter Lehre entwickelt. Auch dieser Teil unterscheidet sich von den anderen Arbeiten. Diese implementieren Annotationen meist innerhalb ihrer eigenen Umgebung als Anhängsel zur Vervollständigung ihres eigenen Systems. Im Gegensatz dazu werden in der vorliegenden Arbeit Annotationen in einem eigenen abgeschlossenen Architekturvorschlag implementiert. Dieser Architekturvorschlag ist von Anfang an so konzipiert, dass zum Einen alle vorgestellten Aspekte von Annotationen verwirklicht sind und zum Anderen damit ein bestehendes System ohne großen Aufwand um Annotationen erweitert werden kann. Die Umsetzung von Annotationen ist somit ungleich allgemeingültiger.

1.3. Gliederung der Arbeit

Nach dieser Einleitung in die Thematik wird im folgenden Abschnitt vor dem Einstieg in den inhaltlichen Teil der Arbeit für ein gemeinsames Verständnis der benutzten Begriffe gesorgt. Anschließend beleuchtet Kapitel *2. Annotationen in der Lehre* (S. 11) Annotationen aus der Sicht der Didaktik, Pädagogik und Psychologie. Aus diesem Teil ergeben sich erste Vorgaben für eine elektronische Unterstützung von Annotationen. Mit dem nun gewonnenen Einblick in das Konzept Annotation in der Lehre wird in Kapitel 3. *Annotationen in Prozessen der Lehre* (S. 33) ihr konkreter Einsatz in der elektronischen Lehre thematisiert. Der dabei entstehende Kriterienkatalog wird in Kapitel 4. *Annotationen in bestehenden Anwendungen* (S.47) genutzt, um sich ein Bild über die Annotationsunterstützung in bestehenden Anwendungen zu machen. Mit den Ergebnissen aus den vorherigen Kapiteln kann dann in Kapitel 5. *Architektur zur Integration von Annotationen* (S. 70) fundiert eine Architektur für eine umfassende Annotationsunterstützung erarbeitet werden. In Kapitel 6. *Konkrete Anwendung der Annotationsarchitektur* (S. 133) wird anhand des CAL-System (Call A Lecture) des Lehrstuhls Prof. Schlichter, Informatik, TU München zur Aufzeichnung von Vorlesungen (vgl. Schütz, 2001) und anhand des WOTAN-Systems im UCAT Projekt (vgl. Rodriguez, 2005) der praktische Einsatz der Architektur und der Referenzimplementierung aufgezeigt.
Kapitel 7. *Bewertung* (S. 154) befasst sich zusammenfassend übergreifend mit Annotationen in der elektronischen Lehre. Kapitel 7.3. (S. 156) schließlich zeigt weitere Anwendungsmöglichkeiten der Ergebnisse der vorliegenden Arbeit auf und gibt Impulse, in welche Richtung Untersuchungen zu Annotationen fortgeführt werden können.

Methodisch folgt diese Arbeit also in den Anfangskapiteln einem deduktivem Ansatz. Wo nötig, werden eigene empirische Untersuchungen ergänzend eingesetzt. Das Vorgehen ab Kapitel 5. ist konstruktiv.

Bevor nun aber Annotationen in der klassischen Lehre genauer untersucht werden, wird im folgenden Abschnitt für ein gemeinsames Verständnis der benutzten Begriffe gesorgt.

1.4. Nomenklatur

Diese Arbeit setzt keinen Fokus auf die universitäre Lehre, wenngleich einige der verwendeten Begriffe aus diesem Umfeld stammen. Diese Begriffe werden als Platzhalter für die den anderen Lehrsituationen entsprechenden Begriffe benutzt. Am Ende dieses Abschnitts werden die Begriffe und ihr Zusammenhang in einer Grafik zu einem ersten Modell zusammengefasst. Wo nötig werden einzelne der aufgeführten Begriffe im Laufe der Arbeit noch weiter thematisiert.

– Kurs

Kurs steht für eine Ansammlung von Unterrichtseinheiten, zum Beispiel „Vorlesung Einführung in die Informatik I, WS 2004/05“ aber auch „Mathematik Grundschule 3. Jahrgangsstufe“.
Auch muss es sich bei einem Kurs in diesem Sinne nicht um eine Präsenzveranstaltung handeln. Unter die Begriffsdefinition fallen somit auch Lehrveranstaltungen, die ausschließlich über den Computer oder ähnliche Medien angeboten werden. Aus diesem Grund wurde der Begriff *Vorlesung* vermieden, der das Zugegensein eines Vorlesenden impliziert.
Erkennungsmerkmale, die auf einen Kurs in diesem Sinne hindeuten sind, dass sich Studenten oftmals für einen Kurs einschreiben müssen und dass es eine abschließende Prüfung gibt.

– Unterrichtseinheit

Ein Kurs ist in *Unterrichtseinheiten* aufgeteilt. Eine Unterrichtseinheit im Sinne dieser Arbeit stellt eine abgeschlossene, vom Dozenten als solche vorbereitete Menge Stoff dar. Ein anderer in der Didaktik gebräuchlicher Begriff ist „Feinlernziel“ (vgl. Brokmann-Nooren, Rieken & Schröder, 1995).
Um den Umfang einer Unterrichtseinheit im Sinne dieser Arbeit abschätzen zu können, kann folgende Regel herangezogen werden: Der Studierende sollte eine Unterrichtseinheit nicht über mehrere Sitzungen verteilt vermittelt bekommen. Innerhalb einer Sitzung können aber mehrere Unterrichtseinheiten abgehandelt werden, wobei dies natürlich von der Dauer einer Sitzung abhängt.
Eine Unterrichtseinheit kann real, das heißt im Zugegensein des Dozenten oder virtuell, das heißt alleine vor dem Computer vermittelt werden. Beispiele für Unterrichtseinheiten in den oben genannten Kursen wären also „Rekursion - Ein Grundbaustein funktionalen Programmierens“ oder „Einführung in schriftliches Multiplizieren“.

– Sitzung

Während die Unterrichtseinheit eine logische Gliederung eines Kurses ermöglicht, stehen *Sitzungen* für administrative Gliederungen. Bei einer Präsenzveranstaltung ist eine Sitzung durch das Zusammentreffen des Dozenten mit den Studenten charakterisiert. So stellt zum Beispiel eine Doppelstunde Mathematik am Freitag den 13. Mai 2000 eine Sitzung dar, in der aber durchaus mehrere Unterrichtseinheiten gelehrt werden können.
In einer elektronischen Lehrumgebung stellt der Aufenthalt vom Starten bis zum Beenden der Lehrumgebung eine Sitzung dar.

– Dozent

Der *Dozent* definiert die Lehrinhalte, bereitet die Unterrichtseinheiten für einen Kurs vor und präsentiert den Kurs. Ein Lehrer an Schulen fällt im Sinne dieser Arbeit also auch unter den Begriff *Dozent*.
Der Dozent benutzt als konkrete Repräsentation des Kurses ein Kursskript, welches er nicht gezwungenermaßen an die Studenten weitergibt. Er hält einen Kurs üblicherweise nicht nur einmal und ist deshalb an einer Weiterentwicklung des Kurses und seines Kursskripts interessiert. Der Dozent agiert im Laufe eines Kurszyklus in drei Rollen:

- Verfasser von Unterrichtseinheiten
- Integrator von Unterrichtseinheiten in einen Kurs
- Präsentator eines Kurses

Im Zuge dieser Arbeit wird die Teilung in diese Rollen nicht weiter thematisiert. Wird an einer Stelle konkret ein Rollenaspekt benötigt, wird dies explizit angegeben. Das Präsentieren eines Kurses im elektronischen Umfeld entspricht meist der Nutzbarmachung des Kursskripts und anderer Artefakte, wie zum Beispiel Übungsaufgaben über elektronische Medien.

– Student

Ein *Student* versucht, durch Teilnahme an Sitzungen den Stoff der Unterrichtseinheiten im Kurs zu verinnerlichen. Da dies während der Sitzung selten vollständig möglich ist, hat der Student ein Interesse, die Unterrichtseinheiten für sich subjektiv möglichst effektiv nachar-

beiten zu können. Als Hilfsmittel für verschiedene Techniken, sich den Stoff anzueignen, dient dem Student dabei unter anderem das Annotieren.
Schüler sollen im Rahmen dieser Arbeit ebenfalls unter den Begriff „Student“ fallen.

– Kursskript

In einem *Kursskript* sind die Lehrinhalte eines Kurses repräsentiert. Zu ein und dem selben Kurs kann es beliebig viele Repräsentationen geben. So existieren zu einem Kurs in der Regel folgende beiden Repräsentationen :

- Das Dozentenskript
- Das Studentenskript

Oft gibt es zwei parallele Dozentenskripte: Einmal die Präsentationsunterlagen und einmal ein Konzeptskript mit zusätzlichen Hinweisen. Allgemein fixiert der Dozent im Dozentenskript in der Rolle des Verfassers und des Integrators die Lehrinhalte eines Kurses. Dieses Kursskript stellt für den Dozenten als Präsentator eine Gliederung oder das „Drehbuch“ dar, nach dem er den Kurs abhält.

Zudem kann ein Kursskript auch den Studierenden - oftmals als eigene Variante - zur Verfügung gestellt werden, das sogenannte Studentenskript. Dem Studenten dient ein Kursskript als Gedächtnisstütze für eine spätere Wiederholung des Stoffs.

Wird vom Dozenten kein Studentenskript vorbereitet, erstellt jeder Student durch seine Annotationen während der Sitzung eine private Version eines Kursskripts. Gibt der Dozent ein wenig detailliertes Kursskript heraus, wird dieses während der Vorlesung dynamisch durch Annotationen vom Dozenten oder Studenten ergänzt. Über die Auswirkungen des Aufbaus beziehungsweise Detaillierungsgrad eines Studentenskripts gibt zum Beispiel James Hartley (1976) Auskunft. Untersuchungen zum Ausgabezeitpunkt eines Studentenskripts stellen beispielsweise I. R. McDoughall, H.W. Gray und G.P. McNicol (1972) an.

Konzeptuell ist zu deuten, dass ein Kursskript beliebig viel Freiraum für Annotationen bietet. In der klassischen Lehre wird dies durch bedarfweises Einfügen oder Anhängen von leeren Blättern erreicht.

– Annotation

Als *Annotation* wird im Zuge dieser Arbeit jedes während einer Sitzung oder der Nachbereitung einer Sitzung entstehende Artefakt bezeichnet.

Dabei ist es unerheblich, ob die Annotation vom Dozenten oder einem Studenten stammt, ob sie handschriftlich oder mit Tastatur angefertigt ist, ob sie sich direkt im Kursskript oder etwa auf einem zusätzlichen Zettel befindet und ob sie inhaltlichen Bezug zum Kurs hat. Ausdrücklich betont sei, dass eine Annotation nicht zwangsläufig aus Text besteht. Auch Skizzen, Videos oder Tonaufzeichnungen können Annotationen sein.

– Notiz

Notizen stellen im Sinne dieser Arbeit eine Untermenge der Annotationen dar. Als Notiz wird im Weiteren eine Annotation bezeichnet, die mit einem Stift, einer Tastatur oder anderen Zeichenwerkzeugen erstellt wurde. Im Gegensatz dazu werden Video und Audioannotationen nicht als Notiz bezeichnet.

Wird im Laufe der Arbeit von Notizen gesprochen, sind explizit nur Annotationen die mit einem Stift, der Tastatur oder einem Zeichenwerkzeug erstellt werden, angesprochen.

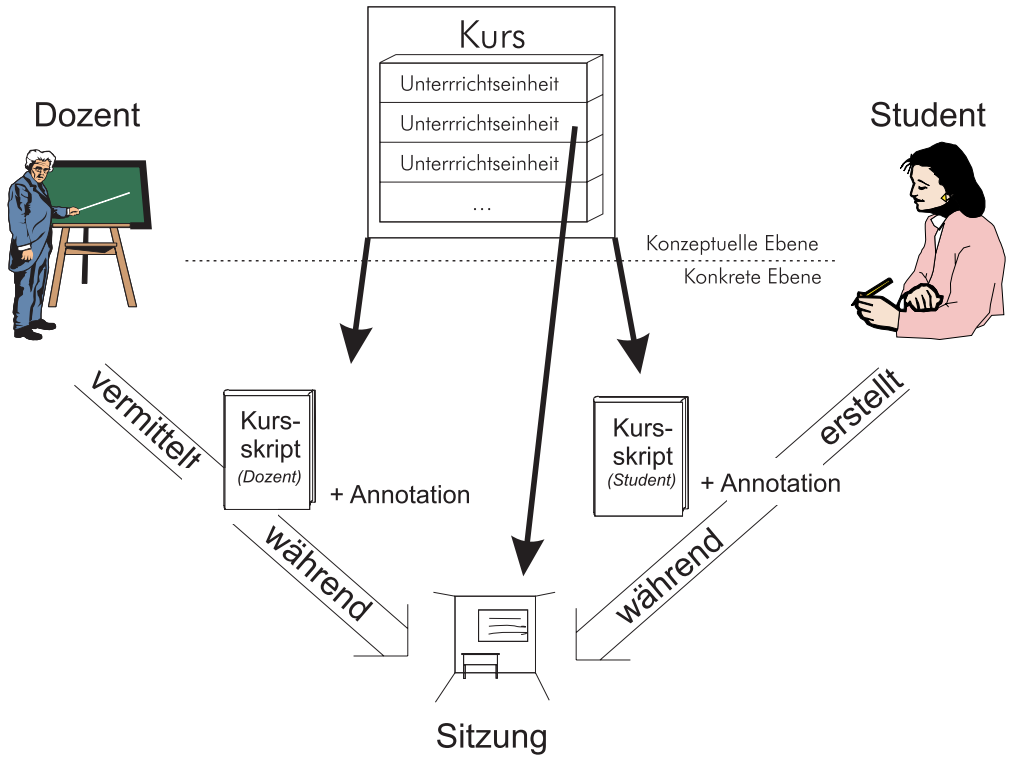


Abb. 1: Zusammenhang zwischen den Begrifflichkeiten der Arbeit

Die Grafik in Abbildung 1 veranschaulicht den Zusammenhang der Begriffe. Zu erkennen ist unter anderem, dass der Dozent eine eigene Version eines Kursskripts besitzt, die er während einer Sitzung vermittelt, wohingegen der Student seine Version eines Kursskripts während der Sitzung erstellt.

Diese Grafik bildet mit der Darstellung der Zusammenhänge auch ein erstes Modell für die weiteren Ausführungen.

Neben den oben genannten Begriffen aus dem Lehrumfeld ist für die Arbeit ein weiterer von zentraler Bedeutung:

– **Elektronisch unterstützte Lehre**

Im Rahmen dieser Arbeit wird unter dem Begriff *elektronisch unterstützte Lehre* jegliche Form von Lehre zusammengefasst, die sich in zentralen Teilen eines elektronischen Vermittlungsmediums bedient. Dabei ist es unerheblich, ob sich an den jeweiligen Enden des Vermittlungsmediums Menschen oder Computer befinden.

– **eLearning**

„E-Learning ist Lernen, das mit Informations- und Kommunikationstechnologien unterstützt bzw. ermöglicht wird. Wichtig ist, dass diese Technologien mit dem Lernprozess selbst un-

mittelbar verbunden sind und nicht nur rudimentäre Hilfsmittel darstellen.“, sagt Seufert (2001, S. 13). Köhne, Ruisz und Krcmar (2002, S. 464) betonen: „... , dass eLearning-Werkzeuge als Hilfsmittel für den Lernenden aufzufassen sind, indem es das Lernen über räumliche und zeitliche Distanz erheblich erleichtert.“

eLearning geht auf das Thema Lehre also konzeptuell von der Studentensicht zu. Die Inhalte werden elektronisch präsentiert. Der Dozent taucht in den unterstützten Szenarien nicht leiblich auf. Der Begriff eTeaching ist wenig gebräuchlich. Unter dem Oberbegriff eLearning versammeln sich Dinge wie Web Based Training (WBT), Computer Based Training (CBT), Distance Learning (DL), Lern Management System (LMS) und etliche andere.

Da Annotationen aber von Studenten und Dozenten genutzt werden und eine getrennte Betrachtung nicht zwingend erforderlich ist, werden in dieser Arbeit der Allgemeinheit wegen beide Seiten betrachtet. Der Begriff eLearning schränkt daher zu sehr ein.

Aus diesem Grunde wird *elektronisch unterstützte Lehre* im Zuge dieser Arbeit als Oberbegriff für alle Formen von Lehre und Lernen über ein elektronisches Vermittlungsmedium benutzt. Dies ist auch damit begründbar, dass all diese Formen eine Annotationsunterstützung bieten sollten, da für alle die Ergebnisse von Kapitel 2 gelten.

Nachdem nun ein gemeinsames Grundverständnis für die benutzten Begriffe geschaffen ist, werden im folgenden Kapitel Annotationen in der Lehre eingehender betrachtet.

2. Annotationen in der Lehre

2.1. Kapitelüberblick

Intention dieses Kapitels ist es, zum Einen allgemein die Notwendigkeit von Annotationen im Unterrichtsumfeld zu belegen. Zum Anderen wird ein Verständnis für das Konzept Annotation in der Lehre aufgebaut. Dieses Verständnis ist für eine spätere sinnvolle Integration von Annotationen in elektronische Systeme zur Unterstützung der Lehre unabdingbar.

Um dem Leser zu Beginn einen gewissen Überblick über Annotationen zu verschaffen, werden in *2.2. Erscheinungsformen* verschiedene Ausprägungen von Annotationen vorgestellt. Dabei werden Erkennungsmerkmale benannt, anhand derer die Kategorisierung in die jeweilige Erscheinungsform durchgeführt wird. Außerdem werden bei jeder Erscheinungsform die für die Erstellung notwendigen Werkzeuge angegeben. Über diese Werkzeuge lassen sich direkt erste Anforderungen an ein System zur Annotationserstellung ableiten. Diese Anforderungen werden am Ende des Abschnitts in einer Tabelle zusammengefasst.

War für die Diskussion der Erscheinungsformen noch keine Zuordnung zu Personen oder Rollen notwendig, ändert sich dies bei der weiteren Betrachtung. Zur Vertiefung des Bezugs zur Lehre wird der Ersteller einer Annotation und seine Beziehungen zu Annotationen in die Betrachtung mit aufgenommen. In erster Linie wird bei Annotationen in der Lehre wohl an den Studierenden, der Notizen in sein Studentenskript macht, gedacht. Diese so genannten Studentenannotationen und ihre Funktion aus lernpsychologischer Sicht werden unter *2.3. Studentenannotation* genauer untersucht. Daraus resultiert unter anderem eine Begründung, wieso Annotationen für Studenten Vorteile im Lernprozess bieten.

Neben den Studierenden darf aber der Dozent als Person, die Annotationen verfasst, nicht vergessen werden. Unter anderem während einer Sitzung ergänzt der Dozent das Kursskript zum Beispiel an einer Tafel oder einem Overheadprojektor. Aus Dozentensicht hat der Pool an Medien zur Erstellung einer Annotation Einfluß auf seinen Unterricht. Dieser Aspekt wird neben der Diskussion über die Funktion einer Annotation aus Dozentensicht unter 2.4. *Dozentenannotation* beleuchtet. Hier wird sich zeigen, dass Dozenten für ansprechende Lehre auf Annotationen angewiesen sind.

2.2. Erscheinungsformen

Unter dem Begriff Erscheinungsform einer Annotation ist eine Kategorisierung auf semantischer Ebene zu verstehen. Im Folgenden werden Erscheinungsformen von Annotationen aufgeführt und gegenseitig in Beziehung gesetzt. Dabei wird auch kurz erläutert, in welchem Kontext die jeweilige Erscheinungsform geeignet ist und wo weniger. Im Weiteren soll jede Annotation in einem Freiraum im ursprünglichen Kursskript als *Randnotiz* bezeichnet werden. Die Annotation behält also in der Regel einen räumlichen Bezug zum annotierten Inhalt im Kursskript, manipuliert das ursprüngliche Kursskript aber kaum direkt. Bis auf die erste im Folgenden angesprochenen Erscheinungsform *Hervorhebung* werden also alle anderen Annotationen als Randnotiz (Marginalie) erstellt. Erinnert sei daran, dass ein Kursskript konzeptuell gesehen beliebig viel Freiraum bietet (siehe Nomenklatur *Kursskript*, S. 8).

Die vorgestellte Liste der Erscheinungsformen entstand motiviert durch Bernd Weidenmanns (1994) Kategorisierung von Bildmedien. Diese werden oft von Dozenten als Annota-

tion eingesetzt. Die vorgestellte Liste basiert auf eigenen Erfahrungen und den bei Studentenannotationen gemachten Beobachtungen. Zudem haben Arbeiten zu Lernstrategien weitere Impulse geliefert (vgl. z. B. Weinstein & Mayer, 1985).

2.2.1. Erscheinungsformen in papiergestützten Umgebungen

Über viele Jahrzehnte waren papiergestützte Systeme der einzig praktikabel erscheinende Weg für gute und nachhaltige Lehre. Es gab kein anderes, bezahlbares Medium zur Sicherung der Ergebnisse. Aus diesem Grund werden nun zuerst Erscheinungsformen von Annotationen in papiergestützten Umgebungen aufgeführt. Im Anschluss daran werden die in einer modernen multimedialen Umgebung hinzugekommenen Erscheinungsformen betrachtet.

2.2.1.1. Hervorhebung

Die Erscheinungsform *Hervorhebung* dient dem Herausstellen einzelner Teile im Kursskript. Praktisch erreicht wird eine Hervorhebung durch Unterstreichen, Highlighten - also farblich Hinterlegen - oder Einrahmen (vgl. Abb. 2).

Zu erwähnen bleibt, dass Hervorhebung nicht nur bei Text angewendet werden kann, sondern ebenso bei Skizzen, Diagrammen oder dergleichen für einen Fokus sorgt.

Diese Erscheinungsform einer Annotation liegt also vor, wenn:

- bestehende Teile eines Kursskripts
- unmittelbar im konkreten Layout
- fokussierend ergänzt

werden.

Eine Hervorhebung hilft, wichtige und weniger wichtige Inhalte optisch zu trennen. Beim Nacharbeiten des Stoffes kann der Studierende sich dann mehr auf die wichtigen Teile konzentrieren. Beim vorteilhaften Einsatz dieser Technik lassen sich Strukturen leichter erkennen und Kernaussagen identifizieren (vgl. *Basic Organisational Strategy* in Weinstein & Mayer, 1985).

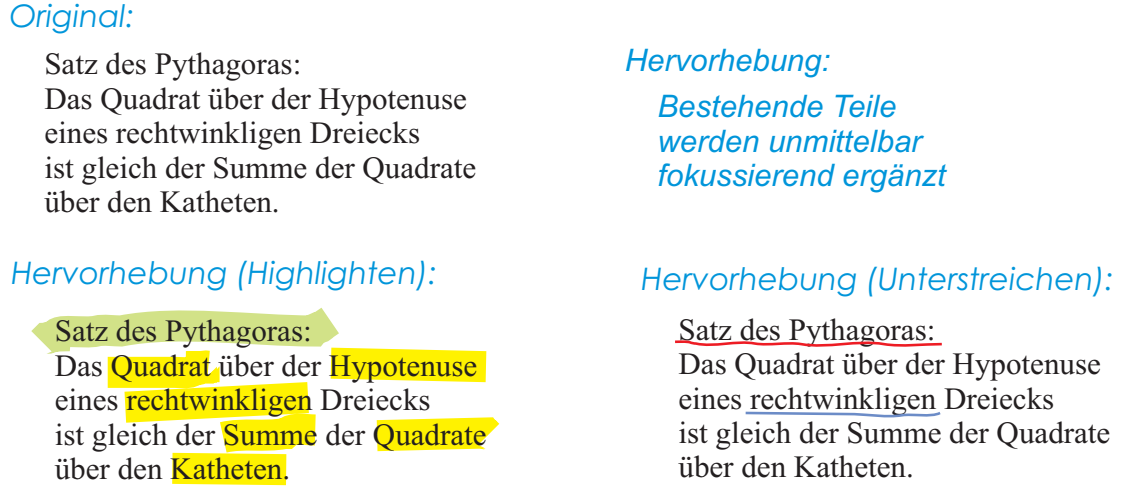


Abb. 2: Beispiele für Hervorhebungen

Hervorhebung unterstützt also das Lernen (vgl. auch Rickards & August, 1975).

Insgesamt ist für den sinnvollen Einsatz dieser Erscheinungsform einer Annotation unbedingt notwendig, dass die Grundlagen für die jeweilige Stelle im Kurs verstanden sind, da sonst eine Entscheidung über die Wichtigkeit einer Stelle kaum fundiert gefällt werden kann. So wurde gezeigt, dass die Effektivität dieser Methode bei Textmaterial direkt mit der Lesekompetenz des Studierenden zusammenhängt (vgl. Schnell, T.R. & Rocchio, D.J., 1978). Eine umfangreiche Zusammenfassung über diesbezügliche Literatur, eine eigene Untersuchung und weitere interessante Aussagen zum Unterstreichen liefert außerdem Caverly und Orlando (1991).
Vor diesem Hintergrund wirkt das Ergebnis der aktuellen Pisastudie zu deutschen Schülern noch alarmierender (Prenzel, Baumert, Blum, Lehmann, Leutner, Neubrand, Pekrun, Rolff, Rost & Schiefele, 2004). Die nach dieser Studie als unterdurchschnittlich bescheinigte Lesekompetenz verhindert somit einen gewinnbringenden Einsatz dieser Erscheinungsform einer Annotationen.

Highlighten oder Unterstreichen läßt sich sehr schnell bewerkstelligen. Damit sind diese Techniken der Hervorhebung gut geeignet, während einer Sitzung eingesetzt zu werden (vgl. Arnold, 1942). Außerdem hilft während eines Vortrags normalerweise die Betonung des Dozenten, wichtige Punkte zu identifizieren.

Die konkrete Entscheidung über die zur Hervorhebung eingesetzte Technik wird meist anhand der zur Verfügung stehenden Hilfsmittel gefällt. Steht ein Werkzeug zur Erstellung breiter transparenter Striche zur Verfügung, wird Highlighten der größeren Auffälligkeit wegen meist bevorzugt. Stehen nur Werkzeuge zur Erstellung opaker Linien zur Verfügung, bleibt nur das Unterstreichen oder Einrahmen als Mittel der Hervorhebung.

2.2.1.2. Stichpunkt

Unter dieser Erscheinungsform einer Annotation soll verstanden werden, dass Teile des Kurses

- inhaltlich
- als Randnotiz in einem Kursskript
- mit Schlagworten
- zusammengefasst

werden.

Zu betonen ist, dass Stichpunkte Teile des Kurses und nicht ausschließlich des Kursskripts zusammenfassen. So können also auch vom Dozenten präsentierte Teile des Kurses in Stichpunkten zusammengefasst werden.

Formulierungen wie „Wichtig!“ fallen, da sie keine inhaltliche Zusammenfassung darstellen, nicht in die Erscheinungsform Stichpunkt, sondern werden zu Symbolen (siehe 2.2.1.5.) gerechnet.

Werden wortwörtlich Phrasen, die dem *eigenen* Kursskript entstammen, direkt als Stichpunkt übernommen, unterscheidet sich dies prinzipiell nicht vom Gedanken des Hervorhebens, entspricht aber der Platzierung neben des ursprünglichen Kursskripts wegen nicht der Erscheinungsform Hervorhebung. Da Stichpunkte zu schreiben länger dauert als das Erstel-

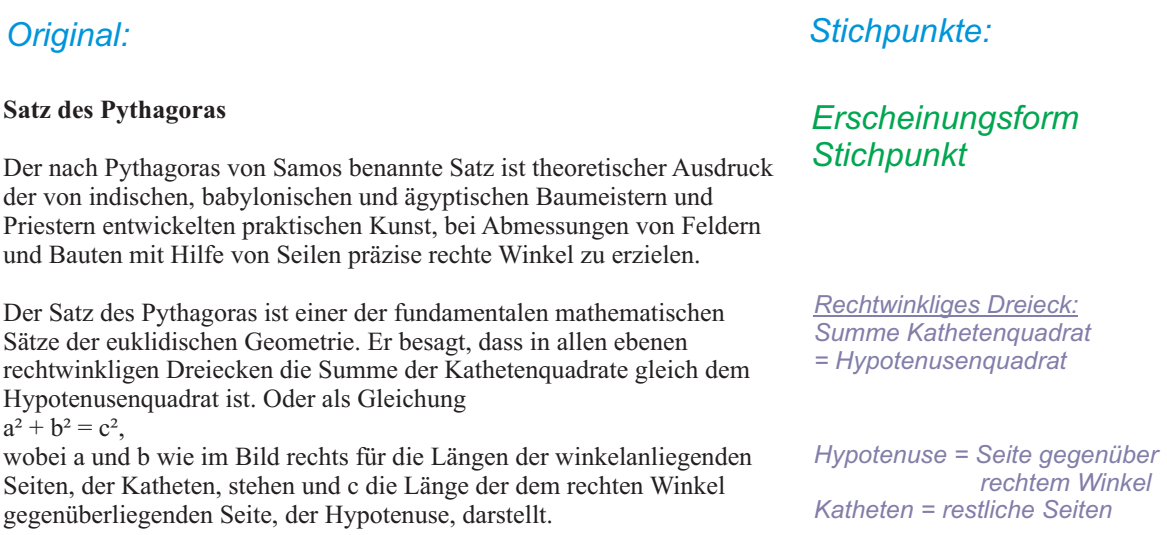


Abb. 3: Beispiel für Stichpunkte

len einer Hervorhebung, ist von einem Einsatz in diesem Sinne, *während* der Sitzung abzuraten (vgl. allgemein Zeitproblematik bei Annotationen Peters, 1972).

Während einer Sitzung Stichpunkte, die vom Dozenten präsentierte Teile des Kurses zusammenfassen, zu erstellen, ist dahingegen sehr verbreitet. Die Komprimierung auf Stichpunkte spart Zeit. Andererseits können Stichpunkte so formuliert werden, dass die Kernaussage erhalten bleibt. Sie steht also beim Wiederholen des Stoffes direkt zur Verfügung.

Beim *Nachbereiten* eines Kurses können Stichpunkte vorteilhaft eingesetzt werden: Durch das eigene Schreiben steigt die Wahrscheinlichkeit des sich Erinnerns. In der Literatur wird dies als *Encoding* bezeichnet. Außerdem bilden Stichpunkte bei entsprechendem und konsequentem Einsatz eine eigenständige Repräsentation des Kurses. Sie formieren dann also ein eigenes, stark reduziertes Kursskript. Dieses eignet sich in der Regel gut, einen Überblick über den Kurs zu geben (vgl. Abb. 3 und z.B. Mandl, 1981).

Als Hilfsmittel für die Erstellung von Stichpunkten sind Werkzeuge notwendig, die das Erstellen von Text erlauben. Zudem ist es für eine Strukturierung der Stichpunkte wünschenswert, eine weitere eigene Kategorisierung einführen zu können. Diese kann etwa über die Farbgebung oder Schriftart dargestellt werden.

2.2.1.3. Erläuterung

Bei Erläuterungen werden Teile des Kursskripts

- inhaltlich
- als Randnotiz
- mit bestehendem Wissen
- in Verbindung gebracht.

Erläuterungen helfen Querbezüge *inhaltlich* festzuhalten. Sie sind ein sehr attraktives Mittel zur Bildung eines Verständnisses der Kursinhalte. Eigene Querbezüge zu bestehendem Wis-

sen vergrößern in hohem Maße auch den Zugewinn an *dauerhaftem* Wissen (vgl. Pressley, McDaniel, Turnure, Wood & Ahmad, 1987). Zu betonen ist, dass es sich bei Erläuterungen um Verknüpfungen auf inhaltlicher Ebene handelt. Bei der Erläuterung wird also der Inhalt der zu verknüpfenden Information von außerhalb des ursprünglichen Kursskripts direkt angegeben und nicht nur ein Verweis auf diesen Inhalt (siehe 2.2.1.4. Querverweis, S. 15 und Abb. 4).

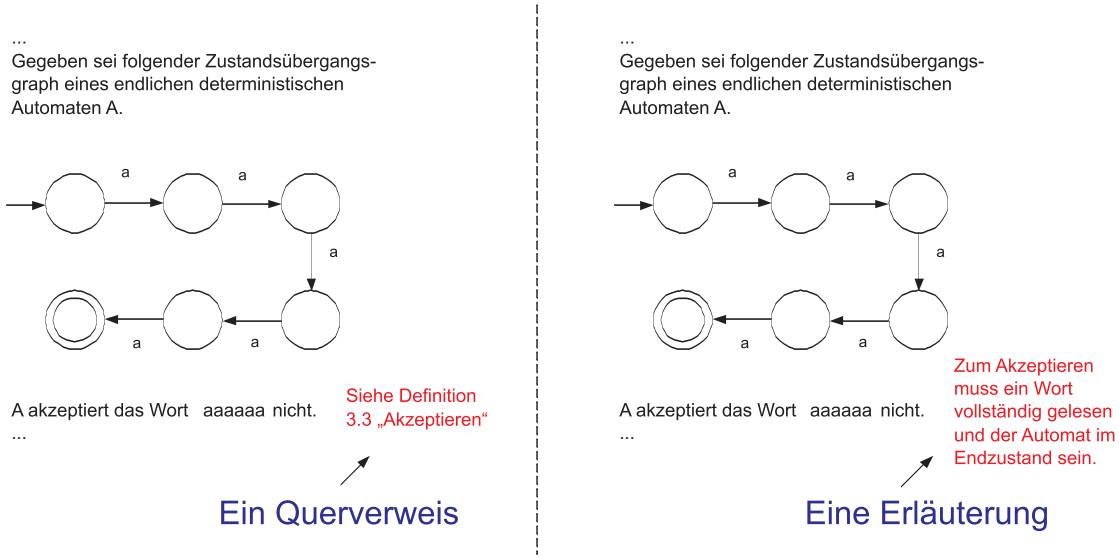


Abb. 4: Gegenüberstellung Querverweis - Erläuterung

Erläuterungen entstehen *während* einer Sitzung meist auf Grund von Ausführungen des Dozenten, da den Studierenden selbst selten Zeit bleibt, sich parallel zur Sitzung, eigenständig Gedanken über den neuen Stoff zu machen.

Nach einer Sitzung hat der Student mehr Zeit und kann zudem andere Wissensquellen konsultieren. Zusammen mit der Bildung dauerhaften Wissens begründet sich damit, warum Erläuterungen eine bedeutende Rolle beim Nacharbeiten einer Sitzung spielen.

Interessant an Erläuterungen ist, dass Sie oft auch direkt anderen Studenten helfen können. Dies liegt daran, dass die verknüpfte Information selbst direkt angegeben wird, sie aber auch nicht direkt aus dem gegebenen Kursskript wie bei Stichpunkten ableitbar ist. Eine Erläuterung stellt also eine echte Anreicherung des Kursskripts dar. Es wäre also wünschenswert, wenn eine Erläuterung eines Studenten auch direkt allen anderen Studenten zur Verfügung stehen würde.

Caverly und Orlando (1991) fassen auch zu Erläuterungen und Stichpunkten wissenschaftliche Arbeiten zusammen und liefern mit Ihrer Untersuchung weitergehende Aussagen.

Als Hilfsmittel zur Erstellung einer Erläuterung ist ein Werkzeug zur Texterstellung notwendig.

2.2.1.4. Querverweis

Eine Annotation, die eine Verbindung zwischen einem Teil des Kursskripts und anderen Quellen herstellt, wird als Querverweis bezeichnet. Dabei kann ein Querverweis auch auf einen anderen Teil des Kursskripts verweisen. Unterschied zur Erläuterung ist dabei, dass es

sich um eine rein formale Referenz handelt. Bei dieser Form der Annotation wird auf den Inhalt des referenzierten Teils nicht weiter eingegangen. Der Querverweis alleine liefert somit kaum zusätzliche Information zum Kursskript (vgl. Beispiel in Abb. 4).

Es werden also Teile des Kursskripts

- rein formal
- inhaltlich
- mit anderen Quellen
- in Verbindung gebracht.

Querverweise auf Teile *außerhalb* des Kursskripts werden meist zur formalen Begründung von Teilen des Kursskripts gemacht. Für das eigene Lernen werden von Studierenden Erläuterungen, die direkt den Inhalt anderer Quellen wiedergeben, bevorzugt. Insgesamt wäre eine Mischform, also zum Einen Erläuterung und zum Anderen Querverweis, worauf sich die in der Erläuterung geäußerten Erkenntnisse stützen, wünschenswert. Nebenbei ergäbe sich durch das wiederholte Kombinieren der Quellinformation mit dem interpretierten Inhalt auch dafür ein Lerneffekt, was insgesamt zu fundierterem Wissen führt.

Für die Erstellung von Querverweisen ist ein Werkzeug zur Texterstellung notwendig. Wird das papiergestützte System in Richtung elektronisches Kursskript erweitert, stehen die Möglichkeiten von Hypertext-Dokumenten zur Verfügung. Ein Querverweis entspricht dann in der Hypertextterminologie einem *Link*. Ein Unterschied zum papiergestützten Querverweis besteht bei einem Link darin, dass die Verweisstelle nicht direkt aus der Darstellung des Links ersichtlich sein muss. So wird zum Beispiel in einem HTML-Link zwar das Ziel des Links (als href=) angegeben. Angezeigt wird dem Benutzer in der HTML-Seite aber der Text zwischen den <a> Tags.

2.2.1.5. Symbol

Symbole werden eingesetzt, um Metainformation zu Stellen im Kursskript zu annotieren. Sie spiegeln also prinzipiell keinen Inhalt wieder. „Wichtig!“, „Prüfungsrelevant!“, ein Ausrufezeichen oder eine Fragezeichen sind Beispiele für diese Erscheinungsform von Annotationen.

Es werden also Teile des Kursskripts

- relativ zum Rest des Kurses
- als Randnotiz
- kategorisiert.

Symbole helfen den Lernprozess zu steuern. Sie folgen einer eigenen Sprache und besitzen eine eigene Semantik. Durch ihr oft grafisches Erscheinungsbild sind sie meist klar von inhaltlichen Annotationen zu unterscheiden. In der Regel sind Symbole schnell zu erstellen und können deshalb gut während einer Sitzung eingesetzt werden.

Da nur ein begrenzter Satz von Symbolen zum Einsatz kommt, ist deren Semantik schnell zu erlernen. Somit kann ein solcher Satz von Symbolen auch gemeinsam genutzt werden. Ein solches gemeinsames Nutzen lehrt dann am Beispiel anderer auch die Verwendung solcher Symbole und fördert somit den Einsatz dieser Technik.

Symbole können auch andere Annotationen ergänzen. Damit wird eine Kategorisierung innerhalb einer Erscheinungsform möglich. Außerdem lässt sich über Symbole auch ein die Erscheinungsformen von Annotationen und das Kursskript übergreifendes Kategoriesystem für Annotationen aufbauen (vgl. Abb. 5).

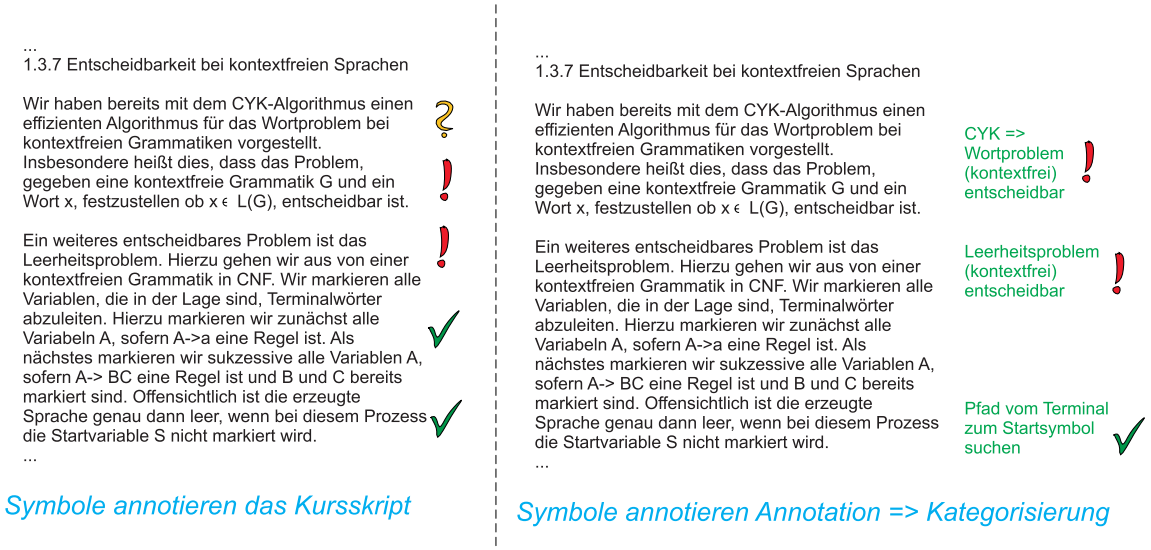


Abb. 5: Beispiele für Symbole

Symbole benötigen ihrer oft grafischen Darstellung wegen zur Erstellung Zeichenwerkzeuge. Darunter fallen Werkzeuge zum Erstellen von freien Linienzügen und Werkzeuge zum Erstellen gefüllter Flächen. Dabei sollte eine freie Farbgestaltung möglich sein. Damit Symbole die für die Wiedererkennung notwendige Gleichheit aufweisen, erweisen Stempel für Symbole einen guten Dienst.

2.2.1.6. Veranschaulichung

Eine Veranschaulichung ist eine in ein anderes Symbolsystem transformierte Darstellung des Inhalts (vgl. Weidenmann, 1994 und Abb. 6). Es handelt sich dabei meist um eine Transformation vom sprachlichen oder numerischen Symbolsystem in eine bildliche Darstellung. Es entstehen Abbilder, logische Bilder, Karten, Cartoons und Bildfolgen.

Es werden also Teile des Kursskripts

- inhaltlich interpretiert
- in eine andere Darstellung transformiert
- mit dem Ziel, in der anderen Darstellung subjektiv leichter erfassbar zu sein.

Da eine Veranschaulichung eine in ein anderes Symbolsystem transformierte Darstellung ist und jedes Symbolsystem über eigene Gehirnregionen interpretiert wird, werden über Veranschaulichungen weitere Gehirnregionen mit in den Lehrprozess einbezogen. Dies steigert nachweislich den Lerneffekt (vgl. Vester, 1978).

Vester (1978) teilt Studenten außerdem in verschiedene Lerner ein. Seiner Meinung nach werden umso mehr Studenten erreicht, je mehr verschiedene Darstellungssysteme benutzt werden. Somit sollten Veranschaulichungen in keinem Kurs fehlen.

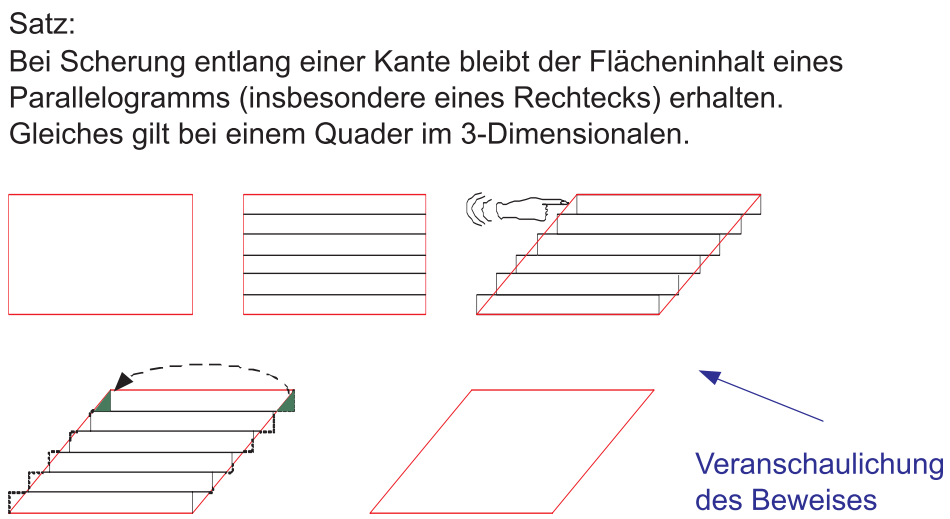


Abb. 6: Beispiel für eine Veranschaulichung

Mappingtechniken wie MindMaps (vgl. Buzan, 1993) werden auch dem Bereich Veranschaulichung zugeordnet. Maps stellen Objekte und ihre Beziehungen in nicht linearer Form dar.

Da eine Veranschaulichung in papiergestützten Systemen meist eine Transformation in eine bildliche Darstellung ist, sind zur Erstellung einer Veranschaulichung Zeichenwerkzeuge notwendig. Die Zeichenwerkzeuge sollten der Kreativität kaum Grenzen setzen und eine farbliche Ausgestaltung zulassen. Ratschläge hierzu bietet zum Beispiel Buzan (1993).

2.2.1.7. Korrektur

Die Erscheinungsform *Korrektur* bietet die Möglichkeit, bestehende Teile eines Kursskripts auszubessern. Eine Korrektur betrifft zumeist alle Beteiligten und wird in der Regel vom Dozenten autorisiert.

Es werden also

- Teile des Kursskripts
- meist rückwirkend
- als falsch gekennzeichnet
- und durch eine Ergänzung richtig gestellt.

Da eine Korrektur für alle Beteiligten von Interesse ist, wäre eine automatische Ausbreitung wünschenswert. In der papiergestützten Praxis werden Korrekturen oft gesammelt als Anhang zentral zur Verfügung gestellt. Es liegt dann in der Verantwortung jedes Einzelnen, sich über Korrekturen zu informieren.

Da potenziell jeder Bestandteil eines Kursskripts eine Korrektur erfahren kann, muss ein Korrekturwerkzeug die Möglichkeiten der Werkzeuge zur Erstellung des Kursskripts bieten.

2.2.2. Erscheinungsformen in multimedialen Umgebungen

Nach den klassischen, mehr papiergestützten Annotationsformen werden nun Annotationsformen, die erst mit dem einsetzenden Multimediazeitalter praktikabel wurden, aufgeführt.

2.2.2.1. Audiosequenzen

In bestimmten Berufssparten oder Positionen ist es üblich, über ein Diktiergerät Annotationen zu einem Dokument zu verfassen. Die Motivation dafür liegt hauptsächlich darin, dass sich der Ersteller der Annotation zum Einen auf den Inhalt des zu annotierenden Dokuments konzentriert und zum Anderen „nebenbei“ eine Annotation erstellt. Dies wird möglich, da zwei verschiedene Kommunikationskanäle unabhängig voneinander genutzt werden: der visuelle Sinn liefert die Eingabe für den Denkprozess und die Sprache stellt das Ausgabemedium dar. Für die anderen, bisher aufgeführten Annotationsformen muss der visuelle Sinn neben der Aufnahme des Kursskripts oder Sitzungsgeschehens auch die Kontrolle der Annotationserstellung übernehmen.

Mit einer Audioannotation werden also

- Teile des Kursskripts
- mit annotierenden Audiosequenzen
- verknüpft.

Vor den multimedialen Möglichkeiten von heute war Audioannotation im Lehrumfeld kaum praktikabel. Das Kursskript als zu annotierendes Dokument ist meist recht umfangreich. Der Vorteil des einfachen Erstellens einer Audioannotation wird durch das umständliche Wiederfinden der zu einer Stelle im Kursskript gehörenden Annotation aufgehoben.

Mit den heutigen multimedialen Möglichkeiten an einem Rechner ist das Verknüpfen von Audioannotationen mit einer Stelle im Kursskript technisch kein Problem mehr. Damit steht sie automatisch zur Verfügung, wenn die Stelle im Kursskript dargestellt wird.

Semantisch gesehen kann der Inhalt der Audioannotation den meisten oben aufgeführten Erscheinungsformen entsprechen. Eine Audioannotation kann also einem Stichpunkt, einer Erläuterung oder etwas anderem aufgeführten gleichkommen. Trotzdem wird die Audiosequenz hier als eigene Erscheinungsform geführt. Gerechtfertigt wird dies durch die oben angeführte Sonderstellung bei den Kommunikationskanälen.

Für die Erstellung einer Audioannotation müssen neben den technischen Geräten (z.B. Mikrofon) ein Werkzeug zur Steuerung und Ablage der Aufnahme zur Verfügung gestellt werden. Außerdem muss dieses Werkzeug die Verbindung zwischen der Audiosequenz und der entsprechenden Kursskriptstelle etablieren.

2.2.2.2. Videosequenzen

Die fortschreitende Technik ermöglicht heute kostengünstige Kameras, die zum Teil schon in Gebrauchsgegenstände wie Mobilfunktelefone eingebaut sind. Technisch und finanziell ist es also auch für einen Studenten durchaus vorstellbar, überall zu jeder Zeit Videosequenzen zu erstellen.

Im Unterrichtsumfeld wird dazu zuerst die Aufzeichnung des Unterrichtsgeschehens, eventuell vom Dozenten organisiert, ins Auge stechen. Über diesen sehr begrenzten Anwen-

dungsbereich hinaus, sind bei weitem breitere Anwendungsfälle denkbar. Führt ein Kurs eine Exkursion durch, kann jeder Student seine eigenen Videosequenzen aus seinem persönlichen Blickwinkel erstellen und damit das Kursskript annotieren. Darüber hinaus sammelt jede Person privat eigene Eindrücke in seiner Umwelt. Werden sie in einer Videosequenz festgehalten, können diese - nicht vom Dozenten initiierten Annotationen - mit dem Kursskript verknüpft werden. Ein einfaches Beispiel wäre die Aufnahme zweier zufällig auf einem Spielplatz wippender Kinder zur eigenen Veranschaulichung der Hebelgesetze in der Mittelstufe des Gymnasiums.

Wie aus dem Beispiel hervorgeht, entstehen über Videosequenzen meist Veranschaulichungen und Erläuterungen, die Verbindungen zu bestehendem Wissen oder Erfahrungen herstellen.

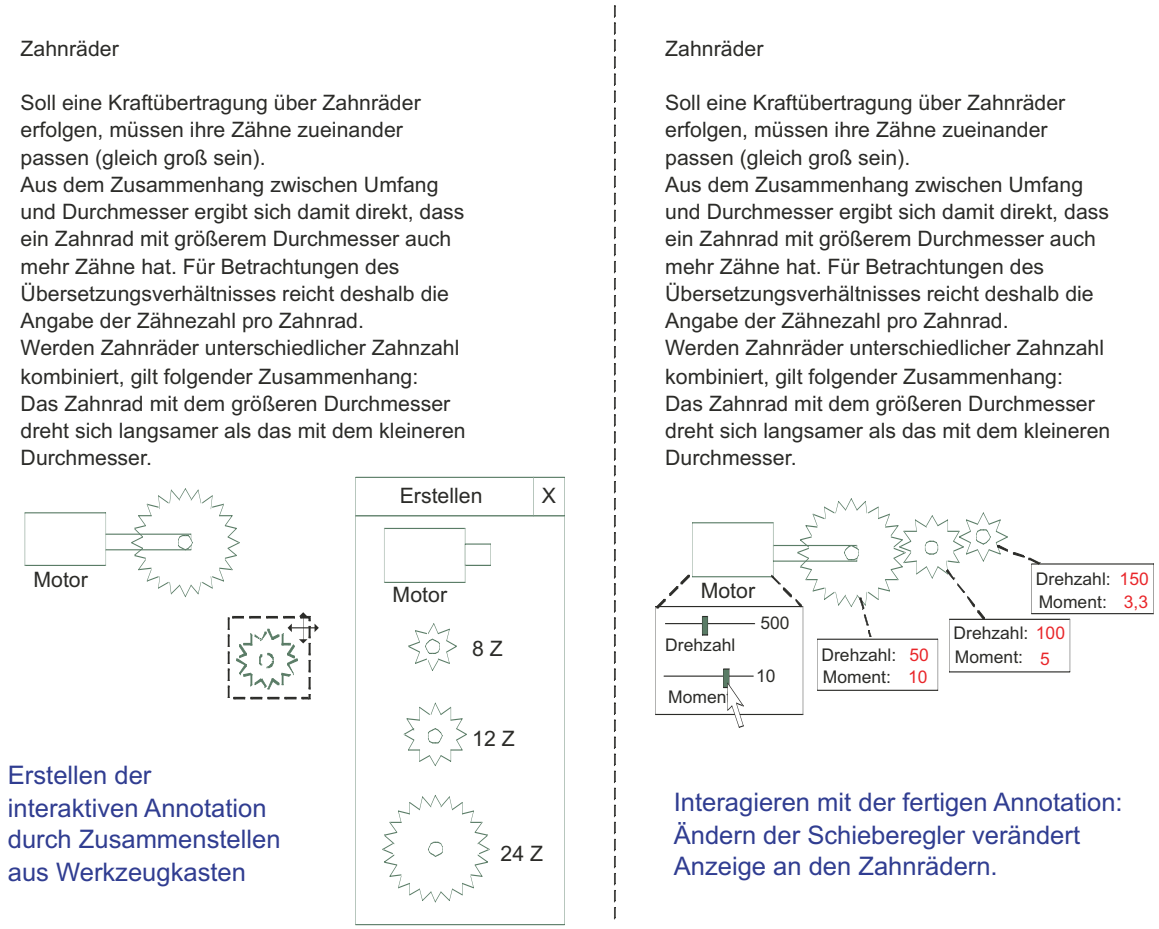


Abb. 7: Beispiele für interaktive Annotationen

Die Sinnhaftigkeit und Mächtigkeit dieser Möglichkeit wurde auf pädagogischer Seite noch nicht weiter erforscht. Bisher standen dazu nicht die notwendigen Systeme und technischen Geräte zur Verfügung. Interessant scheint in diesem Kontext gerade auch die Möglichkeit, Annotationen anderer Personen betrachten zu können. Damit würde auf den Erfahrungsschatz anderer zurückgegriffen werden können.

2.2.2.3. Interaktive Elemente

Interaktive Elemente stellen unter allen angeführten Erscheinungsformen von Annotationen den diesbezüglich am wenigsten untersuchten Bereich dar. Diese Erscheinungsform hat ihrer Verbreitung wegen momentan eher visionären Charakter. Unter der Erscheinungsform *interaktive Elemente* soll verstanden werden, dass dem Annotierenden ein Erstellungswerkzeug an die Hand gegeben wird, welches ihm auf einfache Weise erlaubt, interaktive Annotationen zu erstellen. Interaktiv bedeutet in diesem Zusammenhang, dass der Betrachter einer solchen Annotation durch eine Aktion eine Reaktion der Annotation hervorrufen kann.

Einfache, bei der Erstellung gestaltbare Simulationen sind eine denkbare Variante dieser Erscheinungsform (vgl. Abb. 7, Krupina, 2005 oder Liwicki, 2004).

Wichtig bei der Umsetzung dieser Erscheinungsform und schließlich der Erstellung entsprechender Annotationen ist, dass kein universelles Simulationswerkzeug für alle möglichen Fragestellungen des Kurses entwickelt wird. Dafür gibt es zusätzliche, eigenständige Werkzeuge, die weitaus besser geeignet sind (vgl. z.B. Maple (2005) in Mathematik). Es muss vielmehr das Wesen einer Annotation als Ergänzung einer speziellen Stelle im Kursskript beachtet werden. Eine interaktive Annotation darf also auch nur genau den Aspekt der Stelle im Kursskript behandeln, um nicht den Bezug dazu zu verlieren.

Da es für diese Erscheinungsform noch keinerlei reine Umsetzungen als Annotationswerkzeug und damit auch keine Untersuchungen auf pädagogischer Ebene gibt, kann über ihren Wert für die Lehre noch nichts ausgesagt werden. Weil in der Pädagogik aber immer der Stellenwert eigener Erfahrungen betont wird (vgl. z.B. Dewey, 1938, Savery & Duffy, 1995, u.a.), sollte diese Annotationsmöglichkeit weiter verfolgt werden.

2.2.3. Folgerungen aus den Erscheinungsformen

In diesem ersten zusammenfassenden Schritt ergeben sich aus der Vorstellung der verschiedenen Erscheinungsformen, die in Tabelle 1 zusammengestellten Anforderungen an einen *Satz von Werkzeugen* zur Erstellung von Annotationen.

Bei Betrachtung dieser Aufstellung wird deutlich, dass durch bloßes Anbieten eines Werkzeugs für Freihandstriche die meisten Erscheinungsformen abgedeckt werden können. Dies und die Vielfalt an Attributierungsmöglichkeiten unterstreicht, dass ein Werkzeug für Freihandstriche bei elektronischer Annotationsunterstützung immer implementiert werden sollte. Außerdem erlaubt das Werkzeug Freihandstrich dem Benutzer größtmögliche Freiheit und Kreativität. Dies ist in einem Lehr- und Lernumfeld der besseren Akzeptanz und damit der geringeren Motivationsproblemen wegen wünschenswert.

Der Vermutung, dass mit Anbieten eines Werkzeugs zur Erstellung von Freihandstrichen auf die anderen Werkzeuge verzichtet werden kann, ist trotzdem mit Vorsicht zu begegnen. Die Vorteile spezialisierter Werkzeuge überwiegen oft den möglichen Zusatzaufwand (Werkzeugwechsel, Umgang erlernen). Eine im Rahmen dieser Arbeit durchgeführte Umfrage zeigt, dass Studenten von einem Annotationssystem erwarten, dass es auch eine Texteingabe bietet. Von 387 Befragten, gaben nur 29 an, sie würden ein Annotationssystem benutzen, auch wenn es nur Freihandstriche bietet. Dies entspricht gerade einmal 7,5%. Umgekehrt wären 222 Studenten schon mit einer Art von Texteingabe zufrieden, was 57% entspricht (vgl. Schütz, 2005a, Frage 5).

Werkzeug	Auch simulierbar durch Werkzeug	Mögliche Attributierung über	Erscheinungsform
Markierung	Freihandstrich	Farbe	Hervorhebung
Texteingabe	Freihandstrich	Farbe, Schriftart, Stil	Stichpunkt, Erläuterung, Querverweis
Freihandstrich		Farbe, Strichstärke, Strichaufbau	Hervorhebung, Stichpunkt, Erläuterung, Querverweis, Symbol, Veranschaulichung, Korrektur
Stempel	Grafische Objekte	Farbe, Größe	Symbol
Grafisches Objekt	Freihandstrich	Farbe, Größe	Veranschaulichung
Korrekturwerkzeug	Freihandstrich (evtl. plus Markierung)	Farbe	Korrektur
Aufnahmesteuerung mit Zuordnung zu Kursskriptstelle		Verknüpfungssymbol	Audiosequenz
Aufnahmesteuerung mit Zuordnung zu Kursskriptstelle		Verknüpfungssymbol	Videosequenz
„Baukasten“		Verknüpfungssymbol	Interaktive Elemente

Tab. 1: Werkzeuge zur Erstellung von Annotationen

Aus den multimedialen Annotationen, speziell den praktischen Erfahrungen mit Audiosequenzen zur Annotation, lässt sich direkt eine weitere zentrale Forderung ableiten: Eine Annotation soll sich auf dem gleichen Medium befinden, wie der zu annotierende Inhalt. Damit können aufwendige und damit lästige Such- und Zuordnungsmechanismen umgangen werden. Dieser sogenannte Medienbruch ist also der Grund, wieso sich die angesprochenen multimedialen Annotationen in papiergestützten Systemen nicht etablieren konnten. In einer elektronischen, computergestützten Umgebung können das Kursskript und die multimedialen Annotationen über das gleiche Medium angeboten werden. Ein Medienbruch wird vermieden.

Befinden sich Annotationen auf dem gleichen Medium wie das Kursskript, ist es möglich, einen örtlichen Bezug zu den entsprechenden Stellen im Kursskript herzustellen. Im Weiteren soll dieser Aspekt *Lokalität der Annotationen* genannt werden. Am Beispiel der Erscheinungsform Korrektur lässt sich dies anschaulich darstellen: Werden Korrekturen gesammelt im Anhang aufgeführt (Errata), so spricht man von schlechter Lokalität der Annotation. Werden Korrekturen hingegen direkt an entsprechender Stelle im Kursskript abgelegt, ist die Lokalität dieser Annotation gut.

Nach diesen noch recht allgemeinen Aussagen zu Annotationen wird nun in den nächsten Abschnitten über die Benutzerrollen *Student* und *Dozent* der Bezug zu Lernen und Lehre verstärkt.

2.3. Studentenannotation

2.3.1. Allgemein

Dieses Unterkapitel betrachtet Annotationen aus Studentensicht. Dabei wird komprimiert dargestellt werden, wie sich aus pädagogischer Sicht der Einsatz von Annotationen aus studentischer Perspektive begründet.

Zu Beginn zwei Zahlen, die für sich alleine schon eine Auseinandersetzung mit der Thematik Studentenannotation rechtfertigen: Nach Palmatier und Bennet (1974) machen 99% der Collegestudenten Annotationen während eines Kurses. 94% der amerikanischen Studenten glauben nach Dunkel und Davy (1989), dass Annotieren eine wertvolle und wichtige Aktivität ist.

Grundsätzlich gilt, dass Studentenannotationen zu zwei unterschiedlichen Phasen entstehen:

- Während einer Sitzung
- Beim Nachbereiten einer Sitzung

Diese beiden unterscheiden sich wie folgt. *Während der Sitzung* ist oft von Zeitdruck geprägt. Der Studierende macht seine Annotationen, während der Dozent vorträgt. Dabei wird der Dozent nicht auf jeden Studenten warten. Um den Ausführungen des Dozenten folgen zu können, müssen Annotationen zu diesem Zeitpunkt sehr schnell zu erstellen sein. Die Gedanken der Studenten müssen beim Dozenten bleiben. Weiterführende, abschweifende Überlegungen kann sich der Student zeitlich gesehen kaum leisten (vgl. Kiewra & DuBois, 1991). Der Student wird meist auf Erscheinungsformen von Annotationen zurückgreifen, die den vom Dozenten vorgestellten Stoff gliedern, damit sich der Student beim Wiederholen schnell zurechtfindet. Der Student nutzt hier oft unbewusst einfache Organisationsstrategien, wie sie weiter unten genauer erläutert werden.

Anders liegt die Situation *beim Nachbereiten einer Sitzung*. Hier hat der Student die Möglichkeit mehr Zeit in die Erstellung von Annotationen zu stecken. Somit kann er weitere Informationsquellen konsultieren, den vermittelten Stoff in einen größeren Kontext stellen und seine diesbezüglichen Ergebnisse für weitere Wiederholungen als Annotation ins Kursskript übernehmen (vgl. Kiewra & DuBois, 1991). Dieses Vorgehen kennzeichnet einen anderen Satz von Strategien, den Stoff zu verinnerlichen. Elaborationsstrategien und weitergehende Organisationsstrategien sind lernpsychologische Kategorien, in die sich dieses Verhalten einordnen lässt. Auch diese werden bei den Funktionen von Studentenannotationen weiter unten genauer betrachtet.

Neben dieser zeitlichen Einteilung der Studentenannotationen gibt es weitere grundsätzliche Beobachtungen: Studentenannotationen sind primär privat, da sie heutzutage in erster Linie für den *eigenen* späteren Gebrauch verfasst werden. Der Student achtet also nicht darauf, einer allgemeinbekannten Norm zu folgen. Durch seine erziehungsgeschichtliche Vorgeschichte ist es aber unwahrscheinlich, dass er ein komplett eigenes, für andere unverständliches Annotationssystem verwendet.

Diese Aussage betrifft die Erscheinungsform der Annotation. Der Inhalt beziehungsweise Gehalt hängt natürlich vom eigenen individuellen Wissen und Wesen ab.

In diesem Zusammenhang interessant ist, dass es Untersuchungen gibt, dass Annotationen, obwohl sie von einem anderen Studenten stammen, einem Studenten helfen können, den Stoff zu verinnerlichen (vgl. Kiewra & DuBois, 1991). In papiergestützten Umgebungen ist es aber kaum realisierbar, den Studenten kontinuierlich Einblick in die Annotationen anderer zu geben. Hier könnte, wie später angeregt, eine elektronische Unterstützung von Annotationen gewinnbringend Abhilfe schaffen.

2.3.2. Funktionen von Studentenannotationen

Wieso schreiben Studierende Annotationen in ihr Kursskript? Eine kurze und einleuchtende Antwort ist: Annotationen dienen neben dem Kursskript als Gedächtnisstütze für die Wiederholung des Stoffes.

Der Student besucht einen Kurs, um sich Wissen anzueignen. Da dieses Wissen in der Regel nicht durch den Besuch des Kurses alleine verinnerlicht werden kann, benötigt der Student ein Kursskript als Ausgangspunkt für das Nacharbeiten des Stoffs. Eventuell besteht ein solches „ausgehändigtes“ Kursskript auch nur aus der Gliederung des Kurses oder weniger (vgl. zu Formen von ausgegeben Kursskripten in bezug zu Annotationsverhalten Hartley, 1976).

Im Kursskript werden in der Regel die Gedankengänge des Verfassers zum jeweiligen Thema aufgeführt. Nachdem Denken aber durch die unterschiedliche Vorbildung und das unterschiedliche Umfeld ein sehr individueller Vorgang ist, kommt es nicht selten vor, dass der Student mit diesen Gedankengängen des Dozenten oder Autors alleine nicht in der Lage ist, effizient den Stoff zu wiederholen (vgl. Vester, 1978). Deshalb macht sich der Student während des Kurses selbst Annotationen in sein privates Exemplar des Kursskripts.

DiVesta und Gray (1972) haben bei ihrer Untersuchung zu Annotationen im klassischen Lernumfeld zwei Hauptfunktionen von Studentenannotationen herausgestellt:

- Encoding
- External Storage

Diese beiden Funktionen sind von Kiewra und DuBois (1991) weiter differenziert und daraus resultierend um eine dritte, *encoding plus storage* ergänzt worden. Es folgt eine kurze Erklärung zu den drei Funktionen von Annotationen aus Studentensicht.

– Encoding

Unter *Encoding* versteht die Lernpsychologie den Vorgang des Lernens, sprich also des sich Verinnerlichens. Annotationen dienen diesem Encoding in dem Sinne, dass das eigenständige Formulieren und Aufschreiben einer Information den Lernprozess fördert. „Was ich selbst geschrieben habe, kann ich mir besser merken, als wenn ich es nur gehört habe!“ ist ein in diesem Zusammenhang oft gehörter Ausspruch.

Ob Annotationen dem Studierenden in diesem Sinne helfen, kann durch folgendes unter anderem von DiVesta et al. (1972) durchgeführte Experiment überprüft werden: Zwei Gruppen von Studenten besuchen den selben Kurs. Die einen dürfen während des Kurses Notizen machen, die anderen nicht. Anschließend wird ein Test über den Kurs gehalten. Keine der beiden Gruppen darf dabei nach den Sitzungen das Kursskript beziehungsweise die Annotationen betrachten. Es wird also nur bewertet, ob das bloße Aufschreiben von Annotationen den Lernprozess schon positiv beeinflusst. Solche Experimente fallen der Literatur nach unter-

schiedlich aus (vgl. z.B. Hult et al., 1984, Henk & Stahl, 1985, Hartley, 1983, Kiewra, 1985). Die Grundrichtung, dass Annotationen das Encoding fördern, ist aber zu erkennen.

– External storage

External storage im von Kiewra und DuBois neu definierten Sinn steht für eine Repräsentation, die den Inhalt eines Kurses oder eigener Gedanken in niedergeschriebener, archivierbarer Form wiedergibt. Damit wird also der Kurs „gespeichert". Das Kursskript, aber auch die Annotationen zählen also zum External storage. Mit Hilfe des External storage kann der Stoff eines Kurses später unabhängig von den Sitzungen eingeübt werden. Kiewra und DuBois betonen, dass der sauberen Abgrenzung zum Encoding wegen, bei der Beurteilung von External storage als Hilfe beim Lernen, der Studierende den External storage nicht selbst angelegt haben darf. Ist dies nicht der Fall, können Effekte des Encodings Ergebnisse zu External storage verfälschen. Als Beispiel aus der Praxis für External storage führen Kiewra und DuBois folgendes Szenario an: Ein Student, der eine Sitzung nicht besucht hat, leiht sich von einem Kommilitonen dessen Kursskript mit dessen Annotationen. Dies ist im universitären Umfeld durchaus üblich. Die Frage ist nun also, ob Annotationen, welche nicht selbst verfasst sind, das Lernen unterstützen. Kiewra und DuBois (1991) kommen in ihren Untersuchungen zu dem Ergebnis, dass dies der Fall ist.

– Encoding plus storage

Encoding plus storage umschreibt nun den Umstand, dass ein Student während der Sitzung eigene Annotationen macht und diese dann auch selbst wieder zu einem späteren Zeitpunkt betrachtet. Dies entspricht also der Kombination der beiden eben genannten Funktionen. Andere Autoren bezeichnen manchmal diese Kombination als die External storage Funktion. Encoding plus storage entspricht am ehesten der natürlichen Vorgehensweise eines Studierenden. Im Bezug auf Annotationen heißt dies, der Studierende macht sich während des Kurses Notizen, die er anschließend beim Wiederholen des Stoffes nach der Sitzung wieder betrachtet. Diese gekoppelte Sichtweise läßt, wie oben ausgeführt, aber keine getrennte Bewertung der beiden Aspekte zu. Insgesamt sei dieses Vorgehen der Literatur nach das erfolgreichste der drei vorgestellten, was zum Beispiel Kiewra et al. (1991) mit der zweifachen Beschäftigung mit dem Material begründen.

Übereinstimmend ergeben lernpsychologische Untersuchungen, dass Annotationen über diese Mechanismen das Lernen eher unterstützen. Andererseits zeigen die zum Teil unterschiedlichen Ergebnisse und Versuchsaufbauten auch, dass ohne weitere Ausbildung und Überlegung erstellte Annotationen nicht das mögliche Potential an Lernförderung, das Annotationen inne liegt, ausnutzen (vgl. Weinstein, 1982).

Lernstrategien befassen sich nun gerade mit der Effizienzsteigerung des Lernvorgangs auf Seite der Studierenden. Dabei werden unter anderem Annotationen genutzt. Im folgenden wird dieser Aspekt genauer betrachtet und damit eine weitere Begründung für Annotationen in der Lehre gegeben.

2.3.3. Beziehung zu Lernstrategien

Wie oben erwähnt, haben Lernstrategien eine Effizienzsteigerung des Lernvorgangs zum Ziel. Weinstein und Mayer (1985) haben folgende in der Lernpsychologie anerkannten acht Hauptkategorien für Lernstrategien herausgearbeitet:

- Basic Rehearsal Strategies (einfache Wiederholungsstrategie)
- Complex Rehearsal Strategies (komplexe Wiederholungsstrategie)
- Basic Elaboration Strategies (einfache Elaborationsstrategie)
- Complex Elaboration Strategies (komplexe Elaborationsstrategie)
- Basic Organizational Strategies (einfache Organisationsstrategie)
- Complex Organizational Strategies (komplexe Organisationsstrategie)
- Comprehension Monitoring Strategies (Strategien zur Verständniskontrolle)
- Affective and Motivational Strategies (Affektive und motivationssteigernde Strategien)

Jeder Student setzt dabei bewusst oder unbewusst eine Kombination dieser Strategien während seines Lernprozesses ein.

Im Bereich der Forschung zu Lernstrategien wird außerdem zwischen oberflächenorientiertem und tiefenorientiertem Lernen unterschieden (vgl. Marton & Säljö, 1976a+b, Wild, 1996). Oberflächenorientiertes Lernen hat mit seiner starken Ausrichtung auf Auswendiglernen meist nur das Ziel, eine Prüfung erfolgreich zu bestehen. Ein tieferes Verständnis des Stoffes ist hierbei kein Schwerpunkt.

Aus Sicht des Lernens stellt tiefenorientiertes Lernen die bessere Alternative dar, um sich dauerhaft Wissen anzueignen. Beim tiefenorientierten Lernen wird neuer Stoff mit bestehendem Wissen in Verbindung gebracht. Es entstehen Querverbindungen, die zum Einen für ein tieferes Verständnis sorgen und zum Anderen einen besseren Zugriff auf das neue Wissen erlauben.

Zwei Lernstrategien, bei denen Annotationen auf natürliche Weise eine Rolle spielen und die Ihrer Zuordnung zu den tiefenorientierten Lernstrategien wegen für dauerhaftes Lernen vielversprechend sind, sollen nun genauer betrachtet werden: Elaborationsstrategien und Organisationsstrategien. Damit wird die Rechtfertigung für die Beschäftigung mit Annotationen auf Studentenseite im Lehrumfeld abgeschlossen werden.

– Elaborationsstrategien

Mit dem Begriff der Elaborationsstrategien werden Lerntätigkeiten bezeichnet, die dazu geeignet sind, das neu aufgenommene Wissen in die bestehende Wissensstruktur zu integrieren. Elaborationsstrategien umfassen unter anderem (vgl. Seel, 2000, S.72):

- die Bildung von Analogien zu bereits bekannten Zusammenhängen und vorhandenen Wissensstrukturen;
- eine Verknüpfung des neu gelernten Materials mit Alltagsbeispielen sowie persönlichen Erlebnissen;
- das Herstellen von Beziehungen zwischen neuem Wissen und den Inhalten verwandter Fächer bzw. Lehrveranstaltungen;
- das Ausdenken von konkreten Beispielen;
- Überlegungen zu praktischen Anwendungsmöglichkeiten;

- Herstellung von Beziehungen zum Vorwissen - unter Zuhilfenahme von Beispielen, Analogien, Mnemotechnik (Gedächtniskunst, Steigerung der Gedächtnisleistung durch Hilfsvorstellungen nach Schema) und Visualisierungen;
- Benutzen von Entwürfen, Plänen und Aufzeichnungen.

Insgesamt wird also durch Elaborationsstrategien der neue Stoff mit bestehendem Wissen in Beziehung gesetzt.

– Organisationsstrategien

Organisationsstrategien zielen darauf ab, den Stoff zu strukturieren und in bestehende bekannte Systeme einzuordnen. Dies kann durch Hervorhebungen und Querverweise erreicht werden. Das Wiederholen des Stoffs wird somit durch das Weglassen unwichtiger Teile effizienter.

Eine weitere Technik, die unter Organisationsstrategien fällt, ist das sogenannte Mapping. Beim Mapping versucht der Studierende einen Sachverhalt oder den Inhalt eines Textes in einer Grafik zu verdeutlichen. Dazu werden die Hauptkonzepte als grafische Objekte dargestellt und durch Kanten in Beziehung zueinander gesetzt. Eine populäre Variante einer Mapping Technik sind Mind Maps, die von Tony Buzan entwickelt wurden (vgl. Buzan, 1993).

Beiden Lernstrategien ist gemeinsam, dass der Studierende eigene Annotationen erstellt.

Ein Ergebnis der Forschung im Bereich von Lernstrategien ist also, dass Annotationen ein Mittel zur Effizienzsteigerung des Lernens sind. Es hat sich aber auch gezeigt, dass der geschickte Einsatz erst gelernt werden muss (Nist & Simpson, 1988). So annotieren die meisten sich selbst überlassenen Studenten einzelne wortwörtlich übernommene Stichpunkte (vgl. Bretzing & Kulhavy, 1981). Dabei wird dann oftmals so stark reduziert, dass die logische Verknüpfung zwischen den Stichpunkten verloren geht. Lernpsychologen sprechen von unvollständigen Annotationen. Es werden also nur wenige Erscheinungsformen von Annotationen und diese dann nicht geeignet eingesetzt. Durch eine geschickte Benutzerführung im elektronischen Lernumfeld kann hier sicherlich einiges verbessert werden.

2.4. Dozentenannotation

Nicht nur Studenten ergänzen das bestehende Kursskript. Unter die Definition des Begriffs „Annotation“ in dieser Arbeit fallen neben den Studentenannotationen auch jegliche Ergänzungen des Dozenten während des Kurses.

2.4.1. Funktionen von Dozentenannotationen

In der klassischen Lehrsituation ergeben sich zwei Hauptgründe für Dozentenannotationen.

Einerseits handelt es sich um den geplanten didaktischen Einsatz von Dozentenannotationen. Der Dozent konzipiert seinen Unterricht so, dass er dynamisch während einer Sitzung das Kursskript durch Dozentenannotationen ergänzt oder erst konkret erstellt. Zum Beispiel fördert das vorgeführte Erstellen einer Grafik das Verständnis mehr, als das bloße Präsentieren der fertigen Grafik (vgl. Zirk, 1969, Weidenmann, 1994). Durch das unmittelbar vorgeführte Erstellen der Annotation kann die Aufmerksamkeit der Studenten fokussiert werden. Somit werden Dozentenannotationen zu einem wichtigen didaktischen Hilfsmittel.

Andererseits sind Dozentenannotationen oft das Mittel der Wahl, um auf konkrete Unterrichtsituationen reagieren zu können. Ergeben sich zum Beispiel Fragen von Studenten, können diese mit Hilfe von Dozentenannotationen anschaulich geklärt werden. Außerdem sind Dozentenannotationen geeignet, Studentenbeiträge und Arbeitsergebnisse zu sammeln, fixieren und allen anderen Studenten zur Verfügung zu stellen. Speziell der letzte Punkt belegt, dass öffentliche Annotationen wie Dozentenannotationen keinen Frontalunterricht bedingen.

Dozentenannotationen ziehen in der klassischen Lehre sehr oft Studentenannotationen nach sich (vgl. Laurillard, 1978). Die Studenten übertragen die öffentliche Annotation als eigene Annotation in ihr privates Skript. Dies kostet Zeit, hilft aber auch den Sachverhalt durch eigenes Aufschreiben schon stärker zu verinnerlichen (vgl. Aspekt Encoding Kap. 2.3.2., S. 24). Somit stellen Dozentenannotationen auch ein Mittel dar, das Lernen des Studierenden aktiv zu unterstützen.

Zusammenfassend lassen sich folgende Funktionen von Dozentenannotationen aufführen:

- Fokussieren Aufmerksamkeit der Studenten
- Erlauben dynamisches Erstellen von Skizzen
- Reagieren auf Unterrichtssituation
- Fixieren Studentenbeiträge und Arbeitsergebnisse

Außerdem gilt:

- Öffentliche Annotationen bedingen keinesfalls „Frontalunterricht“.

Da sich die Erscheinungsformen von Annotationen bei Dozenten prinzipiell kaum von denen der Studentischen unterscheiden, wird dieser Teil hier nicht extra thematisiert, sondern auf Kap. 2.2. verwiesen. Der Dozent hat demgegenüber aber eine größere Auswahl in den Medien, die ihm zur Erstellung von Annotationen zur Verfügung stehen. Der Student ist hier meist stärker festgelegt.

2.4.2. Medien für Dozentenannotationen

Alle Formen von Dozentenannotationen haben gemeinsam, dass ihr Inhalt in der Regel allen Studenten zur Verfügung steht. Im Gegensatz zu den Studentenannotationen sind sie also nicht privat, sondern öffentlich. Dies wird durch den Einsatz der im folgenden besprochenen Medien erreicht.

2.4.2.1. Kreidetafel

„Ich möchte die Tafel nicht gerade als Kultgegenstand bezeichnen. Aber eines ist klar: Sie erst macht einen Raum zum Klassenzimmer, sei es nun Baracke oder Kohlenkeller. Sie erst verleiht der Lehrperson die Aura des Weisen und Erhabenen.“ (Brenneisen, W. (1988))

Das wohl am ehesten mit dem Berufsstand Lehrer oder Dozent verbundene Utensil ist die Kreidetafel. Zentral vor den Studierenden für jeden einsehbar angeordnet, erlaubt sie das Erstellen von öffentlichen Annotationen. Prinzipiell ermöglicht die Kreidetafel bei gleicher Farbanzahl durch unterschiedlichen Druck und Kreidehaltung mehr Farbnuancen als zum Beispiel das Whiteboard. Auch flächenfüllendes Färben ist mit der Kreidetafel durch Flachlegen der Kreide mit vertretbarem Aufwand zu bewerkstelligen (vgl. Zirk, 1969).

Die Kreidetafel bietet (vgl. Grüner, 1961, Maras, 1979):

- Flächenfüllung
- Bereiche, die stehen bleiben können
- Einfache Bedienung
- Hohe Ausfallsicherheit
- Dozent bleibt im Vordergrund

Als Nachteil ist zu werten:

- Bei Platzmangel müssen Teile endgültig gelöscht werden
- Subjektives Unbehagen durch Kreidestaub, zum Beispiel an den Händen
- Großschreiben erfordert Gewöhnung
- Anschrieb ist wegen des beschränkten Platzes nur bedingt vorzubereiten
- Dozent muß sich zum Notieren abwenden
- Notwendigkeit von Wasser

2.4.2.2. Whiteboard

In neuerer Zeit findet das Whiteboard als Ersatz für die Kreidetafel immer mehr Verbreitung. Das zum Auswischen der Tafel notwendige Wasser, die Verschmutzung durch Kreidestaub und Wischwasser sind Argumente gegen eine Kreidetafel und für ein Whiteboard. Die Vorteile eines Whiteboards sind:

- Bereiche, die stehen bleiben können
- Einfache Bedienung
- Hohe Ausfallsicherheit
- Dozent bleibt im Vordergrund

Als Nachteil ist zu werten:

- Bei Platzmangel müssen Teile endgültig gelöscht werden.
- Großschreiben erfordert Gewöhnung.
- Anschrieb ist wegen des beschränkten Platzes nur bedingt vorzubereiten.
- Dozent muss sich zum Notieren abwenden.
- Wenige Farbnuancen praktikabel möglich.

2.4.2.3. Overheadprojektor

Der Overheadprojektor, in den siebziger Jahren als neues Medium in der Lehre eingeführt (vgl. Verein zur Förderung Berufspädagogischer Forschung e. V., 1968), hat inzwischen eine weite Verbreitung gefunden. Beim Overheadprojektor wird wie auf ein Stück Papier auf eine

Folie geschrieben. Dabei kann es sich um einzelne Folienseiten oder eine Endlosfolie handeln. Mehrere Folien können auch übereinander gelegt werden. Dies ermöglicht ein Vorbereiten von Folien. Außerdem erlaubt es das Ergänzen des Inhalts einer Folie durch das Überlagern mit einer weiteren Folie. Beim Overheadprojektor kann eine Skizze demnach entsprechend der Kreidetafel durch neu Zeichnen entstehen oder aber sie entwickelt sich durch das Überlagern von vorbereiteten Folien.

Der Overheadprojektor als Medium für Dozentenannotationen bietet also:

- Materialien können vorbereitet werden.
- Kein endgültiges Wegwischen notwendig
- Auch vorbereitete Sequenzen möglich
- Normalgroße Schrift möglich, da Vergrößerung durch Optik einstellbar
- Dozent bleibt den Studenten zugewandt.

Nachteilig ist zu sehen:

- Ausfallrisiko
- Dozent meist hinter dem Gerät
- Oft wackelnde Anzeige während des Schreibens
- Schattenwurf durch den Dozenten

2.4.2.4. Computer mit Videobeamer und Stifteingabe

Eines der wohl jüngsten Medien in der Lehre stellt sicher der Computer mit Anschluss an einen Videobeamer dar. Der Videobeamer dient dazu, ähnlich einem Overheadprojektor, das Monitorbild des Computers an die Wand zu projizieren und damit für alle Studierende im Raum sichtbar zu machen. Der Computer beziehungsweise die Bedienungsschnittstelle dafür befindet sich dabei getrennt von der Projektionseinrichtung direkt beim Dozenten. Ein Schattenwurf des Dozenten kann damit recht einfach vermieden werden. Als Bedienungsschnittstelle bietet der Computer standardmäßig eine Tastatur und eine Maus. Außerdem stehen seit einiger Zeit auch Schnittstellen zur Verfügung, die eine Stifteingabe ermöglichen. Dazu zählen:

- Interaktive Whiteboards (vgl. z.B. SmartBoard von SmartTech, 2005): Kunststofftafel ähnlich einem Whiteboard; das Monitorbild wird per Videobeamer auf die Tafel projiziert; Bewegung von speziellen Stiften auf der Tafel wird direkt an den Computer gemeldet; somit können beispielsweise Linien gezeichnet werden.
- Tablet (vgl. z.B. Volito von Wacom, 2005): Ein Tablet stellt eine kleinformatige Platte dar, auf der ein Spezialstift bewegt werden kann. Auch hier werden die Bewegungen direkt an den Computer weitergemeldet. Das Monitorbild des Dozenten ist an der Platte nicht sichtbar, wird aber für alle Studenten sichtbar über einen Videobeamer an die Wand projiziert.
- Tablet mit integriertem Monitor (vgl. z.B. Cintiq von Wacom, 2005): Diese Variante entspricht dem Tablet, nur dass nicht auf einer matten Platte, sondern auf einem Flachbild-

schirm, der den Inhalt des Computermonitors wiedergibt, gezeichnet wird. Das Bild wird zusätzlich für die Studenten über einen Videobeamer projiziert.

- Tablet-PC: Ein Tablet-PC vereint alle Teile in einem Gerät. Es handelt sich um einen kompakten Rechner mit integriertem Flachbildschirm, auf dem ein Spezialstift bewegt werden kann. Diese Bewegungen werden an den PC weitergegeben.

Je nach Stifteingabegerät entspricht der Computer mit Videobeamer mehr einem Tafelersatz (vgl. Interaktives Whiteboard) oder mehr einem Ersatz für einen Overheadprojektor (vgl. Tablet mit integriertem Monitor und Tablet-PC).

Der Computer mit Videobeamer und Stifteingabe hat folgende positive Merkmale:

- Materialien können vorbereitet werden.
- Kein endgültiges Wegwischen notwendig
- Auch vorbereitete Sequenzen möglich
- Normalgroße Schrift möglich, da Vergrößerung durch Optik einstellbar
- Funktionalität durch Software beliebig erweiterbar
- Automatische Aufzeichnung der Annotationen möglich
- Automatische Verteilung der erstellten Annotationen möglich

Nachteilig ist zu sehen:

- Ausfallrisiko
- Berührungsängste
- Vorkenntnisse notwendig

(vgl. zu beiden Aufzählungen auch Schulte, 2003)

2.4.3. Folgerungen für Dozentenannotationen

Wie Studentenannotationen sollten Dozentenannotationen ohne Einschränkung der Kreativität möglich sein. Nur so hat der Dozent die nötige Freiheit in der Gestaltung seiner Unterrichtseinheit. Wird diese Freiheit beschränkt, ist mit Akzeptanzproblemen zu rechnen.

Um sich möglichst vollständig auf die Sitzung konzentrieren zu können, muss das Medium für die Dozentenannotation möglichst einfach und intuitiv zu bedienen sein. Das Ausfallrisiko sollte sehr gering sein. Somit ist ein robustes System zur Erstellung der Annotationen notwendig. Dies ist speziell bei Systemen der letzten Kategorie, die aus Hard- und Software bestehen, zu berücksichtigen. Eine elektronische Unterstützung von Dozentenannotationen ist auf natürliche Weise am einfachsten in der letzten Kategorie von Medien zu verwirklichen. Als Vorteil kann gewertet werden, dass inzwischen verschiedenartige Geräte zur Verfügung stehen, die die klassischen Medien für Dozentenannotationen nachahmen können.

2.5. Zusammenfassung

Dieses Kapitel hat die vielseitigen Aspekte von Annotationen in der klassischen Lehre beleuchtet. Die Sinnhaftigkeit einer Beschäftigung mit Annotationen auch im Umfeld der elektronischen Lehre ist somit begründet.

Aus dem zuvor in diesem Kapitel Gesagten lassen sich aber auch schon einige Forderungen an Lehrsysteme, die Annotationen sinnvoll zur Effektivitätssteigerung der Lehre verwenden wollen, stellen:

- Möglichkeit der Studentenannotation
- Möglichkeit der Dozentenannotation
- Keine Einschränkung der Kreativität

Bisher wurde vorgestellt, was eine Annotation ist, wozu sie dient und in welchen Formen sie auftritt. Für ein tiefes Verständnis von Annotationen reicht dies aber noch nicht aus. Es muss des weiteren untersucht werden, wie Annotationen in verschiedene Prozesse der Lehre mit hineinspielen. Erst dann ist es möglich, Systeme in Bezug auf Annotationen zu bewerten beziehungsweise Systeme mit einer umfassenden Integration von Annotationen zu entwerfen.

3. Annotationen in Prozessen der Lehre

3.1. Prozess der klassischen Lehre

Der Lernprozess aus Sicht des Studenten unterteilt sich in der klassischen Lehre in zwei grundsätzliche Phasen: dem Unterricht und dem Nacharbeiten.

Für den Unterricht wird in der klassischen Lehre meist ein Multiplikatoreffekt genutzt: Ein Dozent vermittelt vielen Studenten sein Wissen. Eine der Urformen davon ist der sogenannte Frontalunterricht. Dabei vermittelt der Dozent sein Wissen jedem einzelnen Studenten, aber allen gleichzeitig. Vom Geräuschpegel abgesehen, ist es dabei egal, ob der Dozent einen Studenten oder tausend Studenten vor sich hat. Jeder Student hört den gleichen Kurs und arbeitet für sich. Die anderen Studenten interessieren ihn während des Kurses kaum. Ein Interagieren zwischen den Studenten wird meist sogar vom Dozenten, der aufkommenden Unruhe wegen unterbunden („schwätzen"). Sollte ein Student den roten Faden des Kurses verlieren, schafft er es auf sich alleine gestellt meist nicht, den Anschluss während der Unterrichtseinheit wieder zu finden. Der Dozent kann bei dieser Unterrichtsform nur bedingt auf einzelne Studenten eingehen und somit in diesem Fall kaum weiter helfen. Die weitere Teilnahme an der Unterrichtseinheit beschränkt sich auf Mitschreiben ohne tiefergehendes Verständnis.

Nach der Unterrichtseinheit folgt zu einem späteren Zeitpunkt das Wiederholen des Stoffes. In der Urform kann auch hier von einer Isolierung der Studenten ausgegangen werden. Jeder Student wiederholt den Stoff für sich alleine, anhand des Kursskripts mit seinen eigenen privaten Annotationen.

Aus Erfahrungen der klassischen Lehre weiß man, dass eine Interaktion zwischen den Studenten den Lerneffekt positiv beeinflusst (vgl. z.B. Slavin, 1990). In der ersten Phase im klassischen Szenario läßt sich eine solche Interaktion schwer verwirklichen, da eine direkte Interaktion den Gesamtablauf stört. Die Studenten werden aber meist angeregt, über den Dozenten durch Zwischenfragen an ihn miteinander zu kommunizieren. Auch kann durch Gruppen- oder Nachbararbeit Zeit für einen Austausch vorgesehen werden (vgl. Abb. 8).

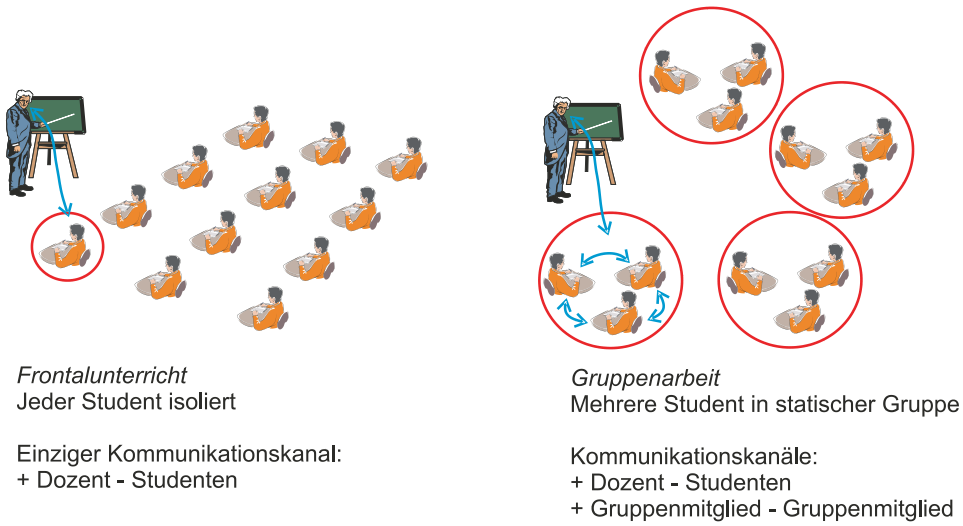


Abb. 8: Unterrichtsszenarien

Diese Möglichkeit besteht in der klassischen Lehre *während* des Kurses nur zeitweise. Die zweite Phase im klassischen Szenario ist dagegen offen für Interaktion. Lerngruppen sind ein gutes Beispiel dafür. Trotzdem wird diese Möglichkeit nicht voll ausgeschöpft. Oft krankt es an der örtlichen und zeitlichen Nähe der Studenten, da es in diesem Szenario ohne elektronische Hilfsmittel erforderlich ist, sich zu treffen.

3.2. Vision

Für die folgende Betrachtung sei angenommen, dass alle Möglichkeiten der Annotationen im klassischen Umfeld in einem elektronischen System, welches auch das Kursskript bereit stellt, schon verwirklicht seien. Darauf aufbauend wird nun visionär über dann mögliche Prozesse nachgedacht. Die sich ergebenden Ideen werden in den folgenden Abschnitten auf ihre technische Machbarkeit hin untersucht und Anforderungen an technische Systeme abgeleitet (Methodisches Vorgehen nach Walt Disney Technik, vgl. Dilts, Dilts & Epstein, 1991).

Die privaten Annotationen der einzelnen Studenten können, wenn sie elektronisch vorliegen, zeitnah an andere erreichbare Stellen gesendet werden. Dazu kann das Vermittlungsmedium mit genutzt werden, welches in der Definition dieser Arbeit für elektronisch unterstützte Lehre postuliert wurde.

Damit ist es dann technisch möglich, jedem Studenten die Annotationen der anderen Studenten zugänglich zu machen. Sollte ein Student den roten Faden während eines Kurses verlieren, wäre es ihm eventuell ohne Störung anderer eigenständig möglich, den Anschluss durch das Betrachten der Annotationen anderer wieder zu erlangen. Dies kommt dem Blick ins Heft des Banknachbarn gleich.

Neben dem Anzeigen anderer Annotationen ist es auch keine große technische Herausforderung, Annotationen anderer per einfachem Klick in das eigene Kursskript zu kopieren. Auf eine didaktische Bewertung des Kopierens per Klick wird hier nicht weiter eingegangen, da nur Visionen aufgezeigt werden.

Wie in Kapitel 2. erläutert, sind Annotationen aus Sicht der Lehre hilfreich. Dennoch nutzen Studierende Annotationen nicht im bestmöglichen Maß. Die Randbedingungen, wie der Entwicklungsstand, die Vorbildung, das Themengebiet und persönliche Vorlieben der Studierenden und Dozenten bedingen verschiedene Erscheinungsformen für Annotationen. Dies liegt hauptsächlich daran, dass der Mensch gerade im Bereich Annotationen selten aus eigenem Antrieb von Grund auf eigene Annotationsformen entwickelt. Meist werden Annotationsformen, die einmal bei jemand anderem gesehen wurden, eventuell leicht adaptiert, übernommen. Es zeigt sich, dass Annotationen auch in der Anwendung einfach benutzt werden, ohne sich vorher genauere Gedanken über die Konzepte dahinter zu machen. Es fehlt, wie heute des öfteren bemängelt, das Lernen zu Lernen (vgl. z.B. Norman, 1980). Untersuchungen belegen, dass Personen die an einer Schulungsmaßnahme zum Annotieren teilgenommen haben, besser, sprich mit höherem Mehrnutzen annotieren (vgl. hierzu zusammenfassend Caverly & Orlando, 1991). Es ist also erst eine Ausbildung im effizienten Einsatz von Annotationen, speziell im Zusammenhang mit Lernstrategien notwendig. Leider findet in heutiger Zeit kaum eine Ausbildung in Lernmethoden statt, auch wenn dies des öfteren gefordert wird. Können nun elektronische Annotationen von anderen Studenten eingesehen werden, läßt sich ein Lernen am Beispiel erhoffen. Ein prinzipielles Interesse der Studenten an den Annotationen anderer lässt sich aus der für diese Arbeit durchgeführten Umfrage ableiten. Demnach sehen 146 von 384 Studenten (38%) den größten Vorteil an elektronischen An-

notationen in der Möglichkeit Annotationen anderer zu durchsuchen (vgl. Schütz, 2005a, Frage 8).

Stehen die Annotationen an einer zentralen Stelle zu Verfügung, könnte auch der Dozent oder Tutor einen Blick darauf werfen. Stellen diese dabei einen Fehler fest, können die Studenten, die diese Annotation benutzen, wieder mit Hilfe einer Annotation darüber informiert werden. Dies entspräche technisch einem Annotieren von Annotationen. So wird ein gemeinsames Weiterentwickeln von Annotationen ermöglicht.

Als weiterer Punkt läßt sich durch statistische Auswertungen über die elektronisch erreichbaren Annotationen automatisch ein Feedback für den Dozenten generieren. Dieses Feedback kann dann für eine Verbesserung des Kurses genutzt werden.

Neben diesen technischen Möglichkeiten, die alle prinzipiell auf der Verteilung der Annotationen über einen elektronischen Kanal basieren, können die elektronisch vorhandenen Annotationen auch für sich alleine genutzt werden. So ist es möglich, dass sich ein Student nur die Annotationen und nicht das Kursskript anzeigen lässt. Somit hat er eine zweite, kürzere Quelle für seine Nachbereitung des Stoffes. Außerdem können Such- und Filtermechanismen für Annotationen implementiert werden. Wie die oben erwähnte Umfrage auch gezeigt hat, denken 150 von 384 (39%) Studenten, der größte Vorteil von elektronisch verfügbaren Annotationen ist, die elektronische Suchmöglichkeit in den *eigenen* Annotationen (vgl. Schütz, 2005a, Frage 8).

Nachdem nun Visionen zu elektronisch verfügbaren Annotationen aufgezählt wurden, wird anhand der Betrachtung einiger dann möglicher Prozesse, die Sinnhaftigkeit der elektronischen Erweiterungen untersucht. Daraus lassen sich Bedingungen und Richtlinien für die tatsächliche Umsetzung in ein reales Computersystem ableiten.

3.3. Resultierende Prozesse und ihre Bewertung

Für die Betrachtungen in diesem Abschnitt wird vorausgesetzt, dass das Kursskript in elektronischer Form vorliegt und der Student auch während des Kurses die Möglichkeit hat, darauf zuzugreifen. Mit der steigenden Verbreitung von Notebooks und Wireless LAN, vor allem in technischen Studiengängen, ist diese Annahme nicht zu gewagt. Projekte wie „Notebook-University“ zeigen ebenfalls in diese Richtung (vgl. BMBF, 2004). Konkret gaben an der Informatik der TU München 476 von 854 befragten Studenten im WS 2004/05 an, für die Übungen regelmäßig auf ein Notebook zurückgreifen zu können (Schütz, 2004). Dies entspricht immerhin knapp 55,7%.

Am Ende dieses Kapitels steht ein Kriterienkatalog zur Verfügung, anhand dessen die Qualität der Einbindung von Annotationen in Umgebungen zur elektronischen Unterstützung von Lehre bewertet werden kann. Jedes Kriterium erhält dabei für die Referenzierung in der weiteren Arbeit eine Kennnummer (z.B. *K1*). Einige Kriterien werden ihrer gemeinsamen technischen Umsetzbarkeit wegen später zusammengefasst. Dies ist bereits beim Einführen solcher Kriterien durch die angehängte Indizierung über Kleinbuchstaben angezeigt (z.B. *K2a* und *K2b*).

3.3.1. Private Annotation

Zuallererst sollte ein System zur Unterstützung von Annotationen zumindest die Möglichkeiten der klassischen Lehre bieten. Deshalb wird damit begonnen, die Unterstützung von privaten Annotationen zu betrachtet.

Drei Dinge sind, wie im folgenden begründet wird, unabdingbar, wenn Annotationen im elektronischen Umfeld akzeptiert werden sollen:

- K1a: Kein Medienbruch bei privater Annotation
- K2a: Lokalität der privaten Annotation
- K3a: Keine Einschränkung der Kreativität bei privater Annotation
- K4: Private Annotationen sind zu jedem Zeitpunkt möglich

Wie sieht heute oft das Lernen am Computer aus? Ein Blatt Papier liegt neben der Tastatur, auf dem Notizen und Annotationen zu den gelehrten Inhalten gemacht werden. Wird der Stoff wiederholt, muss der Zettel hervorgesucht werden. Die klassische Lehre kennt das Problem kaum. Dort kann der Student seine Annotationen direkt ins Kursskript machen. Er notiert sie dort, wo er sie auf natürliche Art und Weise beim Wiederholen auch findet. Auf die Frage hin, was sie davon halten, als Student direkt elektronisch in ein elektronisches Kursskript Notizen einfügen zu können, sagten 77 von 392 Studenten (19,6%), sie würden weiterhin ihre Notizen auf Papier machen, aber 185 Studenten (47%) gaben an, dass ein elektronisches System dann überhaupt erst Sinn macht (vgl. Schütz, 2005a, Frage 3). Zu ähnlichen Ergebnissen kommen auch Truong, Abowd & Brotherton (1999).

In der klassischen Lehre tauchen Annotationen zudem genau an der Stelle auf, die sie näher beschreiben. Dies wird als Lokalität der Annotation bezeichnet. Der Dozent in der klassischen Lehre wird deshalb auch angehalten, genügend freien Platz für Studentenannotationen bereit zu stellen (vgl. z.B. McAndrew, 1983). Nur 18 Studenten von 391 Studenten (4%) sehen keinen Vorteil darin, wenn Dozentenannotationen eine direkte Verbindung zum Kursskriptsystem aufbauen. 236 Studenten (60%) sehen es als nettes Feature an, wohingegen 137 Studenten (35%) klar äußern, dass elektronische Dozentenannotationen erst dann überhaupt Sinn machen (vgl. Schütz, 2005a, Frage 2). Die Lokalität der Annotationen ist also aus Studentensicht ein Thema, auch wenn sich die Frage auf Dozentenannotationen bezog.

Ein weiteres wichtiges Merkmal von Annotationen in der klassischen Lehre ist, dass die Kreativität des Studenten im Bezug auf seine Annotationen nicht eingeschränkt wird. Mit Hilfe des Stiftes kann der Student jeden im adäquat erscheinenden Linienzug als Annotation verwenden. Die farbliche Gestaltung ist ihm freigestellt.
Eine Einschränkung auf Texteingaben in der elektronischen Lehre, die dann obendrein meist mittels Tastatur zu erstellen sind, schließt von vorne herein den Einsatz gewisser Techniken bei Lernstrategien, wie zum Beispiel MindMaps aus (vgl. Buzan, 1993).

Sind die oben genannten Punkte erfüllt, bietet das System die Möglichkeiten der privaten Annotationen in der klassischen Lehre.

3.3.2. Öffentliche Annotation

Neben den privaten Annotationen gibt es in der klassischen Lehre auch öffentliche Annotationen. Darunter fallen alle Dinge, die der Dozent oder ein Student während der Unterrichts-

einheit an ein für alle einsehbares Medium notiert. Solche Medien sind zum Beispiel eine Tafel oder ein Overheadprojektor (vgl. 2.4.2.).

In der klassischen Lehre erlaubt die öffentliche Annotation an der Tafel eine unbeschränkte Freiheit in der Kreativität. Soll die elektronische Unterstützung sinnvoll einsetzbar sein und damit akzeptiert werden, ist dies bei der großen Anzahl an Lehrmethoden und Vorlieben der Lehrenden unbedingt notwendig. So haben beispielsweise Dozenten, die ein spezielles elektronisches System nicht einsetzen, dies darüber begründet, dass das System momentan nur für eine elektronische Tafel, nicht aber für ein Tablet - also konzeptuell einem elektronischen Overheadprojektor - zur Verfügung steht. Eine Tafel entspricht aber nicht ihrer Arbeitsweise (vgl. Schulte, 2003).

Die anderen beiden Kriterien der privaten Annotation, nämlich die Lokalität der Annotation und die Forderung nach keinem Medienbruch sind bei öffentlichen Annotationen in der klassischen Lehre nicht erfüllt oder zumindest kaum erfüllt. Kann man sich noch darüber streiten, ob es sich um einen Medienbruch handelt, wenn das Kursskript während der Sitzung auch als Folien gezeigt wird und während der Unterrichtseinheit etwas auf diesen notiert wird, so ist die Lokalität der Annotation nicht zu halten. Würden das Kursskript mit den in der Unterrichtseinheit gemachten Annotationen auch kopiert und unter den Studenten verteilt werden, so hätte der Student doch zwei Skripte: Das eine, welches er privat annotiert hat und das Skript mit den öffentlichen Annotationen. In der klassischen Lehre sind die beiden unvereinbar.

Mit Hilfe einer elektronischen Unterstützung kann hier Abhilfe geschaffen werden. Sind das Kursskript, die privaten Annotationen und die öffentlichen Annotationen in elektronischer Form greifbar, können sie in ein Gesamtdokument vereint werden. In der elektronischen Form sind vom Prinzip her bei öffentlichen Annotationen also auch die Forderungen nach Vermeidung eines Medienbruchs und nach Lokalität der Annotationen erfüllbar.

In der klassischen Lehre bietet sich jederzeit die Möglichkeit, dass auch ein Student eine öffentliche Annotation vornimmt. Dazu benutzt er einfach das öffentliche Medium, meist nach Aufforderung durch den Dozenten. Die Möglichkeit öffentliche Annotationen zu verfassen liegt also nicht exklusiv beim Dozenten. Auch die elektronische Lehre sollte dies berücksichtigen.

Da in der klassischen Lehre die Lokalität der privaten und öffentlichen Annotationen nicht vereinbar sind und zumeist noch ein Medienbruch vorliegt, behelfen sich die Studenten wie folgt: Sie kopieren öffentliche Annotationen in ihr privates Skript, zum Beispiel durch Abschreiben des Tafelanschriebs. Obwohl dieses Verhalten aus einem Mangel des klassischen Systems entstanden ist, der im elektronischen Fall vermieden werden kann, sollte die Möglichkeit trotzdem in einer elektronisch unterstützten Form mit angeboten werden. Der Student trifft damit nämlich eine Vorauswahl, welche Annotation er für nützlich erachtet und welche nicht.
Die elektronische Variante erlaubt ein Kopieren durch einen einfachen Knopfdruck. Pädagogisch mag dies zweifelhaft erscheinen, da erwiesenermaßen Dinge, die selbst notiert werden, besser verinnerlicht werden (vgl. Vester, 1978). Andererseits ist der Student nicht so lange mit dem Kopieren beschäftigt und kann schneller dem weiteren Unterrichtsgeschehen folgen. Da beide Alternativen technisch gesehen prinzipiell erhalten bleiben, wird der Frage nach pädagogischer Sinnhaftigkeit in diesem Fall nicht nachgegangen.

Bei der elektronischen Unterstützung von öffentlichen Annotationen sind also folgende Kriterien zu erfüllen:

- K1b: Kein Medienbruch bei öffentlichen Annotationen
- K2b: Lokalität der öffentlichen Annotation
- K3b: Keine Einschränkung der Kreativität bei öffentlichen Annotationen
- K5: Öffentliche Annotationen von jedem machbar
- K6b: Übernehmen von öffentlichen Annotationen

Sind diese Punkte erfüllt, bietet das System die Möglichkeiten der öffentlichen Annotationen in der klassischen Lehre und darüber hinaus erste Erweiterungen.

3.3.3. Gruppenannotation

Mit Gruppenannotationen wird ein Gebiet betreten, welches in der klassischen Lehre nur äußerst bedingt genutzt wird. Unter Gruppenannotation wird verstanden, dass eine Gruppe von Studenten die Annotation eines Gruppenmitglieds übernehmen. Eine öffentliche Annotation ist insofern ein Spezialfall von Gruppenannotationen, da hier *alle* Studenten eine Annotation nutzen.

In der klassischen Lehre entstehen Gruppenannotationen meist, wenn der eine Banknachbar sieht, dass der andere eine private Annotation macht und diese dann einfach übernimmt. Motivation für solches Kopieren ist, dass der Student entweder selbst zuerst die Wichtigkeit dieser Stelle übersehen hat (vgl. Kiewra, 1985) oder auch damit zusammenhängend, dass er gerade den Faden verloren hat und zwecks augenblicklich mangelndem Verständnis einfach alles sammelt. Eine zweite Gelegenheit, bei der Gruppenannotationen in der klassischen Lehre entstehen, sind Lerngruppen beim Nacharbeiten des Stoffes. Auch hier fallen oft noch einige übersehene Knackpunkte auf, welche dann als private Annotation ins Skript aller Gruppenmitglieder übernommen werden (vgl. O'Donnell & Dansereau, 1993).

Gruppenannotationen haben sehr viel mit Vertrauen, Kompetenz und Unsicherheit zu tun. Ist ein Student bei einem Sachverhalt des Unterrichtsstoffs unsicher, wird er sich oftmals zuerst an einen anderen Studenten, in der Fachterminologie auch als Peer (Ebenbürtiger) bezeichnet, wenden. Es könnte ja sein, dass es sich um eine Kleinigkeit handelt, für die er sich beim Nachfragen vor der ganzen Klasse schämen würde. Er kann die Sache aus seiner Unsicherheit heraus ja nicht abschätzen. Um auch diesen Studenten nicht zu sehr zu stören, wird er zuerst zum Beispiel durch einen kurzen Blick auf das andere Studentenskript prüfen, ob und welche privaten Annotationen dieser Student gemacht hat.

Der unsichere Student wird dazu einen Studenten fragen, der seiner Meinung nach kompetent ist. Außerdem wird es ein Student sein, dem er vertraut und der seiner Meinung nach die gleiche „Wellenlänge" hat. So ist einigermaßen gesichert, dass der gefragte Student, falls es notwendig war, eine private Annotation gemacht hat. Es ist also ersichtlich, dass jeder Student individuell andere Studenten fragen wird. Andererseits wird sich mancher Student auch durch Einblicke in seine privaten Annotationen belästigt fühlen und sich weigern, jedem Auskunft zu geben. Der befragte Student muss also dem fragenden Studenten auch eine Antwort geben wollen.

Hier können Gemeinsamkeiten mit Communities of practise erkannt werden (vgl. Lave & Wenger, 1991). Bei Communities of practise müssen auch Kompetenzen zueinander finden.

Vertrauen spielt in Communities eine sehr große Rolle. Die Literatur zu Communities bietet schon einen reichen Satz an Mechanismen und Forderungen. Ein für Gruppenannotationen interessanter Mechanismen sind zum Beispiel *Buddylisten*. Hier kann jeder Student zum Einen angeben, wem er Einblick in die eigenen Annotationen gibt und zum Anderen, wessen Annotationen ihn interessieren würden. Durch Abgleich dieser beiden Listen, entscheidet das Unterstützungssystem über die Zugriffsrechte jedes einzelnen.

Für eine elektronische Unterstützung von Gruppenannotationen ist es also wichtig, dass jeder Student festlegen kann, wer Einsicht in seine privaten Annotationen nehmen darf. Kann ein Student außerdem festlegen, wessen Annotationen ihn interessieren, kann das System proakitv, also ohne Aufforderung, diese privaten Annotationen anzeigen.

Da das Prinzip der Gruppenannotationen in der klassischen Lehre nicht sehr verbreitet ist, muss neben der Machbarkeit auch die Sinnhaftigkeit genauer beleuchtet werden. Sonst entsteht die oftmals von Pädagogen kritisierte technische Lösung um der Technik willen.

Gruppenannotationen erlauben es, nachdem sich die jeweiligen Studentengruppe gefunden hat, anonym innerhalb der Gruppe die Annotationen eines anderen Studenten zu betrachten. Dabei wird kein anderer Student belästigt oder in seiner Aufmerksamkeit gestört. Sollte ein Student während des Kurses den roten Faden verloren haben, kann er ihn damit eventuell schnell wiederfinden und so gewinnbringend den Rest des Kurses mitdenken und nicht einfach nur unverstanden mitschreiben.

Untersuchungen haben ergeben, dass in anderen Kulturkreisen eine offene Diskussion mit dem Dozenten unüblich ist. Dies geht soweit, dass die Studenten keine Fragen an den Dozenten richten und dem Dozenten auch kaum Antwort auf Fragen geben. Die Studenten wirken verglichen mit Studenten aus anderen Kulturkreisen schüchtern und zurückhaltend. In einer dieser Untersuchungen konnte gezeigt werden, dass eine anonymisierende Computerplattform dieser Kommunikationsarmut entgegenwirkt (vgl. Litosseliti, Marttunen, Laurinen & Salminen, 2002). Der Dozent erhielt mehr Feedback und die Studenten wurden selbstbewusster in ihrer Meinung, da sie durch die Plattform erkannten, dass auch andere Studenten den gleichen Gedanken hatten. Übertragen auf Gruppenannotationen würde das heißen: Alleine das Erkennen, dass andere Studenten auch eine private Annotation gemacht haben, reicht aus, um eine motivationssteigernde Bestätigung des eigenen Gedanken zu erhalten. Arbeitet das System in dem Sinne proaktiv, dass es eine Benachrichtigung generiert, wenn andere Studenten der Gruppe eine Annotation machen, werden solche „Bestätigungen“ automatisch generiert. Eine solche Benachrichtigung kann zum Beispiel durch eine kleine, stilisierte Leuchte, die aufleuchtet, sobald eine private Annotation innerhalb der Gruppe notiert wird, umgesetzt werden.

Welche Gefahren und Nachteile gehen von Gruppenannotationen aus?
Hier können sicherlich zwei Hauptgefahren genannt werden: Zum Einen können Fehler verbreitet werden. Zum Anderen kann es zu einer Informationsüberflutung kommen.

Gruppenannotationen stellen einen zusätzlichen Kommunikationskanal dar. Dieser Kanal geht am Dozenten vorbei. Sollte also eine falsche private Annotation verfasst worden sein, würde sich diese über den Mechanismus der Gruppenannotationen verbreiten. Andererseits kann sie aber auch durch einen Hinweis aus der Gruppe korrigiert werden.

Auch in der klassischen Lehre kommt es vor, dass ein Student einen Sachverhalt missverstanden hat und trotzdem eine Reihe seiner Kommilitonen zum Beispiel beim Nachbereiten davon überzeugen kann. Das Problem an und für sich besteht also auch in der klassischen Leh-

re. Der Unterschied liegt vielmehr in der Geschwindigkeit der Ausbreitung. In der klassischen Lehre werden solche Dinge meist erst diskutiert. Diese Möglichkeit ist aber in der vorgeschlagenen Version von Gruppenannotationen nicht gegeben. Andere Studenten sollen ja nicht gestört werden.

Das Problem wird zudem auf natürliche Weise durch die Selektion der Studenten, von denen Annotationen angezeigt werden sollen, entschärft. Mit der Zeit wird die Kompetenz eines Studenten klarer. In der Regel wird ein Student nur Annotationen von Studenten, die er zumindest gleich kompetent einschätzt, betrachten. Communities kennen für die Ermittlung und Darstellung der Kompetenz eines Mitglieds einige Mechanismen. Im Fall der Gruppenannotationen könnte bei der Wahl der Studenten zum Beispiel angezeigt werden, wie viele andere Studenten sich schon für die Benachrichtigung eingetragen haben und wie viele Annotationen tatsächlich übernommen wurden. Durch das so stattfindende Herausarbeiten von Experten lässt sich das Problem der Übernahme von falschen Annotationen einschränken, aber sicher nicht beheben. Wird eine Annotation übernommen, sollte für später zumindest ersichtlich bleiben, dass es sich um eine übernommene Annotation handelt und von wem sie übernommen wurde.

Sollte ein Student eine größere Anzahl von Studenten angegeben haben, bei denen er über neue private Annotationen benachrichtigt werden will, kann eine erhebliche Anzahl von Benachrichtigungen entstehen. Mit dieser Informationsflut konfrontiert, bleiben dem Studenten zwei Alternativen: Er ignoriert die Hinweise. Dann haben Gruppenannotationen aber auch keinen Sinn. Oder er betrachtet alle Hinweise und verpasst dabei den eigentlichen Unterricht. Das hier angesprochene Problem hat jede neue Technik: Auf die Art der Benutzung kommt es an. Das wahre Potential kann erst bei richtiger Benutzung entfaltet werden. Dafür braucht es aber eine gewisse Einarbeitungszeit. Nach dieser wird wohl jeder Student sein individuelles Maß für benachrichtigende Studenten gefunden haben. Filtermechanismen können das Problem in der Praxis mildern. Die gewinnbringende Nutzung läuft aber ebenfalls auf einen überlegten Einsatz hinaus.

Wägt man nun die Vor- und Nachteile gegeneinander ab, kann folgender Schluss gezogen werden: Wird der Kurs real und stark interaktiv gestaltet, sprich gibt es während des Kurses vom Wesen her mehr eine Diskussion über den Stoff als Frontalunterricht, so wird der Zusatznutzen des Mechanismus Gruppenannotation gering ausfallen. Meist hängt damit in realen Kursen auch direkt die Anzahl der Studenten zusammen. Je mehr Studenten in einem realen Kurs sitzen, desto mehr werden den Kurs eher passiv miterleben.

Handelt es sich aber zum Beispiel der großen Studentenanzahl oder der Mentalität wegen um Frontalunterricht, bildlich „Einbahnstraßenunterricht“, oder um einen virtuellen Kurs, bietet der Mechanismus Gruppenannotationen mit seinen „Querstraßen“ sicher eine nicht zu unterschätzende Unterstützung des Unterrichtsgeschehens. Eine Übernahme des Mechanismus Gruppenannotation in eine elektronische Unterstützung von Annotationen ist somit auch aus pädagogischen Gründen gerechtfertigt.

In der Nachbereitungsphase eines Kurses haben Gruppenannotationen aus den selben Gründen ihre Berechtigung.

Für eine konsequente elektronische Unterstützung von Gruppenannotationen sind folgende Kriterien zu erfüllen:

- K6a: Übernehmen von Gruppenannotationen

- K7a: Student kann festlegen, wer Annotationen sehen darf.
- K7b: Student kann festlegen, wessen Annotationen er sehen möchte.
- K8a: Filtermechanismen / Kompetenzdarstellung
- K9: Hinweis, wenn neue Annotation erstellt wurde.
- K10: Quellennachweis beim Übernehmen einer Gruppenannotation

3.3.4. Kollaborative Annotation

Eine kollaborative Annotation ist eine Annotation, an der mehrere Personen (Studierende, Dozent) gemeinschaftlich arbeiten. Im Gegensatz zu den bisherigen Annotationen ist eine kollaborative Annotation deshalb in der Regel nie abgeschlossen. Ändert eine Person etwas an einer solchen Annotation, wird diese Annotation auch bei den anderen geändert. Sie stellt somit ein Mittel dar, welches anderen erlaubt, direkt in anderen angezeigten Annotationen zu manipulieren.

In der klassischen Lehre entsteht etwas ähnliches während einer Gruppenarbeit unter gewissen Aufgabenstellungen. Eine mögliche solche Aufgabenstellung ist: „Fassen Sie in der Gruppe den Text <xy> zusammen und präsentieren Sie anschließend ihr Ergebnis.“ Die Zusammenfassung stellt in unserem Sinn eine Annotation dar. Die Annotation entsteht durch Zusammenarbeit mehrerer Studenten während der Bearbeitungszeit in der Gruppe. Die Annotation wird zum Gesamtergebnis der Gruppe.

Trotz dieses Beispiels bleibt die Anwendung von kollaborativen Annotationen in der klassischen Lehre wegen der bedingten Machbarkeit äußerst beschränkt. Es fehlt das echt gemeinsam nutzbare Medium. Anders ist die Situation in der elektronisch unterstützten Lehre. Hier ist es technisch - wie in der kollaborativen Gruppenarbeit zigfach durch konkrete Implementierungen belegt (vgl. z.B. *Tivoli*, Pederson, McCall, Moran & Halasz, 1993 oder *MediaBoard*, Tung, 1997) - kein Problem, einen Bereich zu definieren, auf dem alle Gruppenmitglieder gemeinsam annotieren können. Die Kollaboration geht dabei soweit, dass ein Student den Beitrag eines anderen Studierenden löschen kann.

Neben der Erstellung einer kollaborativen Annotation zum Zweck der Steigerung der Verständlichkeit des Kursskripts kann eine kollaborative Annotation auch zur Gruppenkommunikation genutzt werden. Alle Gruppenmitglieder können darauf zugreifen und dürfen eigene Beiträge beisteuern. So kann das eine Gruppenmitglied eine Frage an die Gruppe formulieren. Ein anderer antwortet oder aber es entsteht eine eigene Diskussion. Über kollaborative Annotationen lassen sich also Peer Groups unterstützen (vgl. Iles, Glaser, Kam & Canny, 2002).

Für eine Unterstützung von kollaborativen Annotationen muss ein elektronisches System folgende Kriterien erfüllen:

- K11: Annotationen können als kollaborativ markiert werden.
- K12: Personen können in eine kollaborative Gruppe eingeladen werden.

3.3.5. Feedback

Für eine erfolgreiche Lehre ist Feedback über die Qualität des Kurses notwendig. In der klassischen Lehre erhält der Dozent ein gewisses Feedback durch die Leistungen, die vom Stu-

denten erbracht werden müssen. Dazu zählen Hausarbeiten, „Vorrechnen an der Tafel“ und Klausuren. Liegen Annotationen in elektronischer Form vor, ist es möglich, auch aus diesen Rückschlüsse über den Kurs zu ziehen. Dabei können vier verschiedene Arten von Feedback festgehalten werden:

- Fehlerkorrekturen
- Umfragen
- Statistische Auswertungen
- Inhalt der Annotationen

Alle genannten Formen von Feedback können über elektronische Annotationen verwirklicht werden. Dies wird im folgenden genauer dargestellt.

3.3.5.1. Fehlerkorrektur

Wie bereits erwähnt, kann der Dozent das Medium Öffentliche Annotationen verwenden, um einen Fehler im Kursskript zu berichtigen. Wird in der elektronischen Unterstützung von Öffentlichen Annotationen das Markieren von Annotationen als Fehlerkorrektur unterstützt, ergeben sich weitere Möglichkeiten. Zum Einen kann diese Annotation dann automatisch, das heißt, ohne dass der Student aktiv werden muss, in jedes Studentenskript kopiert werden. Zum Anderen kann der Dozent beim Nachbereiten des Kurses oder beim Vorbereiten eines nächsten Kurses diese Korrekturen sofort einarbeiten. Konzeptuell gesehen kann der Dozent sich also selbst während des Kurses Feedback geben.

Um dies zu unterstützen, muss das elektronische Annotationssystem folgende Kriterien erfüllen:

- K13: Markieren von öffentlichen Annotationen als Fehlerkorrektur
- K8b: Suchen nach Markierungen von öffentlichen Annotationen

3.3.5.2. Umfragen, Tests

In einem elektronischen Lehrumfeld kann jeder Student seine eigene Meinung zu einem Sachverhalt der Lehrumgebung „mitteilen“, ohne dass andere in der Lerngruppe erfahren, wer diese Meinung vertritt. Ein interessantes Mittel für die Einbeziehung in das Lehrgeschehen bieten dabei Umfragen. Der Dozent kann an beliebigen Stellen in seinem Skript kleine Fragebögen unterbringen. Die gestellten Fragen beziehen sich dabei auf den Inhalt oder aber auf den Kurs im allgemeinen. Technisch erlaubt es das elektronische Medium, eine Auswertung eines solchen Fragebogens schnell und ohne Zeitverlust durchzuführen (vgl. z.B. *TVRemote*, RBG TU Darmstadt, 2005).

In Bezug auf Annotationen kann dies heißen, dass zu einer Frage eine Antwort aus mehreren Möglichen durch Ankreuzen (Annotieren) in einem Feld oder durch ein eigenes Annotationswerkzeug gewählt werden kann. Wenn das System die Möglichkeit bietet, anzugeben, wie viele Studenten eine Annotationen zu einer Antwort gemacht haben, liegt sofort ein Umfrageergebnis vor.

Ähnlich einer Umfrage, kann der Dozent auch kurze Tests in sein Kursskript einbinden und damit den Wissensstand seiner Studenten abfragen. Technisch gibt es hier keinen Unterschied zu Umfragen. Der Unterschied liegt in den konkreten Fragen und Antworten. Wäh-

rend in Umfragen Feedback zum Kurs oder zur Sitzung (zum Beispiel Geschwindigkeit im Fortgang und Umfang des Stoffes aus Sicht der Studenten) generiert wird, gehen Tests inhaltlich auf den Lehrstoff ein.

Nutzt der Dozent Tests, um zeitnah Feedback über den Wissensstand der Studenten zu erhalten und haben Studenten auch Zugriff auf die Ergebnisse der Tests, fördert dies den Lehrfortschritt (vgl. Moallem, Kermani & Chen, 2002).

Damit ein elektronisches Unterstützungssystem Umfragen und Tests über Annotationen erlaubt, müssen folgende Punkte gegeben sein:

- K14: Spezielle Bereiche für Umfrageannotationen können definiert werden.
- K15a: Häufigkeit von Annotationen in den jeweiligen Bereichen werden ausgegeben.

3.3.5.3. Statistische Auswertungen

Statistische Auswertungen über das Annotierverhalten lassen Rückschlüsse auf die Studenten während des Kurses zu. Dabei haben statistische Auswertungen ein entscheidendes Merkmal: Durch die Komprimierung der Daten sind die auszuwertenden Informationen anonymisiert und daher aus Sicht des Datenschutzes zumeist unbedenklich. Die Anonymisierung kommt auch dem Vertrauen der Studenten entgegen.

Welche Daten sollen als Basis für die statistische Auswertung dienen?
Die Frage lässt sich einfacher beantworten, wenn man angibt, was nicht genutzt werden kann. Der Inhalt der Annotationen bleibt für die statistische Auswertung komplett außen vor. Die Kreativität beim Erstellen der Annotationen soll, wie oben ausgeführt, nicht eingeschränkt werden. Damit sind alle Formen von Zeichnungen und Text möglich. Diese Vielfalt erlaubt es aber nicht, die Semantik der jeweiligen Annotation maschinell zu erfassen. Somit kann allgemein über den Inhalt der Annotationen auch keine Statistik erstellt werden. Auf den Inhalt mancher spezieller Erscheinungsformen von Annotationen können zwar statistische Auswertungen angewandt werden (z.B. eingetippter Text). Die Aussagekraft ist aber anzuzweifeln, da andere Studenten ähnliche Annotationen auch mit Freihandwerkzeugen erstellen können, deren Annotationsergebnis dann nicht in die Statistik mit eingeht.

Basis für statistische Auswertungen können der Zeitpunkt, der Ort des Einfügens und die Erscheinungsform einer Annotation im Skript sein. Mit Hilfe dieser Daten lassen sich dann zum Beispiel Stellen im Skript erkennen, an denen viele Studenten Annotationen gemacht haben. Berücksichtigt man dabei auch den Zeitpunkt, erkennt man, ob die Annotationen beim Nacharbeiten oder während des Kurses entstanden.

Der Dozent kann nun vergleichen, ob die Schwerpunkte beim Annotieren an den Stellen lagen, an denen er es erwartet oder sogar provoziert hat. Stellen, die nicht mit seinen Erwartungen übereinstimmen, sollte er genauer überprüfen. Eventuell muss dann die jeweilige Unterrichtseinheit angepasst werden.

Mit einer allgemeinen Bewertung dieser Statistiken muss vorsichtig umgegangen werden. Nur der Dozent selber weiß, ob er einen Annotationsschwerpunkt, eventuell aus didaktischen Gründen provoziert hat, oder ob diese Häufung auf ungenügende Verständlichkeit des Skripts an dieser Stelle zurückzuführen ist. Aussagen wie: „Je weniger Annotationen, desto besser der Unterricht“ sind im allgemeinen nicht machbar.

Mit Hilfe der zeitlichen Verteilung der Annotationen kann man erkennen, ob sich das Annotierverhalten im Laufe des Kurses verändert hat. Werden immer weniger Annotationen gemacht, könnte dies auf eine nachlassende Motivation bei den Studenten schließen lassen oder auf gestiegene Verständlichkeit des Skriptes, da zum Beispiel nach der erforderlichen Einführung in das neue Stoffgebiet durch im weiteren Verlauf mögliche Querbezüge einiges klarer wird. Wie schon bei den Auswertungen zum Ort der Annotation gesagt, ist eine solche Bewertung aber immer in Relation zu den Zielen des Dozenten zu sehen.

Insgesamt kann zusammengefasst werden, dass statistische Auswertungen über Annotationen einen ersten Überblick über das Skript erlauben. Dieser erste Eindruck muss unbedingt mit den Dozentenzielen abgeglichen werden. Erst dann kann man daraus kritische Stellen extrahieren, die eventuell eine Nachbearbeitung erfordern.

Bietet ein elektronisches System zur Unterstützung der elektronischen Lehre diese Auswertungen an, hat der Dozent eine zusätzliche Quelle für Feedback zu seinem Kurs, welche in der klassischen Lehre nicht zur Verfügung steht. In der heutigen Zeit, in der auch in der Lehre immer mehr die Effizienz in den Vordergrund rückt, sollte jede Quelle zur Bewertung der eigenen Effizienz genutzt werden. Also ist eine Umsetzung von Annotationen in diese Richtung gerechtfertigt.

Um solche statistischen Auswertungen zu unterstützen, muss das elektronische Annotationssystem folgende Kriterien erfüllen:

- K15b: Liefern von kumulierten Zeitinformationen zu Annotationen
- K15c: Liefern von kumulierten Ortsinformationen zu Annotationen
- K15d: Liefern von kumulierten Typinformationen zu Annotationen

3.3.5.4. Inhalt der Annotationen

Für eine abgerundete Betreuung der Studenten wäre es wünschenswert, wenn der Dozent die gemachten Annotationen durchsehen würde. So könnte er Fehler und Missverständnisse sofort aufdecken (vgl. Kiewra, 2002). Außerdem hätte er durch das Betrachten der Annotationen ein relativ zeitnahes Feedback zu seinem Kurs.

Neben dem sicher nicht unerheblichen zusätzlichen Zeitaufwand steht dem der Schutz der Privatsphäre der Studenten entgegen. Einerseits könnten Studenten eine Bewertung ihrer Leistung anhand ihrer Annotationen befürchten und dann das System sicherheitshalber erst gar nicht benutzen. Außerdem könnten in den Aufzeichnungen auch Annotationen auftauchen, die nicht direkt etwas mit dem Kurs zu tun haben und nicht für die Augen des Dozenten bestimmt sind. Hier muss ein Mechanismus vorgesehen werden, dass der Student schon während des Anlegens der Annotation bestimmen kann, ob diese vom Dozenten einsehbar sein soll. Damit wird aber die Eingabe einer Annotation aufwendiger, was die Akzeptanz des Systems herabsetzt. Als alternative Lösung kann erlaubt werden, dass der Dozent in die Buddyliste für Gruppenannotationen mit aufgenommen werden kann.

Die oben angesprochene Korrektur der Annotationen ist nur bis zu einem gewissen Grad machbar. Jeder Student kann der uneingeschränkten Kreativität wegen seinen eigenen Stil für Annotationen entwickeln. Dabei entstehen Formen und Abkürzungen, die eine andere Person, hier der Dozent, schwerlich in der ursprünglich beabsichtigen Art und Weise interpretiert.

Insgesamt gesehen überwiegen bei der Nutzung des Inhalts der Annotationen als Feedback die Nachteile. Nichtsdestotrotz kann mit entsprechendem Zeitaufwand des Dozenten und der Einhaltung einiger Konventionen bei den Studenten sicher auch ein Mehrwert für alle Beteiligten generiert werden.

Das System müßte folgendes Kriterium unterstützen:

- K7c: Anzeigen von Studentenannotationen gegenüber dem Dozenten

3.4. Zusammenfassung als Kriterienkatalog

Nachdem nun die Begründung von Annotationen in einigen Prozessen der Lehre gegeben wurde, Kriterien für die Benutzung in diesen Prozessen aufgestellt wurden und außerdem im nächsten Schritt bestehende Systeme zur elektronischen Unterstützung der Lehre auf ihre Fähigkeiten bezüglich Annotationen geprüft werden sollen, folgt hier eine tabellarische Zusammenfassung der erarbeiteten Kriterien (vgl. Tab. 2).

Damit die Wichtigkeit eines Kriteriums innerhalb des Kriterienkatalogs besser abgeschätzt werden kann, wird innerhalb der Tabelle jeweils die Relevanz für ausgesuchte Aspekte angegeben.

Unter dem Aspekt *Akzeptanz Dozent* wird zum Ausdruck gebracht, dass dieses Kriterium erfüllt sein muss, wenn Dozenten mit dem Annotationssystem arbeiten sollen. Ist dieses Kriterium nicht erfüllt, wird es zu Akzeptanzproblemen kommen, die einen erfolgreichen Einsatz von vorne herein fraglich erscheinen lassen. Das gleiche gilt für die *Akzeptanz Student*.

Mit dem Aspekt *Gewinn Dozent* soll angedeutet werden, dass die Erfüllung des entsprechenden Kriteriums einen Vorteil für den Dozenten gegenüber einer klassischen Lehrumgebung bietet. Gerade diese Punkte sind zu betonen, wenn es darum geht, ein solches System einzuführen. Sie stellen den Mehrwert für den Dozenten dar. Dieses gilt sinngemäß auch für *Gewinn Student*.

Die Zuordnung der jeweiligen Relevanz erfolgt aufgrund eigener informeller Beobachtungen, aufgrund einer eigenen Umfrage (vgl. Schütz, 2005a) und nicht zuletzt aus den Erfahrungen der in Kapitel 4.1 beschriebenen Systeme.

Die Wahrscheinlichkeit ein System zur elektronischen Unterstützung der Lehre mit Annotationsmöglichkeit erfolgreich einzuführen, ist somit gut, wenn alle für die Akzeptanz notwendigen Punkte erfüllt sind und darüber hinaus die meisten der als Gewinn markierten.

Für den weiteren Gebrauch wurde der Kriterienkatalog außerdem in fünf Unterbereiche gegliedert: Pflicht, öffentliche Annotation, private Annotation, Gruppenannotation, Auswerten

Kriterien im Bereich Pflicht müssen für eine sinnvolle Annotationsunterstützung immer erfüllt werden. Alle anderen Unterbereiche geben an, welche Kriterien für eine Erfüllung der Ansprüche dieses Bereichs gegeben sein müssen.

	Kriterium	Akz. Dozent	Akz. Student	Gew. Dozent	Gew. Student
Pflicht					
K0	Verbindung Annotation zu Skript		X		
öffentliche Annotation					
K1b	Kein Medienbruch bei öffentlichen Annotationen	X	X		
K2b	Lokalität der öffentlichen Annotation			X	X
K3b	Keine Einschränkung der Kreativität bei öffentl. Annot.	X			
K5	Jeder kann öffentliche Annotationen machen				X
K6b	Übernehmen von öffentlichen Annotationen				X
private Annotation					
K1a	Kein Medienbruch bei privaten Annotationen		X		
K2a	Lokalität der privaten Annotation		X		
K3a	Keine Einschränkung der Kreativität bei privaten Annot.		X		
K4	Private Annotationen sind immer möglich		X		
Gruppenannotation					
K6a	Übernehmen von Gruppenannotationen				X
K7a	Student legt fest, wer Annotationen sehen darf		X		
K7b	Student legt fest, wessen Annotationen er sehen will				X
K7c	Dem Dozenten Studentenannotationen anzeigen			X	
K9	Hinweis, wenn neue Annotation gemacht wurde				X
K10	Quellenangabe bei übernommenen Gruppenannotation				X
K11	Annotationen können als kollaborativ markiert werden				X
K12	Pers. können in kollaborative Gr. aufgenommen werden				X
Auswerten					
K8a	Filtermechanismen / Kompetenzdarstellung				X
K8b	Suchen nach Markierungen von öffentlichen Annot.			X	
K13	Markieren von öffentlichen Annot. als Fehlerkorrektur			X	X
K14	Umfrageannotationen oder spez. Bereiche dafür def.			X	X
K15a	Häufigkeiten von Umfrageannot./Annot. in spez. Bereich			X	X
K15b	Liefern von kumulierten Zeitinformationen zu Annot.			X	
K15c	Liefern von kumulierten Ortsinformationen zu Annot.			X	
K15d	Liefern von kumulierten Typinformationen zu Annot.			X	

Tab. 2: Bewertungskriterien für Lehrsysteme mit Unterstützung von Annotationen

4. Annotationen in bestehenden Anwendungen

4.1. Lehrsysteme

Mit dem in 3.4. erstellten Kriterienkatalog sollen nun bestehende Systeme zur Unterstützung der elektronischen Lehre auf Unterstützung von Annotationen hin untersucht werden. Dabei wurde die Auswahl der Systeme auf solche beschränkt, die schon in irgendeiner Form Annotationen von Studenten oder Dozenten vorsehen. Außerdem wurde weiterhin auf einige repräsentative Systeme eingeschränkt. Für eine umfangreichere Liste von Systemen sei auf Müller & Ottmann (2002) verwiesen.

Im folgenden werden die ausgewählten Projekte kurz vorgestellt und ihre Integration von Annotationen betrachtet. Im Anschluss daran wird eine zusammenfassende Tabelle mit der Bewertung der Kriterien angegeben.

4.1.1. ed.tec

Die Gruppe um Herrn Prof. Sebastian Abeck hat an der Universität Karlsruhe mit ed.tec einen Vorschlag für ein Framework für ein eLearning-System entwickelt (vgl. Feuerhelm, Bonn & Abeck, 2002). Dabei wird im speziellen den verschiedenen beteiligten Rollen und deren Workflow Rechnung getragen. So ist das System in eine Autoren- und eine Dozentenumgebung aufgeteilt. In der Autorenumgebung werden die möglichen Lehrinhalte definiert und zusammengestellt. In der Dozentenumgebung stehen die Lehrinhalte dann zur Präsentation zur Verfügung. Außerdem bietet die Dozentenumgebung die Möglichkeit, ein Mikrofon und ein elektronisches Tablett zu unterstützen. Damit kann der Dozent seine präsentierten Lehrinhalte annotieren. Außerdem kann er vom Framework automatisiert eine Aufzeichnung seines Kurses erstellen lassen. Diese Aufzeichnung kann synchron an andere Orte übertragen oder asynchron den Studenten zum Nacharbeiten zur Verfügung gestellt werden.

Am Beispiel des Kurses „Verteilte Informationssysteme“ (Abeck, S., 2004) wird nun die Einbindung von Annotationen genauer betrachtet.

Während des Kurses präsentiert der Dozent das Kursskript über die Komponente ed.teach. Dabei kann er mit einem elektronischen Tablett Freihandannotationen erstellen. Diese werden archiviert. Nach einem Kurs stehen einem Studenten somit folgende Unterlagen zum Kurs zur Verfügung: Einmal das Kursskript als HTML-Seiten ohne Annotationen. Dann die PowerPoint-Folien im PDF-Format mit und ohne Annotationen. Außerdem die PowerPoint-Präsentation des Dozenten ohne Kurssannotationen (vgl. Abb. 9).

Im Framework von ed.tec wird dem Studenten kein eigenes Modul zugewiesen. Ein aktives Arbeiten mit den Inhalten ist (noch) nicht vorgesehen. Natürlich können auch einzelne interaktive Multimediainhalte (z.B. Macromedia Flash Animationen) angeboten werden. Diese sind aber nicht grundsätzlich im Framework verankert. Damit sieht das System von sich aus auch zum jetzigen Zeitpunkt keine Unterstützung für Studentenannotationen vor. Daraus folgt wiederum, dass das Framework auch keine kollaborative Mechanismen für Studentenannotationen anbietet. Einzig wenn der Student das PDF-Dokument als Variante für sein

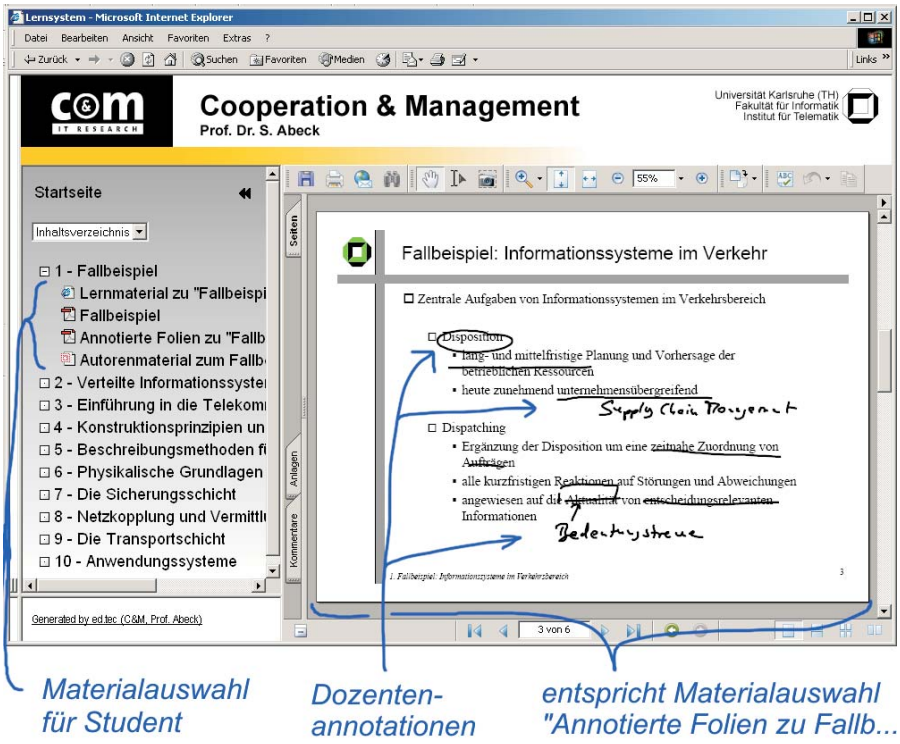


Abb. 9: Studentensicht auf einen Kurs in der ed.tec-Umgebung (vgl. Abeck, 2004)

Kursskript wählt, kann er mit Hilfe des kostenpflichtigen Acrobat Readers eigene Annotationen direkt im Kursskript erstellen. Dies erfolgt dann aber außerhalb des ed.tec Rahmenwerks.

Was die angebotenen Dozentenannotationen angeht, schränken diese durch Verwendung von Freihandeingabe und einem Tablett kaum die Kreativität des Dozenten ein. Durch die direkte Anzeige der annotierten Kursskriptseiten ist auch die Lokalität der Annotationen gegeben. Über ed.tec können wie oben beschrieben mehrere Medien für die Darstellung des Kursskripts genutzt werden. Darunter befindet sich auch eine im PDF-Format, welche direkt die Dozentenannotationen enthält. Ein Medienbruch liegt bei dessen Verwendung also nicht vor.

Die Annotationen tauchen im Framework und im PDF-Dokument nicht als eigenständige Objekte auf. Somit stehen dafür also auch keine Filter-, Sortier- oder Verarbeitungsfunktionen zur Verfügung. Textannotationen sind nicht vorgesehen, sodass eine inhaltliche Weiterverarbeitung für keine Art von Annotationen zur Verfügung steht.

Damit kann ed.tec den Wunsch des Dozenten nach Erstellung von Annotationen während des Kurses erfüllen. Annotationen von Studenten und damit weitere Möglichkeiten bleiben aber zum jetzigen Zeitpunkt außen vor.

4.1.2. eCase

An der RWTH Aachen wurde am Lehrstuhl für Informatik VII im Rahmen des Projekts *ULI - Universitärer Lehrverband Informatik* mit *eCase* ein System entwickelt, welches es dem Dozenten erlaubt, mit sehr geringem Zusatzaufwand einen Kurs aufzuzeichnen und anschließend den Studenten zur Verfügung zu stellen (vgl. Rhode & Thomas, 2003 und eCase, 2004). Die erforderliche Technik befindet sich in einem transportablen Koffer und kann somit in un-

terschiedlichsten Lehrumgebungen eingesetzt werden (vgl. Abb. 10). Einzig ein Videobeamer befindet sich noch nicht im Inhalt des Koffers. Dem Dozenten steht ein Tablet mit integriertem LCD-Bildschirm zur Bedienung der Präsentation und für das Erstellen von Annotationen zur Verfügung. Es ist vorgesehen, dass während des Kurses die Anzeige des LCD-Bildschirms des Dozenten zusätzlich über einen Videobeamer für alle Studenten sichtbar projiziert wird.

Abb. 10: Der Koffer mit der Ausstattung zu eCase (Quelle: eCase, 2004).

Der Dozent hat die Möglichkeit, seinen Kurs aufzuzeichnen. Dabei entsteht eine Videosequenz, die genau den Inhalt des Dozentenbildschirms während des Kurses wiedergibt. Dabei treten durch die Verwendung eines speziellen Codecs von Camtasia keinerlei Qualitätsverluste auf (vgl. TechSmith, 2004). Zusätzlich wird als Audioanteil die Stimme des Dozenten integriert.

Im Anschluss an den Kurs steht den Studenten das Video mit dem Bildschirminhalt und der Stimme des Dozenten und das annotierte Kursskript im PDF-Format über das Internet zur Verfügung.

Bei der Entwicklung von eCase wurde besonderes Augenmerk darauf gerichtet, dem Dozenten eine Umgebung zu liefern, welche das Erstellen einer Onlinevariante eines Kurses ohne großen Zusatzaufwand ermöglicht und er seine vom Overheadprojektor her gewohnten Arbeitsweisen weitgehend beibehalten kann. Durch die Entscheidung, den kompletten Bildschirminhalt des Dozenten aufzuzeichnen, werden automatisch alle Dozentenannotationen, unabhängig von welchem Werkzeug sie erstellt werden, ins Video übernommen. Eine Verbindung zwischen Dozentenannotation und Kursskript besteht also automatisch über das Video.

Nutzt der Dozent, wie an der RWTH Aachen praktiziert, Acrobat 6.0 Professional zum Präsentieren des Kursskripts und auch direkt zum Erstellen der Annotationen, entstehen eigene Annotationsobjekte (vgl. 4.2.2.). Für diese Objekte können dann direkt Filter- und Sortiermechanismen von Acrobat benutzt werden. Über Erweiterungen, die für Acrobat angeboten werden, besteht auch die Möglichkeit, Annotationen über mehrere Benutzer zu verteilen und

kollaborativ zusammenzuarbeiten. Dafür ist dann aber zusätzlich bei allen beteiligten Studenten eine Version von Acrobat 6.0 Professional erforderlich.

4.1.3. Notebook University Tools

An der Technischen Universität Darmstadt sind unter dem Namen *Notebook University Tools* die Werkzeuge *OCLI*, *TVremote* und *ToGather* im Einsatz (vgl. RBG TU Darmstadt, 2004). Diese wurden von der Fachgruppe Telekooperation und der Rechnerbetriebsgruppe TU Darmstadt entwickelt. Im Gegensatz zu den zuvor vorgestellten Ansätzen, wird hier die Studentenseite und die Interaktion Student-Dozent stärker betont. Dies ist technisch möglich, weil in diesem Zusammenhang davon ausgegangen wird, dass jeder Student während des Kurses ein Notebook zur Verfügung hat.

Jeder Student kann so zum Beispiel mit TVremote während des Kurses Fragen an den Dozenten stellen. Außerdem kann der Dozent Multiple-Choice Fragen an die Studenten stellen. Die Ergebnisse lassen sich wegen der maschinellen Verarbeitung sofort darstellen (vgl. Rößling, Bär & Mühlhäuser, 2004).

Mit OCLI (Open Client Lecture Interaction) kann der Student in einem zweiachsigen Koordinatensystem eine Bewertung der aktuellen Situation im Kurs abgeben. In der einen Achse gibt er seine Bewertung für das Tempo des Kurses ab. In der anderen Achse trägt er die für sich empfundene Relevanz an. Der Dozent kann ständig eine Übersicht über die aktuellen Bewertungen einsehen (vgl. Trompler, Mühlhäuser & Wegner, 2002).

ToGather dient im Szenarium der Notebook University Tools als Werkzeug zur Aufzeichnung eines Kurses (vgl. Trompler, Rößling, Bär & Choi, 2003). Dabei kann der Dozent in Wort und Bild aufgenommen werden. Außerdem können Studenten eigene Annotationen im ebenfalls vorhandenen Kursskript erstellen und zusammen mit dem Kursskript ablegen. Die Dozentenannotationen werden über einen Broadcastingmechanismus während des Kurses verteilt und auch bei den angeschlossenen Studenten gespeichert.

Dieses System bietet eine etwas tiefergehende Unterstützung von Annotationen. Neben der Möglichkeit, eigene Annotationen zu erstellen, gibt es hier bereits Möglichkeiten, andere Annotationen, nämlich die Dozentenannotationen, automatisiert zu übernehmen. Eine erste Umsetzung von Gruppenmechanismen erfolgt in Gestalt der Multiple-Choice Fragen. Außerdem wird darüber und über die OCLI Feedback für den Dozenten erzeugt.

Tiefergehende Konzepte zur Unterstützung von Gruppen, wie zum Beispiel Buddylisten, sind integriert. Da die Annotationen momentan nicht als eigene Objekte greifbar sind, gibt es keine Filter- und Suchmechanismen, um Annotationen besser verwalten zu können.

4.1.4. LiveNotes

LiveNotes wurde von Matthew Kam entwickelt. Es handelt sich hierbei um ein Werkzeug zur Echtzeitkonversation über ein Wireless LAN innerhalb einer kleinen Gruppe. Die Konversation findet über einen gemeinsamen Zeichenbereich statt. Jeder Teilnehmer arbeitet dazu mit einem stiftbasierten Handheld-Computer und zeichnet Annotationen auf den Zeichenbereich. Die anderen Teilnehmer sehen diese gleichzeitig und ergänzen diese durch eigene Annotationen. In diesem Sinne entsteht eine Gruppenkonversation (vgl. Iles, Glaser, Kam & Canny, 2002).

Von LiveNotes werden nur wenige Mechanismen zur Koordination der Konversation vorgegeben. Einer ist, dass jeder Teilnehmer eine Zeichenfarbe zugewiesen bekommt. Ein anderer ist, dass beliebig viele neue Zeichenflächen angelegt werden können und jeder Teilnehmer auf eine beliebige Zeichenfläche navigieren kann. Außerdem können Zeichenflächen nach HTML exportiert werden.

Über ein Menü kann jeder Teilnehmer erkennen, wer sonst gerade in der Gruppe angemeldet ist und welche Zeichenfläche die anderen jeweils gerade betrachten.

LiveNotes wurde entwickelt, um Peer Dialogue während eines realen Kurses zu ermöglichen. Unter Peer Dialogue wird dabei verstanden, dass sich eine kleine Gruppe von ebenbürtigen Personen über den zeitgleich in der Vorlesung vermittelten Stoff unterhalten. Ebenbürtig meint hier speziell, dass keine Lehrer - Schüler, Dozent - Student oder Führer - Untergebener Konstellation vorherrscht, sondern sich Studenten mit Studenten unterhalten. Es gibt Experimente, die darauf hindeuten, dass Unterricht mit technisch unterstütztem Peer Dialogue zu besseren Lernergebnissen führt (vgl. zum Beispiel Distributed TVI experiment, Dutra, Gibbons, Pannoni, Sipusic, Smith & Sutherland, 1999).

LiveNotes ging in seiner Entwicklung von der Idee aus, einer Gruppe von Studenten über eine gemeinsame Zeichenfläche eine Konversation während eines Kurses zu ermöglichen. Während der Experimente mit dem System wurde erkannt, dass für eine effektive Gruppenarbeit gewisse Awareness-Funktionalität unbedingt notwendig ist (vgl. Kam, Tarshish, Glaser, Iles & Canny, 2002). So muss für jeden erkennbar sein, wer gerade das System benutzt und welche Seite die anderen jeweils gerade betrachten.

Zu Beginn dachten die Entwickler von LiveNotes an folgenden Prozess während eines Kurses: Einer der Studenten übernimmt die Rolle eines Protokollführers und erstellt ein Skript zum Kurs. Die anderen Studenten folgen dem Kurs und sehen das Protokoll ihres Kollegen. Ergeben sich Fragen oder haben sie Anmerkungen annotieren sie diese im Skript des Protokollführers. Dadurch kann weitere Kommunikation entstehen (vgl. Iles et.al., 2002).
Da sich nicht immer automatisch ein Protokollführer findet und oft auch schon ein Skript des Dozenten in elektronischer Weise vorliegt, kann das Dozentenskript nun in einer weiteren Verbesserung des Systems direkt als vorgefertigte Seiten importiert werden.

Mit der starken Fokussierung auf die Gruppenunterstützung, fehlt LiveNotes das Konzept der öffentlichen Annotation. Der Dozent kann somit zum Beispiel nicht über alle Gruppen hinweg eine Notiz machen.

Die konsequente Nutzung der Freihandnotizen und auch sonst der Verzicht auf strukturelle Vorgaben erlaubt ein hohes Maß an Kreativität und Flexibilität. Die praktischen Versuche ergaben, dass diese minimale Einschränkung wünschenswert ist. Das Fixieren der Farbe für einen Benutzer ist zwar verständlich, schränkt aber gerade beim Annotieren im Lernumfeld merklich ein.

Eine Annotation wird in LiveNotes nicht als eigenes Objekt betrachtet. Ein Sortieren, Suchen und Filtern ist somit nicht möglich.

Insgesamt belegen die Studien rund um LiveNotes, dass es Vorteile bringt, über einen technischen Kanal Kommunikation auch während eines Kurses zu erlauben (vgl. Kam, Wang, Iles, Tse, Chiu, Glaser, Tarshish & Canny, 2005). Die Rolle von Freihandnotizen und die Freiheit in der Prozessgestaltung wird durch die Ergebnisse im Zusammenhang mit LiveNotes gestärkt.

4.1.5. eClass

Das *eClass* Projekt (vormals *classroom 2000*) am Georgia Institute of Technology in Atlanta hat als Ziel, möglichst alle Eindrücke, die ein Student während eines realen Kurses aufnehmen kann, aufzuzeichnen und später als eigene virtuelle Lehrveranstaltung wieder anzubieten (vgl. Abowd, Brotherton & Bhalodia, 1998).

Annotationen wurden von Beginn an als notwendiges Konzept erkannt. Zuerst hatten Dozenten und Studenten gleichermaßen die Möglichkeit, während der Vorlesung Annotationen zu erstellen. Über Umfragen ergab sich interessanterweise, dass die Studenten, welche die meisten Annotationen machten, den geringsten Vorteil in solch einem System sahen. Weitere Untersuchungen ergaben, dass gerade diese Studenten stark dazu neigten, genau die Dozentenannotationen abzuschreiben, obwohl diese im Anschluss an die Sitzung sowieso in das Kursskript übernommen wurden. Aufgrund dieser Erfahrung stellte das Team um damals noch *classroom 2000* die Unterstützung von Studentenannotationen zeitweise zurück. Erst das erklärte Ziel personalisierte Kursunterlagen zu schaffen, rückte Studentenannotationen wieder aus dem Hintergrund heraus (vgl. Truong, Abowd & Brotherton, 1999).

eClass grenzt zwei Phasen stark gegeneinander ab: Das Aufzeichnen der Sitzung gegen das Benutzen der Aufzeichnung. eClass erlaubt es dem Studenten als Konsequenz der technischen Umsetzung nur, während der Aufzeichnungsphase eigene Annotationen intensiv mit dem Kursskript und den entstehenden Unterlagen zu verknüpfen. In der Phase des Benutzens kann der Student zwar Annotationen ergänzen. Diese können aber nur in einem getrennten Bereich pro Sitzung erstellt werden. Es gibt keine Möglichkeit, eine Verbindung zu Passagen im Kursskript herzustellen. Begründet wird dies damit, dass Annotationen außerhalb der Aufzeichnungsphase sowieso meist zusammenfassenden Charakter haben (vgl. Truong et.al., 1999).

Das eClass Projekt spezialisiert sich auf die Aufzeichnung möglichst vieler Eindrücke während einer Sitzung. Neben Video und Audio zählen auch Annotationen dazu. Die Eindrücke werden mit dem Ziel einer sehr guten Navigation in den entstehenden Artefakten möglichst intensiv miteinander verbunden.

Die entstandene Aufzeichnung der Sitzung dient als virtueller Ersatz für eine reale Sitzung. Eine über eine gut navigierbare Präsentation hinausgehende Unterstützung des Lernprozesses, speziell der Phase des Nacharbeitens, ist nicht vorgesehen. Auch die Zusammenarbeit zwischen Studenten eines Kurses wird nicht unterstützt.

Insgesamt scheint im zugrundeliegenden Modell die Rolle eines Kommilitonen nicht vorgesehen zu sein. Einzige vertretene Rollen sind der Dozent und der Student selbst. Eine über die technisch verbesserte Navigation in den Kursunterlagen hinausgehende Unterstützung des Lernprozesses wird nicht geboten. Insbesondere sind eigene Annotationen nicht als eigenständige Objekte vorhanden. Eine Filterung und Einschränkung ist somit auch nicht vorgesehen.

4.1.6. AOF - Authoring-on-the-fly (Universität Freiburg)

Unter dem Titel *Authoring on the fly* ist eine Reihe von Projekten zusammengefasst (vgl. Bacher, Müller, Ottmann & Will, 1997). Kern von Authoring-on-the-fly bildet ein Softwaresystem zur Aufzeichnung und Übertragung von Sitzungen. Bei diesem Projekt steht der Dozent im Mittelpunkt. Es soll ihm eine Möglichkeit an die Hand gegeben werden, ohne großen eige-

nen Aufwand eine Sitzung zu anderen Hörsälen oder zu Studenten zu übertragen oder aber für eine spätere Verwendung aufzeichnen zu können.

Authoring-on-the-fly bietet dafür die Möglichkeit, eine Sitzung über Video und Audio aufzuzeichnen. Außerdem werden die elektronisch präsentierten, vorbereiteten Folien des Dozenten und die in der Sitzung erstellten Annotationen archiviert. Alle so entstehenden Artefakte sind zeitlich synchronisiert. Der Student kann also zu einem späteren Zeitpunkt das Video zur Sitzung betrachten und erhält dazu zeitgenau die entsprechenden Folien und Annotationen eingeblendet. Die Benutzungsschnittstelle des Studenten bietet über das lineare Betrachten der Sitzung hinausgehende Mechanismen zum Wiederholen einer Sitzung. Der Student kann etwa über einen aus Miniaturansichten der Folien bestehenden Index zu Zeitpunkten in der Sitzung springen.

Erwähnenswert ist unter anderem im Bezug zu Annotationen die Möglichkeit des „visible scrollings“: Der Student kann über einen die Zeitlinie der Sitzung repräsentierenden Scrollbalken in der Aufzeichnung navigieren. Dabei werden Änderungen in den angezeigten Folien und Annotationen sofort entsprechend der aktuellen Position des Reglers angezeigt. Im Gegensatz zu anderen Anwendungen funktioniert dies auch während des Veränderns der Reglerposition. Der Student hat also die Möglichkeit den Erstellungsprozess von Annotationen im von ihm festgelegten Tempo zu wiederholen.

Da das System einzig auf die Aufzeichnung und Präsentation einer Sitzung aus Dozentensicht ausgerichtet ist, sind keine Mechanismen für Studentenannotationen integriert.

Die Herangehensweise ähnelt der des eClass- und des eCase-Systems. Insgesamt gehen deren Ansätze davon aus, dass eine Überprüfung der Sinnhaftigkeit aus pädagogischem Blickwinkel nicht notwendig ist, da eine möglichst vollständige, technische Aufzeichnung der Eindrücke während einer Sitzung angestrebt wird. Die Technik dient also der Konservierung der angewandten Pädagogik und Didaktik des Dozenten. Konsequenterweise sind die Systeme somit bemüht, den Dozenten in seiner Präsentation möglichst wenig einzuschränken. Die Pädagogik und Didaktik bleibt einzig in der Verantwortung des Dozenten.

Im Fall von Authoring-on-th-fly werden tatsächliche Lernprozesse der Studenten nicht weiter thematisiert. Eine Unterstützung von Annotationen für Studenten findet nicht statt. Das Authoring-on-the-fly zugrundeliegende Modell geht wie bei eClass nur von einem Studenten aus. Kommilitonen sind darin nicht enthalten. Deshalb werden auch keine Kommunikationskanäle zwischen Studenten thematisiert.

Aus Sicht der Annnotationen ist an Authoring-on-the-fly das „visible scrolling“ interessant. Der Student erhält hier die Möglichkeit, Dozentenannotationen wiederholt in beliebigem Tempo entstehen zu lassen. Zu Untersuchen wäre nun, inwiefern sich damit Annotationen besser im Gedächtnis verankern. Es geht also nicht um die Frage, ob Annotationen inhaltlich verständlicher werden, wenn die Reihenfolge der Entstehung bekannt ist. Es geht vielmehr darum, ob die zusätzlich entstehende Dimension der Bewegung weitere automatische und gewinnbringende Assoziationen erzeugt. Das Erinnern würde damit vereinfacht.

4.1.7. Bewertungen

Nach der Vorstellung der Systeme im Umfeld elektronisch unterstützter Lehre, werden diese nun anhand des in 3.4., S. 45 erarbeiteten Kriterienkatalogs gemäß ihrer Integration von Annotationen nach beurteilt (vgl. Tab. 3).

	Kriterium	ed.tec	eCase	NU tools	LiveNotes	eCLass	AOF
Pflicht							
K0	Verbindung Annotation zu Skript	+	+	+	+	+	+
öffentliche Annotation							
K1b	Kein Medienbruch bei öffentlichen Annotationen	+	+	+	o	+	+
K2b	Lokalität der öffentlichen Annotation	+	+	+	o	+	+
K3b	Keine Einschränkung der Kreativität bei öffentl. Annot.	+	+	+	o	+	+
K5	Jeder kann öffentliche Annotationen machen	-	-	o	o	-	-
K6b	Übernehmen von öffentlichen Annotationen	-	-	+	o	+	-
private Annotation							
K1a	Kein Medienbruch bei privaten Annotationen	o	o	+	o	+	-
K2a	Lokalität der privaten Annotation	o	o	+	o	+	-
K3a	Keine Einschränkung der Kreativität bei privaten Annot.	o	o	+	o	+	-
K4	Private Annotationen sind immer möglich	-	-	-	+	o	-
Gruppenannotation							
K6a	Übernehmen von Gruppenannotationen	-	o	-	+	-	-
K7a	Student legt fest, wer Annotationen sehen darf	-	-	-	+	-	-
K7b	Student legt fest, wessen Annotationen er sehen will	-	-	-	+	-	-
K7c	Dem Dozenten Studentenannotationen anzeigen	-	o	o	o	-	-
K9	Hinweis, wenn neue Annotation gemacht wurde	-	-	-	+	+	+
K10	Quellenangabe bei übernommenen Gruppenannotation	-	-	-	o	-	-
K11	Annotationen können als kollaborativ markiert werden	-	-	-	+	-	-
K12	Pers. können in kollaborative Gr. aufgenommen werden	-	-	-	o	-	-
Auswerten							
K8a	Filtermechanismen / Kompetenzdarstellung	-	o	-	-	-	-
K8b	Suchen nach Markierungen von öffentlichen Annot.	-	-	-	-	-	-
K13	Markieren von öffentlichen Annot. als Fehlerkorrektur	-	-	-	-	-	-
K14	Umfrageannotationen oder spez. Bereiche dafür def.	-	-	+	-	-	-
K15a	Häufigkeiten von Umfrageannot./Annot. in spez. Bereich	-	-	+	-	-	-
K15b	Liefern von kumulierten Zeitinformationen zu Annot.	-	-	-	-	-	-
K15c	Liefern von kumulierten Ortsinformationen zu Annot.	-	-	-	-	-	-
K15d	Liefern von kumulierten Typinformationen zu Annot.	-	-	-	-	-	-

Tab. 3: Bewertung ausgewählter Lehrsysteme nach dem Kriterienkatalog.

Folgende Prädikate wurden vergeben:

- \+ für *ja* beziehungsweise *vorhanden*
- **o** für *teilweise* beziehungsweise *mit Hilfe von Zusätzen*
- \- Für *nein* beziehungsweise *nicht vorhanden*

Beim Betrachten dieser Matrix ohne Fokus auf ein spezielles System fällt sofort ins Auge, dass die Bereiche *Pflicht* und *öffentliche Annotation* von den meisten Systemen gut abgedeckt sind. Im Bereich *private Annotation* sieht die Sache schon schwächer aus. Der Bereich *Gruppenannotation* ist dagegen kaum und der Bereich *Auswerten* praktisch gar nicht abgedeckt.

Dieser Umstand ist schlüssig zu erklären, wenn die Entwicklungsgeschichte der elektronischen Unterstützung von Lehre herangezogen wird: Das Nutzen und zur Verfügung stellen von elektronischen Kursskripten im aufstrebenden WWW, brachte einen Medienbruch mit sich. Dieser betraf den Dozenten wie den Studenten. So konnte etwa der PowerPoint nutzende Dozent sehr einfach sein Kursskript online anbieten, musste aber Annotationen an die Tafel schreiben. Der Student hingegen druckte sich das elektronische Kursskript aus und annotierte dort im Papier. Dieser unbefriedigende Umstand wurde sehr bald erkannt und über die Möglichkeit von elektronischen Annotationen behoben. Dabei wurde meist erst eine Annotationsmöglichkeit für die Dozenten entwickelt, da diese während der Präsentation kaum, wie Studenten, auf eine Papierlösung zurückgreifen konnten. Dass die Bereiche „Gruppenannotation“ und „Auswerten“ recht wenig bearbeitet wurden, liegt daran, dass sie Neuland bedeuten. Es gibt keine Entsprechung in der klassischen Lehre. Betrachtet man gerade die Ergebnisse der Systeme, die Teile dieser Bereiche umsetzen, ist ein vielversprechendes Nutzenpotential auszumachen. Sollen also Annotationen in innovativer und zukunftssicherer Weise in elektronische Systeme zur Unterstützung der Lehre integriert werden, so müssen gerade auch die letzten beiden Bereiche beachtet werden. Hier wird der echte Mehrwert der elektronischen Unterstützung generiert. Im Architekturvorschlag dieser Arbeit sind diese Teile explizit berücksichtigt.

4.2. Allgemeine Anwendungen

Neben der Anwendung in der Lehre tauchen elektronische Annotationen auch in anderen Geschäftsbereichen verstärkt auf. Nachdem im Bereich der elektronischen Lehre im vorhergehenden Abschnitt ein Defizit in der Umsetzung von Annotationen erkannt wurde, soll nun zum Einen überprüft werden, inwiefern allgemeine Anwendungen mit Annotationsunterstützung von sich aus brauchbar für die Lehre sind. Zum Anderen wird betrachtet, wie dort die Unterstützung von Annotationen speziell in den defizitären Bereichen der oben genannten Systeme aussieht.

Im folgenden werden drei allgemeine Vertreter betrachtet. Alle besitzen schon jetzt eine Relevanz in der elektronisch unterstützten Lehre. Sie können wie folgt charakterisiert werden:

Microsoft als hardwarezentrierter Vertreter wurde ausgewählt, da Microsoft im Kontext von Annotationen ein Betriebsystem für eine spezielle Hardware, dem Tablet-PC, entwickelt hat.

Adobe als produktzentrierter Vertreter wurde ausgesucht, da Adobe eine genau auf sein wenn auch verbreitetes Dokumentformat zugeschnittene Annotationsunterstützung bietet.

W3C wurde schließlich als konzeptzentrierter Vertreter gewählt, da es sich weder auf eine spezielle Umgebung noch ein spezielles Produkt, sondern auf ein Protokoll festlegt.

Jedes Annotationssystem wird in den folgenden Absätzen kurz vorgestellt und unter dem Gesichtspunkt der Tauglichkeit als elektronisches Systeme zur Unterstützung der Lehre bewertet. Dabei wird davon ausgegangen, dass das jeweilige Produkt im Kurs zur Darstellung des Kursskripts und zur Erstellung der Annotationen benutzt wird.

4.2.1. Microsoft Unterstützung für Tablet PC's

Ein Tablet-PC ist ein tragbarer Computer mit eingebautem Monitor. Als Eingabemedium dient klassisch ein Stift, mit dem direkt auf dem Monitor geschrieben und bedient wird. Vorstellung der Entwickler ist es, ein elektronisch unterstütztes Klemmbrett anzubieten.

Wieso soll hier der Tablet PC speziell betrachtet werden? Vom Prinzip her könnte der Tablet PC den Overheadprojektor ersetzen. Er bietet eine Fläche, auf der mit Hilfe des Stiftes wie auf einer Folie geschrieben werden kann. Über einen Videobeamer wird das Bild an die Wand geworfen und für anwesende Studenten sichtbar. Außerdem kann der angezeigte Bildschirm über das Netz zum Beispiel über Virtual Network Computing (VNC) anderen Studenten angezeigt werden. Vorgefertigte Folien, wie beim Overheadprojektor möglich, sind kein Problem.

Nachdem der technischen Machbarkeit von Tablet-PC's nichts mehr im Weg stand, waren Anwendungen notwendig, diese auch benutzbar zu machen. Als einer der führenden Betriebssystemehersteller hat Microsoft einige seiner Betriebssysteme auf die Anwendung von Tablet-PC's hin ergänzt und angepasst. Außerdem wurde in Produkte der Office-Reihe eine erweiterte Unterstützung von Stifteingabegeräten integriert (vgl. Microsoft, 2005).

Die aus der Nutzung der Microsoft Unterstützung für Tablet PC's resultierenden Möglichkeiten im Einsatz in der Lehre werden nun anhand einzelner Anwendungsprogramme beispielhaft erläutert.

4.2.1.1. Microsoft Word mit Unterstützung für Tablet PC's

Da Microsoft Word ein sehr verbreitetes Textverarbeitungsprogramm ist, ist es vorstellbar, dass Dozenten ihr Kursskript damit verfassen. Im Weiteren wird angenommen, dass das Kursskript als Worddokument dem Dozenten und den Studenten zur Verfügung steht. Außerdem wird angenommen, dass der Dozent während des Kurses direkt am Worddokument arbeitet. Folgende Unterstützungswerkzeuge für den Einsatz eines Tablet PC's stehen dann direkt in Word zur Verfügung:

- Freihandkommentar
- Freihandbereich

Für ein Freihandkommentar markiert der Benutzer einen Teil des Textes und öffnet über Einfügen - Freihandkommentar einen Kommentar mit Zeichenfläche am rechten Rand des Dokuments. Hier kann der Benutzer nun direkt mit Hilfe des Eingabestifts zeichnen.

Neben den Freihandkommentaren, die über die Erweiterung für Tablet PC's hinzugekommen sind, bietet Word die Möglichkeit von Textkommentaren, um ein Dokument zu annotieren. Sie unterscheiden sich von Freihandkommentaren nur im Kommentarbereich.

Zur Darstellung der Verbindung zu einer Textstelle dient eine Linie vom Textbereich zum Kommentar. Außerdem ergänzt Word den Autor und den Zeitpunkt bei jeder Art von Kommentar. Des Weiteren werden alle Arten von Kommentaren auch in einem speziellen Überarbeitungsfenster angezeigt. Dort kann mit Hilfe verschiedener Suchmöglichkeiten ein spezieller Kommentar gefunden werden.

Der Einsatz von Word Kommentaren durch den Dozenten während der Vorlesung ist nur in beschränktem Maße sinnvoll. Die starre Platzierung von Kommentaren am rechten Rand sorgt recht schnell für Platzmangel. Während dieser Effekt bei den kleineren Textkommentaren erst später auftritt, ergibt sich das Problem bei Freihandkommentaren, die mit einer größeren Standardgröße erscheinen, schon ab drei bis vier Annotationen. Ein jeweiliges Verkleinern des Zeichenbereichs ist möglich, bedeutet aber Zusatzaufwand für den Dozenten.
Da Kommentare immer auf einer zusätzlichen Fläche neben dem eigentlichen Dokument erscheinen, können damit Freiräume im Dokument nicht genutzt werden. Ein Ausweiten der Zeichenfläche in der Breite ist nur bedingt möglich: Die Fläche lässt sich zwar verbreitern, eine vollständige Anzeige wird aber nur gegeben, wenn in dem Freihandkommentar editiert wird. Ein Student kann dies während einer Vorlesung nicht anstoßen.

Die Verwendung von Word Kommentaren zum Annotieren bei Studenten ist eher vorstellbar, da hier oftmals eine Tastatur vorhanden ist und die Textkommentare wie oben erwähnt weniger Platz benötigen. Der überlegte Einsatz von Kommentaren in Kombination zur Nutzung des Überarbeitungsfensters ermöglicht die Unterstützung von Ordnungsstrategien. So lässt sich zum Beispiel bei entsprechender Annotationsstrategie über die automatisch entstehende Stichwortsammlung schnell ein Überblick über das Kursskript gewinnen.

Während Kommentare nur am rechten Rand des Dokuments erscheinen, kann mit dem zweiten oben aufgeführten Werkzeug, dem Einfügen eines Freihandbereichs, auch im Dokumentbereich annotiert werden. Dabei ist zu erwähnen, dass hiermit im Gegensatz zu Arbeiten mit Kommentaren auch direkt im Dokument editiert wird.

Wählt der Benutzer an beliebiger Stelle im Dokument Einfügen - Freihandbereich aus, so öffnet sich an dieser Stelle über die ganze Breite ein grafischer Eingabebereich. Dabei wird bei der Standardeinstellung des Layouts für dieses Objekt der komplette darunterliegende Text nach hinten verschoben. Nun kann im Freihandbereich mit dem Stift annotiert werden. Die Größe des Freihandbereichs lässt sich durch Griffe am Rand verändern.

Zum Einsatz in der Lehre kann für die Rolle des Dozenten folgendes gesagt werden: Ein schnelles Annotieren einzelner Textpassagen ist mit diesem Werkzeug während einer Vorlesung kaum möglich. Dazu ist es nämlich erforderlich, dass die Annotation den Textbereich nicht verändert. Nur so können auf Dauer korrekt referenzierende Annotationen erstellt werden. Bei Verschiebungen des Originaltexts besteht die Gefahr, dass solche Annotationen zum Beispiel bei Seitenumbrüchen nicht mehr korrekt zum referenzierten Text platziert werden. Durch Umstellung des Layouts für den Freihandbereichs kann zwar eine Darstellung im Hintergrund des Originaltexts erreicht werden, die Handhabung ist aber weiterhin unbefriedigend. Beim Einfügen werden automatisch vor dem Umstellen des Layouts schon zwei Zeilenschaltungen im Originaltext eingefügt, die erst wieder manuell entfernt werden müssen. Zum Anderen erfordert die Umstellung des Layouts einige Benutzerinteraktion, was im Umfeld der Lehre während eines Kurses zu sehr vom eigentlichen Geschehen ablenkt. Außerdem verschwindet die Anzeige des Originaltexts während der Bearbeitung des darunterliegenden Freihandbereichs. Ein Verweisen auf den Text zum Beispiel durch einen Pfeil oder ein Freihandunterstreichen sind somit kaum möglich.

Das Werkzeug Freihandbereich ist besser geeignet, wenn der Dozent an vorbereiteter Stelle Veranschaulichungen (vgl. 2.2.1.6.) erstellen möchte.

Insgesamt gilt das eben gesagte auch für die Rolle der Studenten. Der Freihandbereich ist wenig dazu geeignet, Annotationen im Text vorzunehmen.

Zusammenfassend ergibt sich die Gesamteinschätzung, das Microsoft Word als Mittel zur Repräsentation und zur Arbeit mit einem Kursskript, in Bezug auf Annotationen, nicht geeignet ist. Die Kommentarmechanismen entsprechen zwar grundsätzlich der Anforderung. Die fixe Platzierung am rechten Rand stellt aber eine recht starke Einschränkung der Möglichkeiten dar. Die Benutzung von Freihandbereichen im Text ist zu umständlich und gehört nicht zu den Kommentarmechanismen. Es wird damit also der eigentliche Text verändert und die Suchmöglichkeiten im Überarbeitungsfenster stehen nicht zur Verfügung.

4.2.1.2. Microsoft PowerPoint mit Unterstützung für Tablet PC's

Nachdem in 4.2.1.1. Microsoft Word als klassischer Vertreter für Dokumentverarbeitung mit Microsoft Produkten betrachtet wurde, wird nun das Augenmerk auf Microsoft Powerpoint gerichtet. Powerpoint stellt eine Präsentationssoftware dar, ist somit als Unterstützung für Dozenten während einer Vorlesung unmittelbar vorstellbar. Im folgenden wird angenommen, das Kursskript liegt dem Dozenten und den Studenten als Powerpoint-Präsentation vor. Nun sollen Aspekte zu Annotationen in Powerpoint betrachtet werden. Es wird zuerst auf die Anwendung während der Präsentation eingeschränkt.

Während einer Präsentation hat der Benutzer bei der für Tablet-PC's erweiterten Version die Möglichkeit, direkt auf den präsentierten Folien per Stift zu annotieren. Zur Verfügung stehen Kugelschreiber (fixe Breite), Filzstift (unterschiedliche Breite, je nach Druck auf der Stiftspitze) und Textmarker (fixe aber recht große Breite, horizontale Striche breiter als senkrechte). Neben den Stiften steht auch ein Radierwerkzeug, mit dem Annotationen entfernt werden können, zur Verfügung.

Neben den Freihandannotationen bietet Powerpoint auch die Möglichkeit, während der Präsentation über Vortragsnotiz zu jeder Folie und über Besprechungsnotiz allgemein Textnotizen zu verfassen.

Aus Sicht der Lehre ist der in Powerpoint implementierte Annotationsmechanismus äußerst attraktiv. Der Dozent zum Beispiel präsentiert seine vorbereiteten Folien und kann mit zwei kurzen, nicht störenden Mausklicks zu zeichnen beginnen. Dabei steht ihm die komplette Fläche zur Verfügung. Es gibt keine Sonderbereiche, die nicht zu annotieren sind. Alle Annotationen bleiben auch bei Seitenwechseln erhalten und können am Ende mit der Präsentation gespeichert werden. Keine Annotation verändert die eigentliche Präsentation direkt. Ein Tablet-PC mit Videobeamer kann so gesehen sofort einen Overheadprojektor funktional ersetzen. Darüber hinaus wäre es wünschenswert, wenn neben dem Freihandstift auch noch eine begrenzte Auswahl an anderen Zeichenwerkzeugen während der Präsentation zur Verfügung stünden. Kreise, Rechtecke und Pfeile sind in diesem Zusammenhang zu nennen.

Neben der Möglichkeit zu Annotationen bietet Powerpoint eine komplette integrierte Aufzeichnung einer Präsentation an. Dabei kann neben der Interaktionen mit Powerpoint einschließlich der Annotationen auch Audio und Video synchron aufgenommen und live oder zu einem späteren Zeitpunkt über das Netz angeboten werden. Die Präsentation wird also um mit Audio und Video annotiert. Außerdem ist es über einen Besprechungsmodus möglich,

dass alle Teilnehmer direkt und synchron mit der Präsentation arbeiten. Kollaborative Annotationen sind also möglich.

Aus Sicht der Studenten fehlt etwas Äquivalentes zu Word's Überarbeitungsfenster (vgl. 4.2.1.1.), also ein Mechanismus, mit dem Annotationen eigenständig betrachtet werden können. Ein recht mächtiges Unterstützungswerkzeug für Ordnungsstrategien, nämlich das Reduzieren auf die Annotationen, wird damit nicht angeboten. Für das Anzeigen von Präsentationen einschließlich der Annotationen stellt Microsoft einen kostenlosen Viewer zur Verfügung. Sobald auch eigene Annotationen verfasst werden sollen, ist eine Vollversion von PowerPoint nötig.

Zusammenfassend kann gesagt werden, dass Microsoft PowerPoint mit Tablet-PC aus Sicht des Dozenten gut geeignet ist, ohne zusätzlichen Aufwand Freihandannotationen auch während des Kurses anzubieten.

4.2.1.3. Microsoft Journal

Bisher wurden Standardprodukte aus dem Microsoft Office Paket betrachtet, die prinzipiell auf allen PC's mit Microsoft Betriebssystem zur Verfügung stehen und um eine Unterstützung von Tablet-PC's erweitert wurden. Mit Microsoft Journal und Microsoft Kurznotiz sollen nun Produkte betrachtet werden, welche speziell für die Benutzung mit Tablet-PC's entwickelt wurden. Im Weiteren werden diese, als Ergänzung zu bestehenden Programmen zur Präsentation eines Kursskripts angesehen. Es wird betrachtet, inwiefern damit eine, relativ zum Kursskript externe, allgemeine Verwaltung von Annotationen möglich ist.

Mit Microsoft Journal bietet Microsoft einen elektronischen Schreibblock an. Jede Schreibblockdatei besteht aus einem Kopfbereich und ein bis mehreren Seiten. Im Kopfbereich vergibt man einen Titel für die Notiz. Das Datum wird automatisch ergänzt.

Microsoft Journal interpretiert im Hintergrund jede handschriftliche Eingabe und wandelt diese intern soweit möglich in entsprechenden Text um. Eine Konvertierung in der Anzeige muß explizit angestoßen werden. Damit bleibt die Notiz wie sie vom Benutzer verfasst wurde, kann aber trotzdem von Microsoft Journal inhaltlich erfasst werden. So bietet Microsoft Journal zum Beispiel die Möglichkeit, die Freihandnotizen nach Text durchsuchen zu lassen. Außerdem wird der handschriftliche Titel beim Speichern als Dateiname angeboten, so dass ein Eintippen eines Dateinamens meist entfallen kann.

Als Werkzeug steht neben einiger Stifte, einem Radierer und einem Auswahlwerkzeug ein Markierwerkzeug zur Verfügung. Mit dem Markierwerkzeug kann ein Flaggensymbol in verschiedenen Farben auf der Notizfläche platziert werden. Bei der Suche nach Notizen kann angegeben werden, dass nur Notizen mit solchen Marken aufgelistet werden sollen. Andere vorgefertigte Symbole oder Werkzeuge wie Rechtecke oder Kreise sind nicht vorgesehen. Gleichwohl lassen sich Freihandstriche in solche Objekte umwandeln.

Aus Sicht der Lehre ist dieses Werkzeug für Dozenten und Studenten zum Erstellen von Notizen geeignet. Die Möglichkeit der Suche in handschriftlichen Notizen bringt einen klaren Vorteil gegenüber dem klassischen Notizblock. Zu betonen ist, dass auch eine Suche nach Wörtern im handschriftlichen Inhalt von Windows Journal Notizen zum Beispiel aus dem Windows Explorer heraus funktioniert.

Neben der Erstellung und der Verwaltung der Annotationen ist auch die Verbindung zum eigentlichen Kursskript wichtig. Microsoft Journal Annotationen lassen sich zum Beispiel über

den Mechanismus Object Linking and Embedding (OLE) in die meisten Windowskonformen Applikationen einbinden. Dabei wird in das ursprüngliche Dokument ein Platzhalter eingefügt, der die verknüpfte Microsoft Journal Annotation repräsentiert. Ein Doppelklick auf dieses Symbol öffnet Microsoft Journal mit der entsprechenden Notiz. Es besteht also die Möglichkeit, das Kursskript und die Annotationen prinzipiell getrennt zu verwalten und über einen Betriebssystemmechanismus wie OLE miteinander zu verknüpfen. Diese Möglichkeit besteht prinzipiell auch bei Microsoft Kurznotiz als Werkzeug für Annotationen.

4.2.1.4. Microsoft Reader

Microsoft bietet mit Microsoft Reader eine Software zum Lesen von sogenannten eBooks. Ein eBook ist ein elektronisch zur Verfügung gestelltes Buch.

Aus Sicht der Lehre bietet Microsoft Reader einige interessante Aspekte: Zum Einen ist der Reader kostenlos. Zum Anderen wird auch ein kostenloses Plugin für Microsoft Word angeboten, um aus Worddokumenten ein eBook zu erstellen. Somit ist die Erstellung eines eBooks für einen Dozenten mit einer meist vorhandenen Microsoft Word Lizenz möglich. Die Studenten können das Kursskript-eBook jederzeit über den kostenlosen Microsoft Reader lesen.

Microsoft Reader bietet die Möglichkeit, Annotationen zum Kursskript zu erstellen. Dabei wird die Annotation mit einer Stelle im Text verbunden und über ein kleines Ankersymbol angedeutet. Es stehen neben Textnotizen, Lesezeichen und Markierung bei der Version für Tablet-PC's auch Freihandnotizen zur Verfügung. Wichtig ist hier, dass die Annotationen das eigentliche Kursskript nicht verändern. Freihandnotizen werden auf dem ursprünglichen Text liegend dargestellt. Ein Fenster mit der Textnotiz öffnet sich, sobald auf das Ankersymbol der Annotation geklickt wird.
Eine Hervorhebung von Text wird durch Hinterlegen mit gelber Farbe angezeigt.

Alle Annotationen werden in einem extra Fenster aufgelistet und können dort verwaltet werden. So besteht beispielsweise die Möglichkeit, die Annotationen nach verschiedenen Kriterien zu sortieren und zu filtern. Komplexere Filteroperationen sind aber nicht implementiert.

Aus Sicht der Lehre stellt Microsoft Reader eine interessante Anwendung dar.

4.2.1.5. Zusammenfassung

Microsoft bietet mehrere Möglichkeiten, allgemein Annotationen und speziell Freihandannotationen mit in ein elektronisches Lehrumfeld zu integrieren. Als Betriebsystemhersteller und unter Einschränkung auf Tablet PC's bietet Microsoft mit Microsoft Journal eine Lösung an, mit der Annotationen relativ eigenständig, aber trotzdem umfassend angeboten werden können. Wie bei den meisten allgemein anwendbaren Lösungen, stellt auch hier die Integration in das Produkt für die Darstellung des Kursskripts den größten Haken für die Alltagstauglichkeit dar. Dies ist aber produktspezifisch und muss im Einzelfall genauer betrachtet werden.

Insgesamt wäre eine Kombination aller vorgestellten Annotationsmechanismen wünschenswert: Die Verwaltungsmechanismen von Microsoft Word Kommentaren, das direkte Zeichnen ohne Verändern der Vorlage von Microsoft PowerPoint und die für Suche heranziehbare im Hintergrund laufende Schriftenerkennung von Microsoft Journal.

4.2.2. Adobe PDF

Nachdem Annotationen Ergänzungen zu Dokumenten sind und Adobe mit seinen PDF-Dokumenten eines der verbreitesten Dokumentformate im Dokumentenbereich anbietet, hat auch Adobe für PDF-Dokumente die Möglichkeit für eigene Annotationen geschaffen. Wegen der Zusicherung der identischen Darstellung über unterschiedlichste Rechensysteme hinweg, findet das PDF-Format auch immer mehr Einzug in die Lehre. So bieten viele Dozenten, wenn sie ein Kursskript anbieten, dieses im PDF-Format an.

Im Weiteren werden die Fähigkeiten von Adobe Acrobat 6.0 Professional hinsichtlich Annotationen betrachtet (vgl. Adobe, 2005).

Acrobat 6.0 bietet unter dem Stichwort „Comments“ eine Vielzahl von Alternativen, ein PDF-Dokument zu annotieren:

- Notes
- Text edits
- Stamps
- Drawing tools
- Attachments

Außerdem bietet Acrobat 6.0 über „Reviewing documents“ die Möglichkeit, Annotationen zu einem Dokument mit anderen Benutzern auszutauschen und nach Annotationen über verschiedene Kriterien zu suchen.

Nachdem sich die Annotationswerkzeuge in Acrobat 6.0 in interessanter Weise unterscheiden, werden sie nun kurz einzeln betrachtet. Dabei fällt auf, dass die verschiedenen Werkzeuge jeweils einer in 2.2. ab Seite 11 beschriebenen Erscheinungsform von Annotationen entsprechen. Dies deutet auf eine tiefergehende Beschäftigung mit Annotationen auf Seiten von Adobe hin.

4.2.2.1. Notes

Die Option *Notes* erlaubt es, eine Textnotiz an beliebiger Stelle in einem PDF-Dokument abzulegen. Dabei wird im Dokument an entsprechender Stelle ein kleines Symbol als Anker eingefügt und an anderer Stelle ein Fenster für die eigentliche Notiz angelegt. Das Symbol und das Notizfenster können einzeln beliebig verschoben werden. Um Anzeigeplatz zu sparen, kann das Notizfenster geschlossen und bei Bedarf über das zugehörigen Symbol wieder geöffnet werden. Zudem ergänzt Acrobat 6.0 beim Erstellen der Notiz automatisch den Autor und den Zeitpunkt der Erstellung.

Das System der *Notes*, wie es von Acrobat 6.0 zur Verfügung gestellt wird, eignet sich für Studentenannotationen. Das kleine Symbol verdeckt kaum Originalinformation. In der eigentlichen Notiz kann beliebig langer Text untergebracht werden. Die Notizfenster können zum Beispiel bei zu wenig Platz einzeln in der Größe verändert oder ausgeblendet werden. Über die Möglichkeiten des „Reviewing documents“ kann gezielt nach Notizen gesucht werden. Dabei darf auch der Inhalt als Kriterium herangezogen werden. Außerdem ist es möglich, nur die Notizen zu einem Dokument auszugeben. Bei geschicktem Einsatz der *Notes* ergibt sich somit automatisch eine Kurzzusammenfassung des Gesamtdokuments.

Der Nachteil, dass *Notes* reine Textnotizen sind, fällt bei Studentenannotationen nicht allzu sehr ins Gewicht, da die Studenten zumeist direkt an einem Computer/Laptop mit Tastatur sitzen. Für Freihandnotizen bietet Acrobat 6.0 weitere Hilfsmittel, die weiter unten beschrieben werden.

Für Dozentenannotationen eignen sich *Notes* weniger gut. Zum Einen wird das Arbeiten mit der Tastatur von den meisten Dozenten während der Vorlesung als lästig empfunden. Zum Anderen liegen *Notes* auf den ersten Blick ohne Zusammenhang auf einer Seite. Nur durch Überfahren mit dem Mauszeiger werden Zusammenhänge durch Linien verdeutlicht. Dies kann aber von einem Studenten während der Vorlesung nicht initiiert werden. Hat also ein Student einmal den Überblick verloren, kann er nur noch schwerlich die Notizen zu den Symbolen zuordnen. Schließt der Dozent einzelne Notizfenster, zum Beispiel aus Platzmangel, verschärft sich diese Situation.

4.2.2.2. Text edits

Acrobat 6.0 bietet mit *Notes* die Möglichkeit, an beliebiger Stelle im Dokument Anker für Annotationen zu setzen. Mit den *Text edits* Werkzeugen wird kein neuer, freier Anker definiert, sondern eine bestehende Textstelle als Anker für verschiedene Annotationen genutzt.

Über *Text edits* lassen sich zum Einen Notizfenster, wie sie bei *Notes* benutzt werden, direkt mit Textstellen verknüpfen. Die Konzepte und Aussagen, die bei *Notes* gemacht wurden, gelten auch in diesem Zusammenhang.

Zum Anderen können speziell zu Textstellen über *Text edits* folgende spezialisierte Annotationen verfasst werden:

- *Highlight*
- *Underline*
- *Cross out*
- *Insert text*
- *Replace text*

Highlight und *Underline* dient dem Hervorheben einer Textstelle. Mit *Cross Out* können Textstellen durchgestrichen werden. Sie bleiben aber trotzdem erhalten und sichtbar.

Die beiden weiteren Optionen *Insert text* und *Replace text* arbeiten etwas anders als eventuell auf den ersten Blick vermutet. Grundsätzlich verändert keines der Annotierwerkzeuge in Acrobat 6.0 das Originaldokument. Auch diese beiden Werkzeuge fallen in diese Kategorie. Ein Einfügen von Text per Annotation (*Insert text* Werkzeug) funktioniert deshalb wie folgt: Der Benutzer markiert eine Stelle im Text und wählt das Hilfsmittel *Insert Text* aus. Damit erscheint an der Textstelle ein Einfügesymbol. Zudem öffnet sich ein Eingabefenster für den einzufügenden Text. Ist die Eingabe des einzufügenden Texts abgeschlossen, verschwindet das Fenster wieder. Nur das Einfügesymbol bleibt erhalten. Um den einzufügenden, annotierten Text anzuzeigen, klickt der Benutzer auf das Einfügesymbol.
Das Werkzeug *Replace text* stellt eine Kombination von *Cross out* und *Insert text* dar. Der Benutzer markiert den zu ersetzenden Text und wählt das Werkzeug *Replace text* aus. Der Text wird durchgestrichen, bleibt aber komplett erhalten. Zudem erscheint ein Einfügesymbol am Ende des gestrichenen Texts. Im sich öffnenden Eingabefenster trägt der Benutzer

den neuen Text ein. Nach dem Eingeben verschwindet dieses Fenster und kann nur über das Einfügesymbol wieder sichtbar gemacht werden.

Als Annotation im Umfeld der elektronischen Lehre sind nur die beiden ersten *Highlight* und *Underline* uneingeschränkt einsetzbar. Für den Dozenten stellen sie eine einfache Möglichkeit dar, Textstellen hervorzuheben. Bei Studenten entsprechen diese Werkzeuge genau dem Umgang mit Papierskripten. Eine Tastatur ist für das Erstellen nicht notwendig.

Cross out ist zwar von der Bedienung her kein Problem. Der Benutzer markiert Text und wählt das Werkzeug *Cross out*. Der markierte Text wird nun durchgestrichen dargestellt. Die Semantik dürfte im Umfeld der elektronischen Lehre aber selten sinnvoll sein, beziehungsweise stellt sich die Frage, welche Semantik dieser Darstellung zugeordnet werden soll.

Insert text und *Replace text* sind als Fehlerkorrekturen semantisch gesehen im Umfeld der elektronischen Lehre durchaus wünschenswert. Die Darstellung des neuen, korrigierten Texts ist aber so zurückhaltend beziehungsweise umständlich, dass sie für ein System zur Unterstützung der Lehre kaum brauchbar ist. Ein schnelles Erfassen der neuen, korrekten Inhalte, wie es im Lehrumfeld notwendig ist, ist nicht möglich. Pädagogisch gesehen, ist die „falsche“ Lösung zu sehr im Vordergrund.

4.2.2.3. Stamps

Wie in 2.2.1.5. erläutert, dienen spezielle Symbole der Darstellung von Metainformation. Acrobat 6.0 bietet hierzu das Werkzeug Stamps. Mit Stamps ist es zum Einen möglich, ein Symbol frei zu definieren und zum Anderen solche Symbole an beliebiger Stelle per Klick zu platzieren. Über die Funktionalität Reviewing comments können Stamps sehr schnell gefunden oder aufgezählt werden. Dabei können auch gezielt Filterkriterien angegeben werden. Zu jedem Symbol können weitere Angaben in Textform gemacht werden.

Im Hinblick auf Lehre können Acrobat 6.0 Stamps ein wichtiges und mächtiges Werkzeug sein. Dabei sind sie für den Dozenten in gleichem Maße wertvoll wie für die Studenten. Die einfache Bedienbarkeit und das freie Platzieren durch einen Klick erlaubt schnelles und effizientes Arbeiten. Außerdem sind ansprechende grafische, somit motivierende Symbole möglich. Durch den definierten, aber erweiterbaren Satz von Symbolen, ist deren Semantik schnell zu erlernen.

Wurde ein Satz von Symbolen im Sinne einer Ordnungsstrategie zusammengestellt, kann über die Funktionalität Reviewing comments sehr schnell ein Überblick über ein Kursskript gewonnen werden. Durch Filtern zum Beispiel auf Symbole für „Wichtig“ bekommt man alle entsprechenden Annotationen aufgelistet und kann darüber zur Stelle im Kursskript navigieren.

4.2.2.4. Drawing tools

Im Gegensatz zu den wiederverwendbaren Stamps bietet Acrobat 6.0 mit den Drawing tools die Möglichkeit, einzelne, freie Annotationen zu gestalten. Unter Drawing tools werden folgende Zeichenhilfen zusammengefasst:

- Rectangle
- Oval
- Arrow

- Line
- Cloud
- Polygon
- Polygon line
- Pencil

Bis auf Pencil erlauben alle Werkzeuge, dass Zeichnen des entsprechenden Objekts in beliebiger Größe an beliebiger Stelle im PDF-Dokument. Mit Pencil erhält der Benutzer die Möglichkeit Freihandfiguren zu zeichnen. Bei allen Werkzeugen kann die Farbe und die Strichstärke gesetzt werden.

Mit Pencil hat der Dozent die Möglichkeit, Veranschaulichungen (vgl. 2.2.1.6.) jeglicher Art während der Vorlesung zu erstellen. Einzig eine flächige, farbliche Füllung ist nur mit dem Werkzeug Polygon möglich. Das nachträgliche Ausfüllen von begrenzten Flächen wird nicht angeboten.

Acrobat 6.0 erlaubt es auch bei diesen Annotationen, zusätzliche Angaben in ein Notizfenster abzulegen. Die Annotation dient dabei als Anker.

Im Umfeld der elektronisch unterstützten Lehre betrachtet sind diese Zeichenwerkzeuge sinnvoll. Der Dozent kann mit Hilfe dieser Werkzeuge und einem entsprechenden Stifteingabegerät komfortabel während der Vorlesung freie Annotationen gestalten, wie es zum Beispiel klassisch an einer Tafel möglich wäre. Durch die Funktion, alle Annotationen auch in unterschiedlichem Maß transparent zu definieren, bleibt der Originaltext noch lesbar, auch wenn Annotationen dazugekommen sind.

Zu den Drawing tool Annotationen Textnotizen abzulegen, ist im Umfeld der Lehre für den Dozenten wenig nützlich. Zum Einen benötigt er dann wieder eine Tastatur. Zum Anderen ist die Zuordnung von Notizfenstern zu einzelnen Linienzügen nicht offensichtlich. Eine Veranschaulichung besteht meist aus mehreren Linienzügen und auch Teilaspekte benötigen oft schon mehrere Linien. Somit ist für den Studenten nicht klar, hinter welchem Linienzug ein Notizfenster „versteckt“ ist.

Aus Sicht des Studenten sind die Drawing tools im allgemeinen nicht so wichtig, wie für den Dozenten. Der Student kommt in seinen Annotationen meist mit Werkzeugen zum Markieren und Hervorheben aus. Steht dem Student ein Stifteingabegerät zur Verfügung, kann er mit Hilfe der Drawing tools arbeiten, als hätte er Papier und Stift vor sich. Das erlebte Arbeiten entspricht weitgehend dem klassischen papierorientierten.

4.2.2.5. Attachments

Acrobat 6.0 erlaubt neben dem Erstellen von Annotationen im Dokument auch das Verweisen auf andere Dokumente. Es besteht die Möglichkeit, auf ein bestehendes Dokument zu zeigen oder aber mit Hilfe eines Audiorecorderwerkzeugs eine Tonsequenz aufzuzeichnen. Somit können auch Sprachnotizen zu einer Stelle im PDF-Dokument als Verweis annotiert werden.Wird ein solcher Verweis ausgewählt, öffnet sich das entsprechende Dokument.

Aus Sicht der Lehre stellt die erste Option eine Variante dar, Prinzipien von Elaborationsstartegien einzubinden. Durch die Vernetzung mit anderen verwandten oder weiterführenden Dokumenten entsteht die Möglichkeit, das Präsentierte in anderem Kontext zu sehen. Der

Dozent wird diese Variante wohl eher im Vorfeld oder im Anschluss an eine Vorlesung nutzen, da dies vom eigentlichen Kursskript und damit dem Inhalt der Vorlesung ablenkt. Ähnliches gilt für den Studenten. Er wird selten eigenständig Verweise während der Vorlesung anlegen, da hier keine Zeit ist und oft der Überblick fehlt. Beim Nacharbeiten des Stoffes hingegen ist ein Verweis auf andere Dokumente durchaus vorstellbar.

Die Möglichkeit der Aufzeichnung einer Tonsequenz könnte genutzt werden, zu jedem Absatz oder jeder Seite das vom Dozenten Gesagte aufzuzeichnen. In der momentanen Umsetzung in Acrobat 6.0 ist dies aber kaum sinnvoll, da während der Aufzeichnung keine Bedienung des restlichen Programms möglich ist. Somit könnte kein einziges anderes Annotationswerkzeug benutzt werden. Zum Blättern auf die nächste Seite müsste die Aufzeichnung angehalten werden. Wird die Aufzeichnung über ein anderes Programm durchgeführt, eventuell auch mit Video, so kann aus der jeweiligen Seite im PDF-Dokument auf die entstehende Datei als Annotation verwiesen werden. Die tatsächliche Umsetzung dieses Gedankens erfordert zum brauchbaren Einsatz in der Lehre eine weitere Automatisierung. Der Dozent kann während der Vorlesung kaum zu jedem Seitenwechsel die vorhergehende Aufzeichnung beenden, den Verweis als Annotation einfügen und zur neuen Seite eine neue Aufzeichnung starten. Dieser Ablauf ist aber grundsätzlich automatisierbar.

4.2.2.6. Zusammenfassung

Insgesamt bietet Acrobat 6.0 aus Sicht der elektronischen Lehre eine recht umfassende Unterstützung von Annotationen. Adobe bietet außerdem Werkzeuge, die ein direktes Verteilen der Annotationen auf mehrere Benutzer ermöglichen (vgl. Adobe Acrobat 7.0 Professional, Funktion: Beteiligung von mit Acrobat Reader ausgestatteten Benutzern am Prüfprozess, Adobe, 2005). Somit wären auch die kollaborativen Aspekte abzudecken.
Interessant ist zudem, dass das eigentliche Dokument durch Annotationen nicht verändert wird. Außerdem können alle Annotationen auf einen Knopfdruck ausgeblendet werden. In einem eigenen Verwaltungsfenster für Annotationen können über relativ mächtige Filtermechanismen spezielle Annotationen gesucht und angezeigt werden.

Nachteil bei Acrobat 6.0 ist, dass zum Erstellen eine kostenpflichtige Acrobat 6.0 Lizenz erforderlich ist. Außerdem steht das Verwaltungsfenster für Annotationen mit seinen Filtermöglichkeiten auch nur in der kostenpflichtigen Version zu Verfügung. Damit wird ein breiter Einsatz bei den Studenten unwahrscheinlich. Das Betrachten des Kursskript inklusive aller Annotationen funktioniert auch mit dem kostenlosen Acrobat Reader alleine.

4.2.3. W3C - Annotea

Das WorldWideWeb (WWW) stellt wohl die größte Sammlung von Dokumenten überhaupt dar. Als „Hüter“ der technischen Seite des WWW hat das World Wide Web Consortium (W3C) mit Annotea im Rahmen der Erweiterung des WWW zu einem Semantischen Netz ein Protokoll zur Verwaltung und Einbindung von Annotationen zu WWW Inhalten entwickelt (vgl. Kahan, Koivunen, Prud’hommeaux & Swick, 2001 und W3C, 2005a). Für dieses Protokoll stehen Implementierungen wie zum Beispiel Amaya als Annotea-fähiger Browser zur Verfügung (vgl. W3C, 2005b).

4.2.3.1. Das Protokoll

Annotea selbst stellt keinerlei Oberfläche oder Werkzeug zur Erstellung von Annotationen bereit. Wieso soll Annotea dann in diesem Zusammenhang betrachtet werden? Interessant an Annotea ist, dass es eine offene Plattform zur Ablage, zum Austausch und zum Referenzieren von Annotationen bietet. Die Architektur sieht neben dem Webserver, der das Ursprungsdokument zur Verfügung stellt noch einen Server für die Verwaltung der Annotationen vor. Somit wird also das eigentliche Dokument nie durch Annotationen verändert, was aber nicht heißt, dass der Browser das Dokument nicht anders anzeigt. Außerdem kann durch Wahl eines anderen Verwaltungsservers für die Annotationen ein komplett anderer Satz von Annotationen gewählt werden.

Annotea definiert folgende Funktionen:

- Annotation anlegen
- Annotationen suchen
- Annotationen bearbeiten und löschen
- Annotationen beantworten

4.2.3.2. Annotationen anlegen und auslesen

Zum Anlegen einer Annotation stehen laut Protokoll zwei Möglichkeiten zur Verfügung. Zum Einen kann der Inhalt der Annotation dem Anlegenbefehl direkt mitgegeben werden. Zum Anderen wird nur eine Referenz auf den Inhalt der Annotation, hier speziell ein URI, beim Anlegen übergeben.

Die Stelle, zu der eine Annotation im Originaldokument gehört, wird über einen XPointer-Ausdruck spezifiziert. XPointer erlaubt es, einzelne Elemente in einem XML-Dokument zu adressieren.

Aus Sicht der Lehre ergeben sich aus diesem Mechanismus zur Anlage von Annotationen folgende Möglichkeiten und Einschränkungen:

Textliche Annotationen und andere, welche sich in XML beschreiben lassen, können direkt als Inhalt der Annotation übergeben werden und werden somit direkt über den Annotationsserver abgelegt. Da eine Freihandeingabe als Vektorgrafik dargestellt werden kann, kann diese auch direkt in SVG, einer XML-Beschreibungssprache für Vektorgrafiken, notiert werden. Somit können auch Freihandgrafiken direkt als Inhalt einer Annotation dienen.

Die Referenzierung von Annotationen zum Dokument über Xpointer geschieht auf inhaltlicher und nicht auf Darstellungsebene. Damit kann zum Einen sehr exakt und maschineninterpretierbar beschrieben werden, zu welchem Teil im Ursprungsdokument eine Annotation gehört. Zum Anderen ist aber eine Annotation, welche auch von der aktuellen Darstellung abhängt, kaum möglich. Als Beispiel ist hierfür Einkreisen und mit einem Pfeil referenzieren zu nennen.

Sollen Annotationen, welche nicht direkt zu übergeben sind, referenziert werden, müssen sie auf einem anderen erreichbaren Server angeboten werden. Werden diese Annotation erst bei der Anlage neu erstellt, muss der Ersteller auf dem Server Schreibzugriff haben. Der Annotationsserver kann diese Aufgabe mit erfüllen, ist aber vom Protokoll her nicht dazu verpflichtet.

Über die Benutzungsschnittstelle für das Anlegen und Anzeigen von Annotationen ist im Protokoll nichts ausgesagt. Denkbar ist zum Beispiel, dass der Browser für das Kursskript den Inhalt und die Annotationen auf zwei übereinanderliegenden transparenten Ebenen darstellt. Vorstellbar ist auch, dass der Endbenutzer durch Zwischenschalten eines Proxies mit seinem normalen Browser arbeiten kann. Der Proxy kümmert sich dann um die Verwaltung und Einflechtung der Annotationen.

Neben dem Erstellen und Anzeigen von Annotation ist die Suche und Auswertung von Annotationen eine wichtige Funktion. Durch die maschineninterpretierbare Referenzierung mittels XPointer sind erweiterte Suchmechanismen möglich. Eine Anfrage wie: Gib alle Annotationen in Kapitel xy oder Absatz a zurück ist, ist problemlos zu bearbeiten.

Das Annotea Protokoll stellt selbst noch keinen Mechanismus zur Erstellung solcher Abfragen bereit. Im Anhang zur Annotea-Dokumentation wird aber mit ALGAE schon eine mögliche Abfragesprache, die dann schon auf Serverseite bearbeitet werden kann, vorgestellt.

Annotea spezifiziert bisher nur eine Anfrage, die zu einer Seite alle Annotationen eines bestimmten Annotationsservers liefert. Die Antwort auf diese Anfrage kann dann aber vom Client sehr einfach zum Beispiel per XSLT gefiltert und verarbeitet werden, da alle Attribute in XML-Notation (RDF) vorliegen.

4.2.3.3. Kollaborative Aspekte

Über die Möglichkeit Annotationen zu verändern und zu löschen, stellt Annotea rudimentäre Möglichkeiten für ein gemeinsames Arbeiten an Annotationen bereit. Außerdem sieht das Protokoll einen Mechanismus zur Erstellung und Verwaltung von Antworten auf Annotationen vor. Beide Möglichkeiten werden nun kurz vorgestellt und bewertet.

Das Annotea Protokoll erlaubt es, eine Annotation zu verändern oder zu löschen. Dabei wird keine Einschränkung auf eine Rolle oder Person hin gemacht. Somit kann potentiell jeder Benutzer die Annotation eines anderen verändern. In der elektronischen Lehre ist dies aus Sicht des Dozenten kritisch zu bewerten. Eine Annotation, welche er im Unterricht gemacht hat, könnte so von Studenten verändert und eventuell zu deren Vorteil manipuliert werden. Innerhalb der Annotea Architektur lässt sich das Problem dadurch lösen, dass für Dozentenannotationen und für Studentenannotationen zwei getrennte Annotationsserver bereitgestellt werden und nur der Dozent auf Server für seine Annotationen Schreibzugriff hat. Auch bei den Studenten ist es zweifelhaft, ob ein Student einverstanden ist, dass ein anderer seine Annotationen verändert oder löscht. Dieser Mechanismus macht also nur bei verantwortungsvollem Umgang für kollaborative Zwecke Sinn.

Eine weitaus bessere Alternative für kollaboratives Arbeiten mit Annotationen bietet der zweite Mechanismus. Annotea sieht darin vor, dass eine Annotation „beantwortet“ werden kann. Konzeptuell kann hier auch von einer Annotation der Annotation gesprochen werden. Der Vorteil ist, dass die Originalannotation erhalten bleibt.

In der Lehre kann der Antwortmechanismus zum Beispiel einer Newsgroup entsprechend genutzt werden. Ein Student stellt eine Frage zu einer Stelle im Kursskript als Annotation bereit. Andere Studenten können nun Ihre Meinung als Antwort auf die Annotation schreiben. Auch diese Antworten können wieder beantwortet werden. Der Vorteil ist, dass der jeweilige Gesprächsstrang direkt zur entsprechenden Stelle im Kursskript verknüpft ist. Insbesondere als Feedback für den Dozenten ist dies ebenfalls interessant.

4.2.3.4. Zusammenfassung

Prinzipiell ermöglicht Annotea alle gewünschten Annotationsmöglichkeiten. Die Konzentration auf die Referenzierung in die innere Struktur des Kursskripts erlaubt eine bessere maschinelle Interpretation und damit Suche beziehungsweise Auswertung. Andererseits sind Referenzierungen im Layout, die damit auch intensiv den visuellen Kanal unseres Gedächtnisses nutzen, nur eingeschränkt möglich. Gerade in der Lehre ist dieser Aspekt zu beachten, da hier alle menschlichen Aufnahmekanäle genutzt werden sollten. Der Mechanismus zur Beantwortung von Annotationen ist aus Sicht der Lehre sehr interessant. Eine Newsgroup zur Vorlesung und das Kursskript verschmelzen damit zu einer zusammengehörigen Einheit.

4.2.4. Bewertung

In Abschnitt 4.2. wurden verschiedene bestehende Produkte und ihre Unterstützung von Annotationen betrachtet. Nun wird der in Kap.3.4., S. 45 aufgestellte Kriterienkatalog für Annotationen in der Lehre auf diese Produkte angewandt (vgl. Tab. 4, Prädikate vgl. 4.1.7).

Wird diese Matrix wieder ohne Fokus auf ein bestimmtes System betrachtet, fällt auf, dass die Bereiche „Pflicht“ und „öffentliche Annotation“ wieder von den meisten System, wenn auch nicht ganz so deutlich, gut abgedeckt werden. Im Gegensatz zu den betrachteten Lehrsystemen wird der Bereich „private Annotation“ und „Auswerten“ deutlich besser abgedeckt. Der Bereich „Gruppenannotation“ fällt ähnlich schwach aus.

Auch hier ist die Begründung wieder in der Historie der Produkte zu finden. Die Produkte waren zu Beginn für einen einzelnen Benutzer konzipiert. Als die Produkte um Annotationsunterstützungen erweitert wurden, wurde dies vorerst beibehalten. Der Benutzer konnte für sein eigenes Arbeiten private Annotationen erstellen. Ziemlich bald wurde aber auch erkannt, dass für ein effektives Zusammenarbeiten und bei der Benutzung von Annotationen Mechanismen zum Austausch der Annotationen notwendig sind. Neben diesen Gruppenmechanismen wurden auch sehr früh Mechanismen zum Verwalten und Auswerten der Annotationen integriert. Auch dies wurde zur Effizienzsteigerung im Umgang mit elektronischen Annotationen durchgeführt.

Insgesamt blicken die allgemeinen Applikationen auf einen größeren praktischen Erfahrungsschatz zurück. Schon alleine deshalb wäre es schändlich, sie nicht näher betrachtet zu haben. Außerdem lässt sich wegen der unterschiedlichen Ausgangspunkte der Systeme beim Studium der Annotationsunterstützung in Lehrsystemen und allgemeinen Applikationen ein breiteres Verständnis für Annotationen erwarten.

Konkret konnten über die Erkenntnisse aus 4.1. hinaus weitere interessante Konzepte zur Erstellung und Verwaltung von Annotationen identifiziert werden. Sinnvoll wäre aus Sicht der Lehre eine Umsetzung der folgenden Konzepte:

- Keine Änderung des Originaldokuments durch Annotationen
- Eigenständige Verwaltung für Annotationen mit Suche nach verschiedenen Kriterien
- Interpretation von Freihandtext, ohne explizite Umwandlung in der Darstellung und somit zur Verfügung stellen des Inhalts für eine elektronische Verwaltung der Annotationen
- Kollaborative Mechanismen im Bereich der Annotationen

Aus den in diesem Kapitel gewonnen Einsichten wird nun im folgenden Kapitel ein Vorschlag für eine Architektur abgeleitet.

	Kriterium	MS Word	MS PowerPoint	MS Journal	MS Reader	Adobe PDF	W3C Annotea
Pflicht							
K0	Verbindung Annotation zu Skript	+	+	+	+	+	+
öffentliche Annotation							
K1b	Kein Medienbruch bei öffentlichen Annotationen	+	+	o	+	+	o
K2b	Lokalität der öffentlichen Annotation	+	+	o	+	+	o
K3b	Keine Einschränkung der Kreativität bei öffentl. Annot.	o	+	+	+	+	o
K5	Jeder kann öffentliche Annotationen machen	-	-	-	-	o	-
K6b	Übernehmen von öffentlichen Annotationen	-	-	-	-	o	+
private Annotation							
K1a	Kein Medienbruch bei privaten Annotationen	+	+	o	+	+	+
K2a	Lokalität der privaten Annotation	+	+	o	+	+	o
K3a	Keine Einschränkung der Kreativität bei privaten Annot.	o	+	+	+	+	o
K4	Private Annotationen sind immer möglich	+	+	+	+	+	+
Gruppenannotation							
K6a	Übernehmen von Gruppenannotationen	-	o	o	-	+	o
K7a	Student legt fest, wer Annotationen sehen darf	-	-	-	-	+	-
K7b	Student legt fest, wessen Annotationen er sehen will	-	-	-	-	o	-
K7c	Dem Dozenten Studentenannotationen anzeigen	-	-	-	-	+	+
K9	Hinweis, wenn neue Annotation gemacht wurde	-	-	-	-	+	o
K10	Quellenangabe bei übernommenen Gruppenannotation	-	-	-	-	+	o
K11	Annotationen können als kollaborativ markiert werden	-	-	-	-	o	-
K12	Pers. können in kollaborative Gr. aufgenommen werden	-	-	-	-	o	-
Auswerten							
K8a	Filtermechanismen / Kompetenzdarstellung	+	+	o	o	+	+
K8b	Suchen nach Markierungen von öffentlichen Annot.	+	+	o	o	+	+
K13	Markieren von öffentlichen Annot. als Fehlerkorrektur	o	o	o	o	+	o
K14	Umfrageannotationen oder spez. Bereiche dafür def.	o	-	-	-	o	o
K15a	Häufigkeiten von Umfrageannot./Annot. in spez. Bereich	-	-	-	-	-	o
K15b	Liefern von kumulierten Zeitinformationen zu Annot.	-	-	o	o	o	o
K15c	Liefern von kumulierten Ortsinformationen zu Annot.	-	-	-	-	-	o
K15d	Liefern von kumulierten Typinformationen zu Annot.	o	o	o	o	o	o

Tab. 4: Bewertung der Kriterien für ausgewählte allgemeine Systeme

5. Architektur zur Integration von Annotationen

5.1. Konzept

Im folgenden wird eine Architektur vorgestellt, welche es erlaubt, Annotationen in bestehende Systeme zur Unterstützung elektronischer Lehre zu integrieren. Wird im weiteren - nun technischem - Zusammenhang von Benutzer gesprochen, ist damit Dozent und Student gleichermassen gemeint. Insgesamt wurde die Entwicklung entlang der von Bernd Oesterreich (1998) vorgeschlagenen Vorgehensweise zur Entwicklung objektorientierter Software durchgeführt. Speziell die Designregeln und -heuristiken (Seite 195 f., ebd.) wurden, nicht zuletzt wegen eigener guter Erfahrungen damit, beherzigt.

5.1.1. Anforderungen

Zu Beginn der Konzeptvorstellung werden nun die sich aus den in der bisherigen Arbeit im Zusammenhang mit Annotationen aufgeführten Kriterien ergebenden Anforderungen aufgelistet. Dabei werden schon einige Kriterien, der Möglichkeit der gemeinsamen Umsetzung wegen, zusammengefasst. Diese Möglichkeit der gemeinsamen Umsetzung basiert größtenteils darauf, dass Studenten und Dozenten innerhalb des Konzepts als Personen, die Annotationen benutzen, gleichgestellt werden. Eine Differenzierung der beiden Rollen kann, wie später dargestellt, über die anpassbaren Benutzungsoberflächenelemente erfolgen.

Die Kürzel der Ursprungskriterien (vgl. Kiterienkatalog, Kap. 3.4.), aus denen eine Anforderung abgeleitet wird, sind in Klammer angegeben.

- A1: Verbindung von Annotation zu Skript (K0)
- A2: Annotationen können ohne Medienbruch im elektronischen Kursskript integriert, abgerufen werden (K1a, K1b, K2a, K2b)
- A3: Keine Einschränkung der Kreativität bei Annotationen (K3a, K3b)
- A4: Eigene Annotationen sind immer und ohne Netzzugriff machbar/lesbar (K4)
- A5: Annotationen sind zwischen Benutzern austauschbar (K5, K6, K9, K10)
- A6: Eigene Annotationen sind nicht automatisch allen zugänglich (K7a, K7b, K7c)
- A7: Annotationen können kategorisiert, gefiltert und gesucht werden (K8a, K8b, K13)
- A8: Kollaborative Annotationen sind möglich (K11, K12)
- A9: Über Annotationen kumulierende Funktionen werden angeboten (K14, K15a, K15b, K15c, K15d)

Die Aufgabenstellung, fast beliebige bestehende elektronische Systeme zur Unterstützung der Lehre einfach und effizient um Annotationen zu erweitern, bedingt folgende weitere Anforderungen an eine Lösung:

- A10: Die Schnittstelle zum bestehenden Kursskriptssystem ist schmal

- A11: Die Palette der Annotationsformen ist auf die jeweilige Anwendung anpassbar und erweiterbar.

Die Anforderungen führen direkt zu weiteren Folgerungen für die Umsetzung:

- F1: Annotationen werden eigenständig außerhalb des Kursskripts verwaltet (wegen A7)
- F2: Annotationen sind zeitlich und örtlich mit dem Kursskript verlinkt (wegen A1, A2)
- F3: Das Konzept erfordert eine Benutzeridentifikationskomponente (wegen A5, A6)
- F4: Daten zu Annotationen müssen (auch) lokal gespeichert werden (wegen A4)
- F5: Das Konzept erfordert eine Replikationskomponente (wegen A4, A5)

Die aufgeführten Anforderungen und Folgerungen bilden zusammen die Vorgabe, welche von der Annotationsarchitektur zu erfüllen ist.

5.1.2. Systemumgebung

In einem nächsten Schritt wird nun die Umgebung, in die sich das zu entwickelnde Annotationssystem einbinden muss, spezifiziert.

Die vorliegende Arbeit behandelt die Integration von Annotationen in *bestehende* elektronische Systeme zur Unterstützung der Lehre. Da das zu erstellende Teilsystem nur Annotationsunterstützung bietet und ein Kurs nach der Festlegung im Kap.1 auf einem Kursskript basiert, ergibt sich im Umkehrschluss, dass das umgebende elektronische System zur Unterstützung der Lehre für das Kursskript verantwortlich ist. Das umgebende System verwaltet also das Kursskript, stellt es dar und ermöglicht die Navigation darin. Für diese Funktionalität bietet es außerdem schon eine Benutzungsoberfläche an.

Zur sinnvollen Einbindung der Annotationen ist es außerdem notwendig, dass das Kursskript selbst „stabil“ ist. Dies bedeutet, dass eine Sicht auf das Kursskript zu jedem Zeitpunkt wieder hergestellt werden kann. Annotationen machen vom Standpunkt des Lernenden kaum Sinn, wenn die annotierten Kursskriptteile verschwinden oder sich verändern. Die Grundlage der Annotation wäre in Frage zu stellen.

Für die Einbindung von Annotationen muss die Umgebung ein Mindestmaß an Schnittstellen zur Verfügung stellen. Die Anforderungen an diese Schnittstellen sind einer einfachen Einbindung der Annotationen wegen gering zu halten, denn sie bestimmen den Arbeitsaufwand für die Integration von Annotationen auf der Seite des bestehenden Systems (vgl. Anforderung A10).

Prinzipiell sind drei Arten von Schnittstellenmethoden notwendig:

- Die Methoden, die es der Annotationsunterstützung erlauben, auf die Kursskriptdarstellung Einfluß zu nehmen. Darunter fällt das Anspringen einer Stelle im Kursskript, aber auch die Wahl der Darstellung.
- Die Methoden, welche die Annotationsunterstützung von Änderungen in der Darstellung beziehungsweise Platzierung des Kursskripts informieren. Dies entspricht also einem Ereignis, das vom Kursskript in der Annotationskomponente ausgelöst wird.

- Methoden, die es der Annotationsunterstützung erlauben, sich in die Benutzungsoberfläche der Kursskriptkomponente zu integrieren. Wie später gezeigt wird, ist dieser Punkt nicht unbedingt notwendig, erhöht aber die vom Benutzer empfundene Integration der beiden Teilsysteme.

5.1.3. Anwendungsfälle (Use Cases)

Die im Rahmen der Analyse erkannten Anwendungsfälle werden nun vorgestellt. Da eine Anwendungsfallanalyse auf konzeptueller Ebene angesiedelt ist, wird jeder Anwendungsfall auch dahingehend beurteilt, ob er nur der technischen Umsetzung wegen entstanden ist oder ob er inhärent in Annotationen Relevanz hat. Mit diesem Vorgehen wird sichergestellt, dass anwenderzentriert und nicht technikzentriert analysiert wurde.

Prinzipiell wird zwischen den Phasen „Erstellen“ und „Benutzen“ von Annotationen unterschieden. Die Phase „Erstellen“ ist durch eine möglichst einfach gehaltene und intuitive Benutzungsschnittstelle zur Erstellung von Annotationen charakterisiert.

In der Phase „Benutzen“ wird zum Einen über einen einfach zu bedienenden Mechanismus, jederzeit ein Zugriff zu zeitlich, oder örtlich zur aktuellen Stelle im Kursskript korrelierten Annotationen zur Verfügung gestellt. Zum Anderen erlaubt diese Phase speziell auch die vom Kursskript unabhängige Nutzung der Annotationen. Dazu gehört unter anderem die Suche nach einer Annotation.

Für die Phase „Erstellen“ wurden die beiden in Abbildung 11 dargestellten Anwendungsfälle identifiziert:

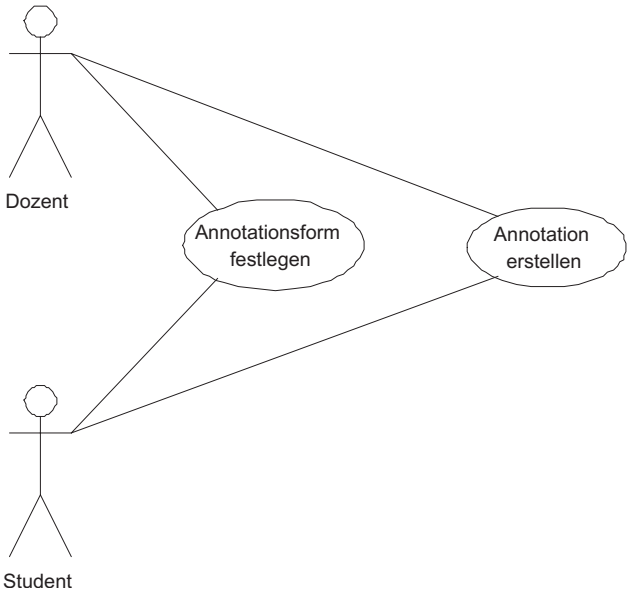


Abb. 11: Anwendungsfälle der Phase „Erstellen“

Der Anwendungsfall *Annotationsform festlegen* steht dafür, dass der Anwender aus einem Satz von Annotationstypen auswählen muss. Da seine Wahl automatisch für alle folgenden, also mehrere Anwendungsfälle *Annotation erstellen* gilt, ist *Annotationsform festlegen* berechtigterweise ein eigener Anwendungsfall.

Dieses Vorgehen findet sich auch schon in papiergestützten Umgebungen: Möchte der Student die Annotationsfo r m ”Highlighten“ anwenden, wird er einen Textmarker als Schreibwerkzeug wählen, will er Stichpunkte zusammenfassen wählt er einen Schreibstift. Solange er die Annotationsform beibehält, bleibt der Benutzer auch bei seinem Werkzeug. Somit ist dieser Anwendungsfall nicht nur ein Resultat der technischen Umsetzung.

Andererseits kann man sich überlegen, ob dieser Anwendungsfall in einem elektronisch unterstütztem Umfeld überhaupt notwendig ist. Es ist leicht einzusehen, dass der Schritt an und für sich unabdingbar ist. Er könnte einzig durch einen entsprechenden Algorithmus, der eventuell aus dem Kontext schließt, welche Annotationsform gewünscht ist, ersetzt werden. Da für den Kontext sicher auch das Benutzerverhalten analysiert werden muss, würde dies einer impliziten Festlegung der Annotationsform durch den Benutzer gleichkommen.

Der Anwendungsfall *Annotation erstellen* steht für das eigentliche Erstellen der Annotation. Für die Implementierung ist zu berücksichtigen, dass das elektronische Umfeld eine gegenüber der klassischen Lehre erweiterte Palette von Annotationen zulässt (Audio, Video, ...). Dies und ergonomische Aspekte haben zur Folge, dass dieser Anwendungsfall über verschiedene, eventuell austauschbare Benutzungsschnittstellen umgesetzt werden muss.

Da der Anwendungsfall *Annotation erstellen* den Einstiegspunkt in die gesamte Thematik der Arbeit - klassisch wie elektronisch unterstützt - darstellt, erübrigt sich die Frage nach der Notwendigkeit.

Die Phase „Benutzen“ wird durch die Anwendungsfälle in Abbildung 12 charakterisiert.

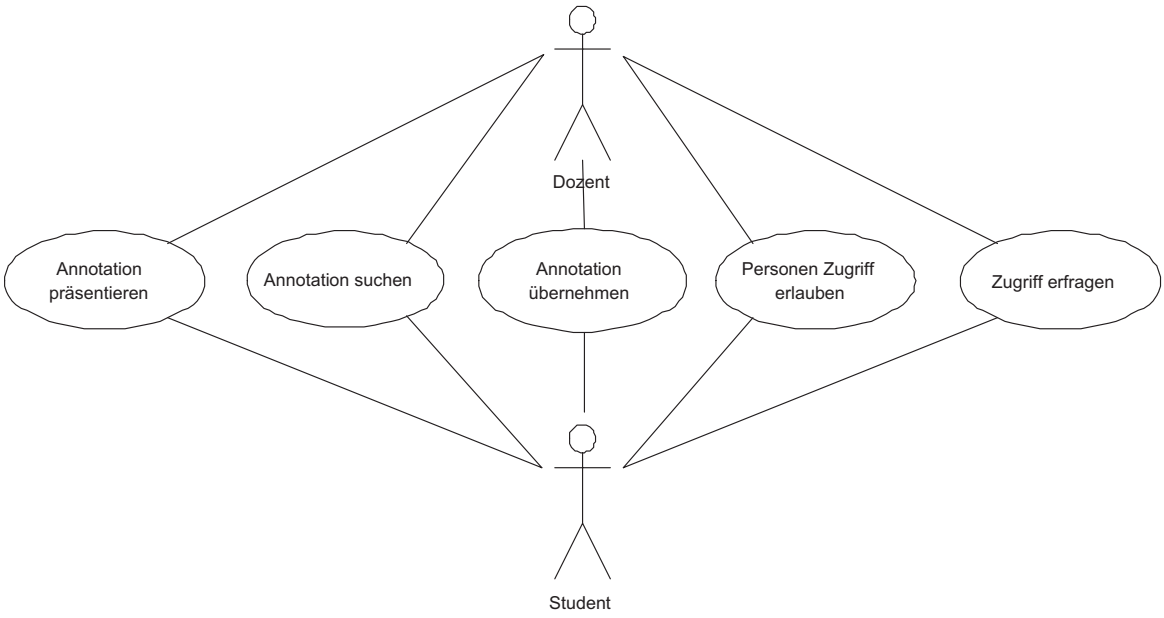


Abb. 12: Anwendungsfälle der Phase „Benutzen“

Alle diese Anwendungsfälle haben gemeinsam, dass sie auch in der klassischen papiergestützten Lehre vorkommen: Ein Student schaut seine Annotationen während des Wiederholens noch einmal an. Er blättert das Kursskript auf der Suche nach einer speziellen Annotation durch. Beim Wiederholen in einer Lerngruppe werden Annotationen anderer abgeschrieben. Die Bitte um die Unterlagen eines anderen gehört genauso in das Studentenleben, wie

das Verleihen der eigenen Aufzeichnungen. Somit liegen in dieser Auflistung keine Anwendungsfälle nur auf Grund der technischen Umsetzung vor.

Andererseits sind auch alle genannten Anwendungsfälle notwendig, um das geforderte Spektrum zu unterstützen. Weitere Anwendungsfälle, also dem Benutzer zur Verfügung zu stellende, globale Arbeitsabläufe sind nicht notwendig.

Der Anwendungsfall *Annotation präsentieren* erscheint auf den ersten Blick im Anwendungsfall *Annotation erstellen* enthalten zu sein. Dies ist der verschiedenen möglichen Annotationsformen und ergonomischer Erfordernisse wegen aber nicht der Fall und somit für eine klare Gliederung eindeutig zu trennen. Zum Beispiel erlaubt das Erstellen einer Audioannotation nur schwerlich die gleichzeitige Präsentation. Außerdem deckt der Anwendungsfall *Annotation präsentieren* in diesem Vorschlag zwei Aspekte ab: Zum Einen wird darüber die Annotation im Kontext des Kursskripts angezeigt. Zum Anderen sollen Annotationen auch eigenständig ohne Kursskript dargestellt werden können. Dies ist etwa im Anwendungsfall *Annotation suchen* wünschenswert.

Der Anwendungsfall *Annotation suchen* selbst erlaubt dem Benutzer einzelne Annotationen, sowie eine Menge von Annotationen anzusprechen. Hierüber wird also auch der geforderte Filtermechanismus implementiert (vgl. A7). Dieser Anwendungsfall kann bei entsprechender Umsetzung neben der Zeitersparnis eine erste echte Erweiterung beziehungsweise Vereinfachung gegenüber dem papiergestützten System bedeuten. Außerdem wird das Generieren von Feedback jeglicher Art, ob für den Dozenten oder den Studenten, in dieser Arbeit ebenfalls unter diesen Anwendungsfall subsumiert. Hierzu ist es notwendig, auch Attribute der Annotationen kumulieren und in entsprechender Form anzeigen zu können (vgl. A9).

Die Idee einer lokalen Annotationssammlung je Benutzer wird im Architekturvorschlag dieser Arbeit ein zentraler Punkt (vgl. A4, F4). Nachdem auch ein Austausch der privaten Annotationen unter den Benutzern zu unterstützen ist (vgl. A5), ist daher der Anwendungsfall *Annotation übernehmen* notwendig. Das System muss dafür Annotationen anderer bereitstellen. Dies ist aber kein eigener Anwendungsfall im ursprünglichen Sinn, da dieser Schritt nicht explizit vom Benutzer angestoßen wird. Vielmehr stellt das Bereitstellen einer Annotation eine Konsequenz der nächsten beiden Anwendungsfälle dar.

Damit beim *Annotation übernehmen* auch die Privatsphäre der Benutzer gewahrt bleibt (vgl. A6), läßt der Anwendungsfall *Personen Zugriff erlauben* einem Benutzer einen Benutzerkreis definieren, der Zugriff auf die eigenen Annotationen des Benutzers hat. Die dabei konzeptuell gesehen entstehende Liste wird im folgenden als *Erlaubt-Liste* bezeichnet.

Während beim vorhergehenden Anwendungsfall ein Benutzer nur passiv darauf warten kann, den Zugriff auf andere Annotationen zu erhalten, kann er diesen mit Hilfe des Anwendungsfalls *Zugriff erfragen* auch aktiv anfordern. Die Entscheidung bleibt aber beim Besitzer der jeweiligen Annotation. Als Ergebnis des Anwendungsfalls *Zugriff erfragen* entsteht ein Liste der Benutzer, deren Annotationen eingesehen werden wollen. Sie wird im folgenden *Gesucht-Liste* genannt.

Dem aufmerksamen Leser entgeht nicht, dass kein Anwendungsfall *Annotation löschen* oder *Annotation verändern* existiert. Dies liegt daran, dass sich keine allgemein akzeptierte Semantik hinter diese beiden Operationen legen lässt, falls andere Benutzer Annotationen schon übernommen haben. Sollen deren übernommene Annotationen dann auch gelöscht werden, oder machen sie für den einzelnen auch so weiterhin Sinn? Dies ist ein systemimmanentes Problem. Versionskontrollsysteme haben hier ähnliche Anforderungen. Im hier prä-

sentierten System werden diese beiden Anwendungsfälle durch eine Annotation der Annotation, also durch den Anwendungsfall *Annotation erstellen* abgedeckt. Ähnlich geht zum Beispiel CVS als Vertreter eines Versionskontrollsystems vor. Konkret heißt dies: Will ein Benutzer eine Annotation löschen, so annotiert er die Ursprungsannotation mit einer „gelöscht"-Annotation. Ein anderer Benutzer, der die Ursprungsannotation schon übernommen hat, bekommt im normalen Ablauf auch die „gelöscht"-Annotation zur Übernahme angeboten. Er kann diese Annotation aber auch nicht übernehmen und behält somit die Ursprungsannotation. Im CVS entsteht dabei dann ein eigener Versionszweig.

5.1.4. Lösungsaspekte

Bevor im Weiteren auf den konkreten Architekturvorschlag eingegangen wird, sollen der Übersicht und damit Verständlichkeit wegen, einige Konzepte der Lösung vorab isoliert vorgestellt werden.

5.1.4.1. Erweiterbarkeit und Anpassbarkeit

Der Anforderung A11 nach Erweiterbarkeit und Anpassbarkeit der Annotationsformen wird durch eine Zweiteilung der Architektur Rechnung getragen (vgl. Abb. 13):

Der eine Teil der Architektur stellt eine Infrastruktur für die Verwaltung der Annotationen zur Verfügung, ohne genauer auf das Wesen einzelner, spezieller Annotationen oder der Umgebung einzugehen. Dabei bildet dieses Rückgrat prinzipiell über die internen Abläufe schon alle identifizierten Anwendungsfälle ab. Die Endstücke dieses Rückgrats bilden jeweils Controller-Objekte, die im Einzelnen später erläutert werden. Wichtig ist, dass diese Controller-Objekte noch zum Rückgrat der Annotationsarchitektur gehören, die Erstellung der auf den konkreten Annotations-, Darstellungs-, Netz und Datenbanktyp spezialisierten Klassen anstossen und dann als Schnittstelle zwischen diesen und den Objekten des Rückgrats dienen.

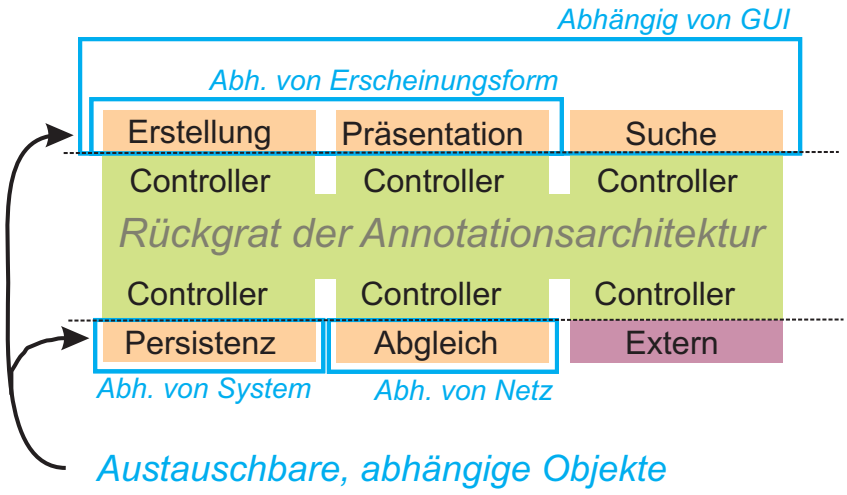


Abb. 13: Zweigeteilte Architektur

Diese Technik ist das Erfolgsrezept vieler erfolgreicher Systeme, bei denen es auf Skalierbarkeit, Robustheit und Anpassbarkeit ankommt. Meist wird diese Technik in objektorientiertem Umfeld eingesetzt, da sich hier die Beschäftigung mit einem nur teilweise spezifizierten

Konzept - hier der Annotation - schon in der Programmiersprache über Oberklassen formulieren lässt.

Beispiele für den erfolgreichen Einsatz dieser Technik liefern:

- ET++ von André Weinand (1992) mit seinem weitreichenden Einfluß auf heutige Fenstersysteme
- CORBA der OMG (Object Management Group) (vgl. OMG, 2005) mit seinem Setzen von Akzenten im Bereich verteilter Anwendungen
- Eclipse Plattform der Eclipse Foundation (vgl. Object Technology International, 2001), als weit verbreiteter, exzellent erweiterbarer Entwicklungsumgebung im Produktiveinsatz, einem aktuellen Musterbeispiel dieser Technik (vgl. Bolour, 2003)

Vorteil dieser Trennung ist, dass die Infrastruktur und die einzelnen Annotationsformen unabhängig voneinander programmiert werden können. Steht der Infrastrukturteil, erfährt er kaum Änderungen, was der Robustheit des Gesamtsystems zu Gute kommt. Erweiterungen bei den Annotationen erfordern nur Eingriffe an relevanten Teilen, nämlichen direkt den Werkzeugen für Annotationen. Diese Annotationswerkzeuge sind problemlos und isoliert austauschbar.

Neben den Annotationen sind im Architekturvorschlag dieser Arbeit auch alle Schnittstellen zur Umgebung hin auswechselbar gestaltet. Dies entspricht der Trennung der Businesslogik von der konkreten Darstellung, dem konkreten Mechanismus für die Persistenz und dem konkreten Kommunikationsmechanismus zwischen den verteilt laufenden Komponenten. Für die Trennung gibt es immer eine abstrakte Implementierung, die dann in der konkreten Instanz der Anwendung dynamisch über ein Factory Pattern in eine konkrete Implementierung umgesetzt wird.

5.1.4.2. Integration in ein bestehendes Kursskriptverwaltungssystem

Je nach getriebenem Aufwand lassen sich verschiedene Stufen von Integration des Annotationssystems in ein Kursskriptverwaltungssystem erreichen. Je vollständiger die Integration ist, desto mehr Aufwand ist auf Seiten des bestehenden Systems zu erwarten. Bei entsprechend guter Architektur des bestehenden Kursskriptverwaltungssystems werden der schmal gehaltenen Schnittstelle wegen, wie sich später zeigt, Änderungen aber trotzdem nur an wenigen Stellen notwendig. Der Integrationsgrad läßt sich über die folgenden beiden großen Berührungspunkte genauer spezifizieren:

- Die jeweiligen Darstellungsflächen
- Die jeweiligen Steuerungselemente

Unter Darstellungsflächen sind hier die Bildschirmteile, die von den jeweiligen Systemen für ihre Hauptobjekte, also Kursskript zum Einen und Annotationen zum Anderen, genutzt werden zusammengefasst. Die Darstellungsflächen sind entweder *vertikal geschichtet* oder *horizontal verschränkt*.

Sind sie *vertikal geschichtet* bedeutet dies, die Annotationskomponente legt seine Darstellungen für Annotationen über die Darstellung des Kursskripts, gleichsam einer Glasplatte, die über ein Dokument gelegt wird. Wird nun auf der Glasplatte geschrieben, hat dies außer eventueller Abdeckungen keinen Einfluß auf die darunter liegende Schicht. Wird das Doku-

ment verschoben, bleiben die Teile auf der Glasplatte gleich und passen eventuell nicht mehr zum darunter liegenden Inhalt. Dies entspricht somit offensichtlich einer losen Integration.

Sind die Darstellungsflächen *horizontal verschränkt*, teilen sich die beiden Komponenten eine Ebene, in der beide Ihre Darstellungen gleichberechtigt parallel unterbringen. Dies erfordert, dass beide Komponenten Zugriff auf den gemeinsamen Fensterbereich haben. Durch den gemeinsamen Zugriff auf den Fensterbereich kann ein System auch die Darstellung von Teilen des anderen Systems direkt manipulieren. Die sich ergebende Integration ist deshalb relativ eng.

Unter Steuerungselementen sind Menüs und Werkzeugleisten zu verstehen. Sie sind entweder *parallel getrennt* oder *untergeordnet integriert* zueinander.

Parallel getrennt steht hierbei dafür, dass jedes System sein eigenes unabhängiges Menü anzeigt und verwaltet. Der Benutzer hat also zwei getrennte Gruppen von Steuerungselementen, die unabhängig voneinander existieren.

Sind die Steuerungselemente *untergeordnet integriert*, so liefert das eine System die oberste Stufe an Steuerungselementen, erlaubt aber dem anderen System, eigene Teile mit einzubauen. Dem Benutzer präsentiert sich das ganze als ein zusammengehöriger Block. Dies entspricht dann einer engen Kopplung.

Eine Matrix über diese Attribute ergibt folgende Kategorisierung für die Integration:

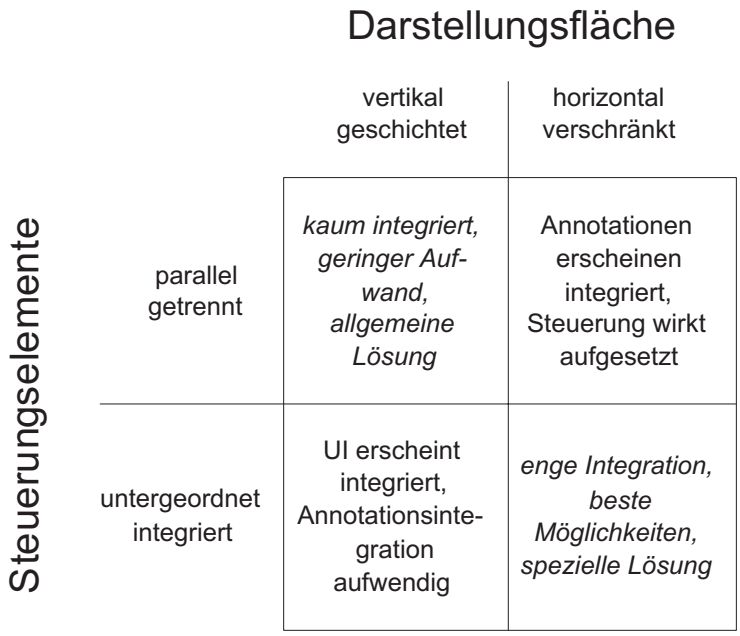


Abb. 14: Matrix für den Integrationsgrad des Annotationssystems im umgebenden System

Als Beispiel für eine enge Integration kann Microsofts OLE (Object Linking and Embedding) Technologie angeführt werden (vgl. Brockschmidt, 1995). Wird zum Beispiel ein Bild über OLE in ein Microsoft Word Dokument eingefügt, verschiebt sich der Text entsprechend. Ist das Bild markiert, stehen weitere Menüpunkte der das Bild anzeigenden Grafikapplikation im normalen Microsoft Word Menü integriert zur Verfügung.

Ein extremes Beispiel für eine vertikal geschichtete und parallel getrennte Integration liefert das TeleTeachingTool von Peter Ziewer und Helmut Seidl (2002). Hier läuft das System zur

Kursskriptverwaltung auf einem Rechner und die Annotationskomponente auf einem anderen. Der Rechner mit der Annotationskomponente zeigt als Hintergrund den Bildschirm des Rechners mit dem Kursskript an und erlaubt dem Benutzer die entfernte Steuerung dieses Rechners. Vorteil dieser Technologie ist, dass im System zur Darstellung des Kursskripts keinerlei Änderungen für die Annotationsunterstützung notwendig sind. Erkauft wird dieser Vorteil, neben gestiegenen Hardwareanforderungen, über aufwendige Verfahren zur Rekonstruktion der Navigation im Kursskriptsystem auf Seite des Annotationssystems, um eine zeitlich entkoppelte, direkte Verbindung zwischen Kursskript und Annotation herstellen zu können.

Die im Rahmen dieser Arbeit entwickelte Architektur erlaubt alle Varianten. Dies wird über die in Kap. 5.1.4.1., S. 75 angesprochene Trennung der Businesslogik von den Schnittstellen zur Umgebung des Annotationssystems erreicht. So gibt es für die Steuerungselemente die abstrakten Klassen *Menu*, *MenuItem* und *Icon*. Beim Starten der Annotationsanwendung wird dynamisch die jeweilige, auf die Umgebung abgestimmte Unterklasse instanziiert. Wird keine geeignete gefunden, so wird eine Standardklasse geladen. Diese Standardklasse arbeitet nach dem Prinzip *parallel getrennt*. Sie stellt ein eigenes Menüfenster mit vom Kursskriptsystem unabhängigen Einträgen zur Verfügung.

Technisch wird bei der Darstellungsfläche ebenso verfahren. Konzeptuelle Feinheiten werden im folgenden Abschnitt erläutert.

5.1.4.3. Trennung von Annotationserstellung und Annotationsdarstellung

Die Darstellungsfläche wird im Rahmen der Erstellung und der Darstellung der Annotationen meist zusammen mit dem Kursskriptverwaltungssystem genutzt. Zuerst wird die im ersten Augenblick nicht offensichtlich erscheinende, komplette Trennung von Erstellung und Darstellung thematisiert. Anschließend widmet sich dieser Abschnitt der vorgeschlagenen technischen Umsetzung unter Berücksichtigung des Kursskriptverwaltungssystems.

Die Erstellung einer Annotation ist im Gegensatz zur Darstellung für den Benutzer ein eventuell komplexer Vorgang. Gerade das elektronische Umfeld erlaubt andererseits eine mannigfaltige Unterstützung des Erstellungsprozesses. Während der Erstellung einer Annotation können dem Benutzer zum Beispiel verschiedenste Hilfsmittel angeboten werden: Lineale und Raster bei Veranschaulichungen, Rechtschreibkorrektur und Wortergänzungen bei Texteingaben und Hilfslinien bei Konstruktionen. Diese Hilfsmittel sollten bei der späteren Darstellung der Annotationen nicht mehr eingeblendet werden, da sie zu diesem Zeitpunkt nurmehr die visuelle Komplexität der Darstellung erhöhen.

Die Unterstützung des Dozenten verdient bei der Erstellung von Annotationen ein gesondertes Augenmerk. Findet das Erstellen der Annotationen nämlich während einer Sitzung statt, muss der erforderliche administrative Aufwand minimal sein. Die Studenten sollen nicht durch vom Dozenten durchgeführtes unnötiges Navigieren, Umschalten und Auswählen von Werkzeugen vom eigentlichen Stoff abgelenkt werden. Außerdem ist eine hohe Ausfallsicherheit von besonderer Wichtigkeit. Das System beim Dozenten sollte in jedem Fall auch ein Stifteingabegerät unterstützen. Wie aus den Folgerungen in Kap. 2.2.3., S. 21 hervorgeht, können mit diesem Hilfsmittel eine Vielzahl der Erscheinungsformen von Annotationen abgedeckt werden. Die Kreativität und damit auch Reaktionsfähigkeit des Dozenten während der Sitzung wird kaum eingeschränkt. Eine Tastatur ist hier zumeist unangebracht.

Damit das System auch von Studenten für ihre Annotationen genutzt wird, ist eine Integration in deren Arbeitsweise wichtig. Da die Studenten das Kursskript über elektronische Medien erreichen, kann hier durchaus die Eingabe der Annotationen über eine Tastatur angenommen werden. Dabei ist aber in jedem Fall ein freies Platzieren oder zumindest ein eindeutiges Verankern der Textannotation zu erlauben. Eine Stifteingabe ist momentan nicht die Regel, sollte aber auch vorgesehen werden.

Die beiden letzten Absätze zeigen, dass die Annotationserstellungskomponente so flexibel angelegt sein muß, dass es möglich ist, verschiedenen Benutzern unterschiedliche Erstellungswerkzeuge für gleiche Annotationstypen anzubieten.

Bei der Darstellung einer Annotation ist es wünschenswert, auf verschiedene Gegebenheiten der potentiell unterschiedlichen Zielsysteme der einzelnen Benutzer eingehen zu können. So ist vorstellbar, dass ein Student das Kursskript und die Annotationen unterwegs auf einem Handheld Computer mit kleiner Anzeige ansieht und zuhause dafür einen Arbeitsplatzrechner mit erheblich größerem Bildschirm nutzt. Hier sollte die Anzeige einer Annotation durch Wahl einer geeigneten Darstellung angepasst werden können.

Der letzte Absatz zeigt also, dass für eine Annotation mehrere Darstellungen möglich sein müssen.

Außerdem werden auch Annotationen angezeigt, die nicht selbst erstellt wurden und auch das Erstellungswerkzeug dazu bewusst fehlt. Zum Beispiel wird eine Annotation für eine Fehlerkorrektur eventuell dem Dozenten vorbehalten bleiben, aber trotzdem natürlich auch bei den Studenten angezeigt.

Um all diese Anforderungen abzudecken, wird in dieser Architektur das Entwicklungsmuster aus Abbildung 15 für die Annotationserstellung benutzt.

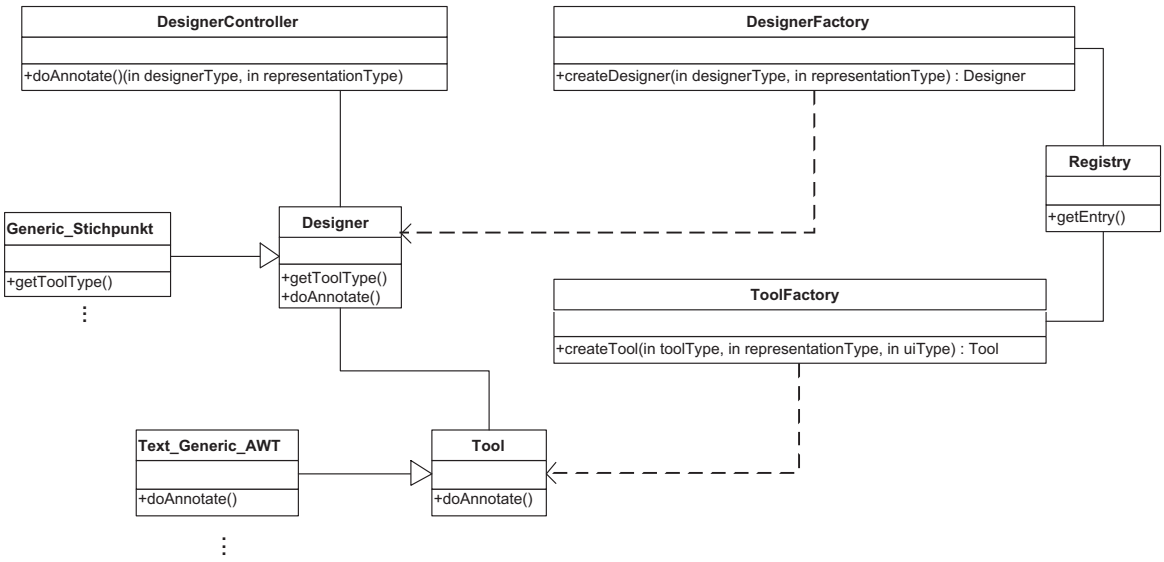


Abb. 15: Entwicklungsmuster für die Annotationserstellung

Einer guten Wiederverwendbarkeit wegen handelt es sich um ein zweistufiges Konzept: Die Annotation wird über ein *Designer*-Objekt, welches wiederum ein *Tool*-Objekt beinhaltet, erstellt. Insgesamt wird die Estellung einer Annotation zum restlichen System hin über ein *DesignerController*-Objekt repräsentiert. Dieses Objekt bekommt, vom Benutzer über die

Benutzungsoberfläche angestossen, durch den Aufruf der Methode `doAnnotate()` die Aufforderung eine Annotationserstellung zu starten. Zwei Parameter bestimmen die Art der Annotation: Der `designerType` und der `representationType`. Über `designerType` wird die Umgebung und damit die Interaktionsmechanismen bestimmt. *Generic*, *FatClient* und *Servlet* sind denkbare Varianten. Über *Generic* wird beispielsweise kodiert, dass die Darstellungsfläche des Kursskriptsystems nicht mitgenutzt werden kann (vertikal geschichtete Darstellungsflächen). `representationType` legt fest, welche Erscheinungsform einer Annotation zu erstellen ist (vgl. Kap. 2.2., S. 11). Basierend auf diesen beiden Parametern und globalen Einträgen in einer Registry erstellt die *DesignerFactory* ein entsprechendes *Designer*-Objekt.

Das *Designer*-Objekt kümmert sich nun während der Annotationserstellung um die weitere Interaktion mit dem Benutzer. Dazu erstellt das *Designer*-Objekt über die *ToolFactory* mindestens ein *Tool*-Objekt. Werden dem Benutzer weitere Hilfsmittel wie Lineale oder Raster angeboten, sind diese über weitere *Tool*-Objekte verwirklicht und die Synchronisation über das *Designer*-Objekt geregelt. Die *ToolFactory* entscheidet über die konkrete zu instanziierende *Tool*-Klasse anhand dreier Parameter: `toolType, representationType` und `uiType`.

`toolType` wird vom konkreten *Designer*-Objekt festgelegt und gibt die Art des Werkzeugs an (vgl. Tab. 1, S. 22). `representationType` entspricht dem beim *Designer*-Objekt. Ausserdem wird über `uiType` die Art der Benutzungsoberfläche für die Annotationen(!) angegeben. Im Java-Umfeld ist beispielsweise *AWT* oder *Swing* denkbar. Der konkrete Wert ist global festgelegt (z.B. Registry). Das *Designer*-Objekt ist damit also noch unabhängig von der verwendeten Benutzungsoberfläche. Ein Wechsel der Benutzungsoberfläche erfolgt also einzig durch die Wahl der entsprechenden *Tool*-Klassen.

Das *Tool*-Objekt kapselt damit alle Aspekte des konkreten Werkzeugs (vgl. Kap. 2.2.3., S. 21) zur Erstellung der Annotation. Ein *Tool*-Objekt kann über dieses Verfahren bei verschiedenen *Designer*-Objekten wiederverwendet werden. So ist es möglich, ein Text-Tool von einem Stichpunkt- und einem Erläuterungs-Designern einsetzen zu lassen. Die Objekte welcher konkreten *Designer*- beziehungsweise *Tool*-Klasse instanziiert werden, entscheiden die jeweiligen, statischen Factoryklassen anhand der übergebenen Parameter und einiger Einstellungen der Registry auf dem lokalen System. Über die Registry kann somit für jedes System individuell festgelegt werden, welche Klassen genutzt werden. Insbesondere können hierüber auch spezielle Ausgabemedien berücksichtigt werden.

Ein Geschwindigkeitsproblem ist trotz dieses hochgradig dynamischen Vorgehens nicht zu erwarten. Ein Benutzer erstellt immer nur eine Annotation gleichzeitig. Das dynamische Erstellen der meist maximal zwei Objekte fällt somit im Antwortverhalten der Applikation kaum auf.

Das verwendete Entwicklungsmuster für die Annotationsdarstellung hat den in Abbildung 16 gezeigten Aufbau. Das *PresenterController*-Objekt stellt wieder die Schnittstelle zum restlichen Annotationsystem dar. Bekommt es über den Methodenaufruf `presentAnnotation(Annotation a)` den Auftrag, dem Benutzer eine Annotation zu präsentieren, erstellt es über das *PresenterFactory*-Objekt ein geeignetes *Presenter*-Objekt. Die notwendigen Informationen werden dabei direkt vom anzuzeigenden Annotationsobjekt erfragt. Da beim Darstellen keine weitere Synchronisationsebene, wie bei mehreren Erstellungshilfsmitteln (Zeichenwerkzeug, Raster, Lineal, ...) notwendig ist, reicht ein einstufiges Konzept.

Für eine Beschreibung des genauen Ablaufs sei auf die Ausführungen bei der konkreten Architektur verwiesen (vgl. Kap. 5.2.1.4.).

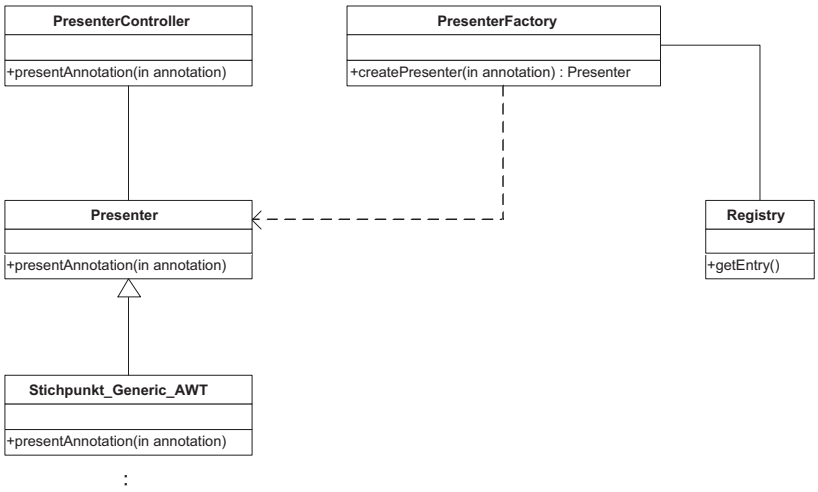


Abb. 16: Entwicklungsmuster für die Annotationsdarstellung

5.1.4.4. Verteilen der Annotationen

Neben dem bloßen Erstellen von Annotationen wird die erarbeitete Architektur auch das gemeinsame Nutzen von Annotationen erlauben. Die diesbezüglichen Konzeptentscheidungen werden nun kurz vorgestellt.

Eine erste wichtige Festlegung ist, dass es keine Unterscheidung zwischen privat und öffentlich direkt bei der Annotation selbst gibt. Diese Aufteilung ergibt sich, durch die im weiteren Verlauf vorgestellten Mechanismen zur Verteilung der Annotationen und den Ausführungen zu den Gruppenmechanismen.

Die Anforderungen A4, A5 und A6 betreffen direkt den Mechanismus zur Verteilung der Annotationen (vgl. Kap. 5.1.1., S. 70). Die Anforderung A4 - *Eigene Annotationen sind immer auch ohne Netzzugriff machbar und lesbar* impliziert, dass sich alle Annotationen, die ein Benutzer gemacht oder übernommen hat direkt auf seinem Rechner befinden (vgl. Folgerung F4). Jeder Benutzer besitzt also einen lokalen Annotationsspeicher. Damit ist es möglich, dass zwar die Annotationen lokal verfügbar sind, nicht jedoch das Kursskript, über dessen Ort keine weiteren Vorgaben festgelgt sind. Der Benutzer kann also unabhängig vom Kursskript immer auf die Annotationen (zum Beispiel über einen Annotationsbrowser, vgl. Kap. 5.2.1.5.) zugreifen.

Wegen Anforderung A5 - *Annotationen sind zwischen Benutzern austauschbar* müssen Annotationen zusätzlich auch über ein Netzwerk zur Verfügung gestellt werden und falls eine fremde Annotation durch den Benutzer übernommen wird, in den eigenen lokalen Annotationsspeicher geholt werden.

Die Anforderung, Annotationen austauschen zu können, lässt sich über mehrere verschiedene Ansätze verwirklichen. Prinzipiell bieten sich hier folgende Grundformen an:

- Client-Server
- Peer-to-Peer
- Mischform

Alle drei Ansätze werden nun vorgestellt und kurz diskutiert. Im Anschluss fällt die Entscheidung für einen Ansatz.

Ein Client-Server Ansatz bedingt, eine eigene zentrale Softwarekomponente losgelöst von jedem Benutzer, den Server. Der Server befindet sich auf einem für alle Benutzer mit Netzzugriff erreichbaren Rechner. Auf den Benutzerrechnern läuft jeweils ein Client. Wird auf einem Benutzerrechner eine eigene Annotation im lokalen Annotationsspeicher abgelegt, so wird die Annotation im Hintergrund über die Client-Komponente auch zur Serverkomponente übertragen. Prinzipiell werden alle, also auch private Annotationen an den Server übertragen. Vom Server aus wird die Annotation dann je nach Zugriffsregelung zu interessierten Clients übertragen. Dies geschieht durchaus auch erst zu einem späteren Zeitpunkt, also auch wenn das erstellende System selbst nicht mehr im Netzwerk erreichbar ist (asynchron). Über den zentralen Server, der prinzipiell immer zur Verfügung steht, ist dies kein Problem.

Erstellt ein Benutzer Annotationen während er keinen Netzzugriff hat, werden sie nur im lokalen Annotationsspeicher abgelegt. Sobald eine Netzverbindung besteht, werden die neuen Annotationen dann auch auf den zentralen Speicher übertragen. Damit einfach zu entscheiden ist, welche Annotationen neu und damit zu übertragen sind, werden neue Annotationen im lokalen Annotationsspeicher markiert. War eine Übertragung an den Server erfolgreich, wird diese Markierung entfernt.

Vorteile einer Client-Server Lösung:

- Öffentlich zugängliche Annotationen liegen alle an einer genau definierten Stelle. Statistiken, Auswertungen und Durchsicht durch Tutoren oder Dozenten sind somit einfach zu realisieren.
- Der Zugriff auf die Annotationen kann umfassender protokolliert werden.
- Der Zugriff kann effektiver gegen unerlaubte Zugriffe gesichert werden.
- Eine Archivierung ist einfach zu realisieren.
- Insgesamt einfach zu Implementieren
- Zugriff auf Annotationen anderer sind, auch wenn der Ersteller gerade nicht im Netz erreichbar ist, über den Server möglich (asynchron)

Nachteile einer Client-Server Lösung:

- Single point of failure. Fällt der Server aus, können keine Annotationen mehr übertragen werden. Eigene Annotationen sind weiterhin machbar.
- Treffen sich einzelne Studenten zum Nachbereiten des Kurses und haben keine Verbindung zum Server, können sie keine Annotationen austauschen, auch wenn sie ein lokales Netzwerk zwischen ihren Rechnern einrichten.

Bei einer Peer-to-Peer Lösung prüft der einzelne Rechner beim Start, ob schon andere Rechner im Netz arbeiten. Wenn ja, übermittelt er allen eine Liste der bei ihm lokal zur Verfügung stehenden Annotationen. Kommt eine neue Annotation dazu, wird auch diese Information an alle weitergegeben. Will nun der Benutzer an einem Rechner eine Annotation eines anderen übernehmen, fordert sein Rechner diese Annotation direkt beim entsprechenden Rechner an und kopiert sie in den lokalen Annotationsspeicher.

Vorteile einer Peer-to-Peer Lösung:

- Fallen einzelne Knoten aus, können alle anderen trotzdem weiterarbeiten und Annotationen austauschen.
- Es können jederzeit und überall kleine Adhoc-Netze gebildet und darüber Annotationen getauscht werden.

Nachteile einer Peer-to-Peer Lösung:

- Eine globale Archivierung ist aufwendig zu realisieren.
- Eine globale Protokollierung ist aufwendig zu realisieren.
- Globale Auswertungen und Statistiken sind kaum möglich, da selten alle Knoten auf einmal erreichbar sind.
- Zugriff auf Annotation eines anderen Benutzers sind nur möglich, wenn sich dessen Rechner auch zeitgleich im selben Netz befindet.

Der im Lehrumfeld nicht zu verachtende Vorteil der Bildung von Adhoc-Gruppen beim Peer-To-Peer kann durch eine Mischung der beiden Ansätze mit den Vorteilen der Client-Server Lösung kombiniert werden. Der Ablauf in Abbildung 17 stellt die Mischform beider Ansätze grafisch dar.

Startet ein Benutzer sein Annotationssystem, prüft es die Erreichbarkeit des zentralen Servers. Ist dieser erreichbar, arbeitet das System nach dem ersten Ansatz. Ist kein zentraler Server erreichbar, wird geprüft, ob ein netzlokaler Server erreichbar ist. Ist auch dies nicht der Fall, kann davon ausgegangen werden, dass das gestartete System momentan das einzige in diesem Teilnetz ist. Es startet zusätzlich einen netzlokalen Server. Meldet sich ein weiteres System in diesem Netz an, findet es nun zwar auch keinen zentralen Server. Dafür steht ein

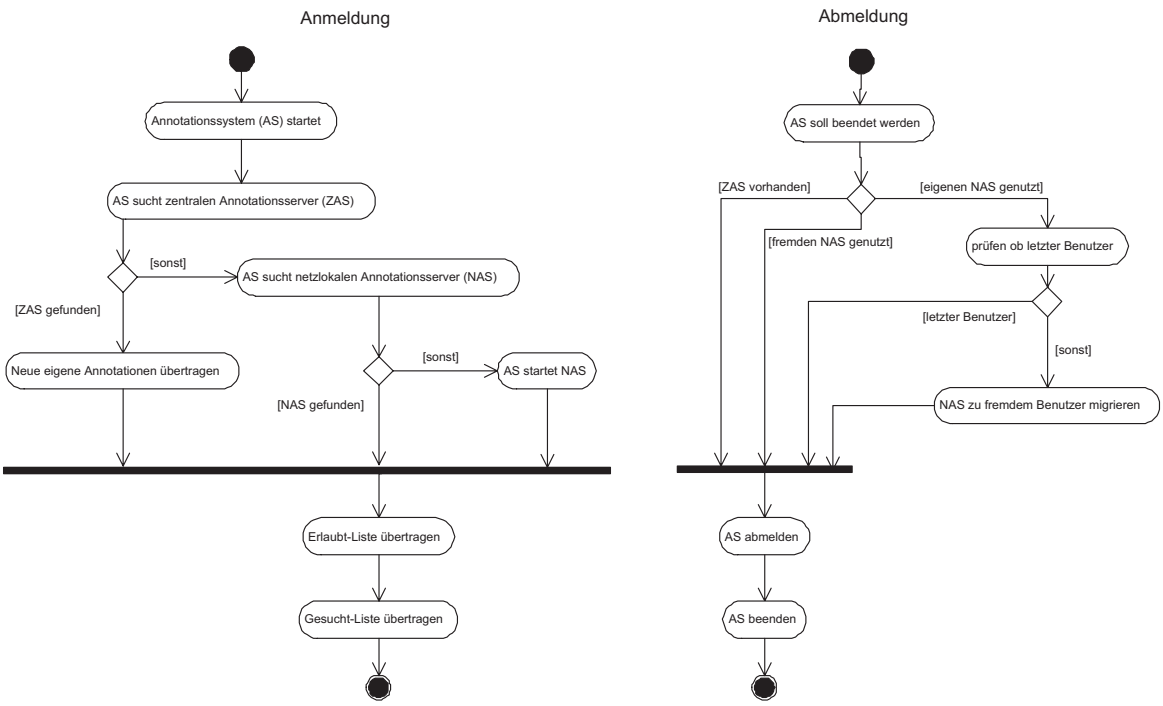


Abb. 17: Schema zur Verteilung von Annotationen

netzlokaler Server zur Verfügung. Über diesen tauschen die beiden Knoten und alle weiteren hinzukommenden Annotationen im unter Client-Server beschriebene Verfahren aus, wobei beim Übertragen zu einem netzlokalen Server die Markierung, dass eine Annotation neu ist, im Ursprungssystem nicht entfernt wird. Wird in einer späteren Sizung wieder ein zentraler Server erreicht, werden alle hinzugekommenen Annoationen übergeben.

Beendet sich ein hinzugekommener Knoten, hat dies keinen Einfluss. Beendet sich das System mit dem netzlokalen Server, startet ein anderes, im Netz verbleibendes System, nach Aufforderung durch das beendende System einen netzlokalen Server als Ersatz. Dieser neue Server übernimmt vor Beendigung des ursprünglichen netzlokalen Servers dessen Annotationen. Der netzlokale Server migriert also. Wird das letzte System in diesem Netz beendet, beendet sich auch der netzlokale Server. Eine Sicherung der Annotationen über das Beenden des netzlokalen Servers hinaus findet nicht statt. Dies ist nicht notwendig, da jeder Knoten seine eigenen Annotationen und die übernommenen jeweils lokal gesichert hat.

Meldet sich ein System später wieder erfolgreich an einem Netz mit dem zentralem Server an, übermittelt es wie im ersten Ansatz geschildert alle eigenen Annotationen.

Im Rahmen dieser Arbeit wird des Lehrumfeldes wegen die vorgestellte Mischform umgesetzt.

5.1.4.5. Global eindeutige ID's

Aus den Anforderungen A4 und A5 und dem daraus entstehenden Verteilungsmechanismus für Annotationen (vgl. vorhergehenden Abschnitt) ergibt sich gesamt gesehen eine verteilte Anwendung mit verteilter Datenhaltung und Replikationsmechanismus.

In diesem Architekturvorschlag bekommt jedes Datenobjekt, wie im praktischen Arbeiten mit Datenobjekten üblich, eine eindeutige ID zugewiesen, egal ob es sich in der konkreten Implementierung um eine hierarchische Ablage in XML oder eine relationale Ablage in einer SQL Datenbank handelt. Diese ID muss im gesamten System für die Datenobjekte pro Kategorie eindeutig sein. Dafür gibt es zwei Möglichkeiten:

- Benutzen von allgemeinen Algorithmen zur Erzeugung so genannter GUID's (Globally Unique Identifier, Microsoft) oder UUID's (Universal Unique Identifier, OpenGroup).
- Stufenkonzept und Arbeiten mit vorläufigen ID's

Mit der Verbreitung verteilter Anwendungen, speziell wenn sie aus Komponenten verschiedener Anbieter bestehen, wurde es notwendig, den Datenobjekten über die Modul- und Rechnergrenzen hinweg, auch ohne zentrale Komponente eindeutige Identifikatoren zuordnen zu können. Für diesen Zweck wurden mehrere Verfahren vorgestellt (vgl. z.B. Open Group, 1997) und implementiert (vgl. z.B. Guidgen.exe erklärt in Brockschmidt, 1995, java.rmi.dgc.VMID im Lieferumfang Java 2 Plattform, Standard Edition v. 1.3.1). Alle diese Verfahren haben gemeinsam, dass sie aus einer zeitlichen und einer räumlichen Komponente eine ID generieren, die über dieses Verfahren mit extrem hoher Wahrscheinlichkeit kein zweites Mal entsteht.

Da dieser Architekturvorschlag eine zentrale Komponente vorsieht, über die Annotationen im Regelfall ausgetauscht werden, ist auch eine andere Implementierung vorstellbar: Wird ein Objekt auf einem Benutzersystem lokal erstellt, so erhält es eine vorläufige, lokal eindeutige ID. Dies lässt sich unschwer über einen Zähler realisieren. Solange das System nicht mit anderen kommuniziert, gibt es keine Probleme mit Duplikaten. Tritt das System mit anderen

in Kontakt, also zum Beispiel dem zentralen Server oder einem netzlokalen Server (vgl. Kap.5.1.4.4.), so werden alle vorläufigen ID's vor dem Übertragen von der Kommunikationskomponente mit dem Identifikator des Rechners erweitert. Dabei wird auch der lokale Identifikator durch den neuen, zusammengesetzten ersetzt. Durch die äußere Form ist entscheidbar, ob der Identifikator schon ersetzt wurde oder nur lokal eindeutig ist. Jeder Rechner bekommt seinen Identifikator bei der erstmaligen Anmeldung am zentralen Server. Dies würde bedingen, dass jeder Rechner vor dem Austauschen von Annotationen einmal am zentralen Server angemeldet sein müsste, um einen Identifikator des Rechners zu erhalten.

Dieser Einschränkung und dem zusätzlichen Ersetzungsaufwand wegen, wurde in diesem Architekturvorschlag der Weg über UUID's gewählt. Durch Zurückgreifen auf eine der vorhandenen Methoden (z.B. Java.rmi.dgc.VMID) ist der Aufwand dafür minimal. Da die UUID's global eindeutig sind, sind sie natürlicherweise auch lokal eindeutig. Somit sind alle Anforderungen erfüllt.

5.1.4.6. Gruppenmechanismen

Das bei der Untersuchung der bestehenden Systeme erkannte Defizit bei Gruppenannotationen (vgl. Kap. 4.1.7. und Kap. 4.2.4.) wird in der präsentierten Architektur getilgt. Konkret werden die beiden Gruppenkonzepte *Gruppenannotation* (vgl. Kap. 3.3.3.) und *kollaborative Annotation* (vgl. Kap. 3.3.4.) umgesetzt.

Die Umsetzung der Gruppenannotationen in diesem Architekturvorschlag sieht vor, dass alle Mitglieder einer Gruppe vom System proaktiv auf eine neue Annotation eines Gruppenmitglieds aufmerksam gemacht werden. Die Serverkomponenten (zentral und netzlokal) sind dafür als Notification Server ausgebildet (vgl. Xu, C., 2000). Dies erweitert die Serverkomponente um Notifikationsfunktionalität. Ein Benutzersystem registriert sich bei einem Notification Server für den Erhalt gewisser Meldungen. Ab diesem Zeitpunkt wird das Benutzersystem vom Notification Server über das Eintreffen solcher Meldungen informiert. Konkret handelt es sich hier um Meldungen über neu eingehendeAnnotationen.

Insgesamt sind von der Architektur zwei Aspekte zu erfüllen:

- Ein Mechanismus zum Definieren einer Gruppe muss implementiert werden.
- Die Information über eine neue Annotation muss propagiert werden.

Außerdem wird davon ausgegangen, dass der zentrale Server eine Liste aller bei ihm jemals angemeldeten Benutzer führt. Der netzlokale Server mit seinem temporären Charakter unterhält eine Liste aller Benutzer, die aktuell mit ihm in Verbindung stehen.

Im hier präsentierten Architekturvorschlag entsteht eine Gruppe implizit. Ein Benutzer gibt an, über wessen Annotationen er informiert werden möchte. Die so entstehende Liste je Benutzer wird im Weiteren *Gesucht-Liste* genannt. Diese Gesucht-Liste wird auf dem Rechnersystem des Benutzers abgelegt. Außerdem wird diese Liste beim Erstellen einer Verbindung zum zentralen Server dort repliziert. Arbeiten Benutzer über einen netzlokalen Server zusammen, werden ihre Listen dort auch temporär repliziert. Das Rechnersystem des Benutzers dient also als „mobiler" Speicherplatz. Die jeweiligen Server benutzen die Listen für das Propagieren der Annotationsinformation, solange ein Benutzer mit dem Server verbunden ist.

Neben dieser Gesucht-Liste geben die Benutzer auch an, wem sie Zugriff auf die eigenen Annotationen geben möchten. Diese Information wird ebenfalls in einer Liste, der *Erlaubt-Liste* abgelegt. Die Verwaltung dieser Liste entspricht der der Gesucht-Liste.

In beiden Listen ist es auch möglich, nicht nur konkrete Benutzer einzutragen. Es können auch fiktive Benutzer eingetragen werden. Damit wird der vorgestellte Mechanismus ohne Zusatzaufwand um ein Benutzergruppenkonzept erweitert: Jeder fiktive Benutzer repräsentiert ein Benutzergruppe. Im System vordefiniert sind: *Jeder*, *Dozenten* und *Studenten*.

Nun haben die Serverkomponenten die Möglichkeit, über ein Abgleichen der beiden Listen festzustellen, welche Annotationsinformationen tatsächlich weitergegeben werden soll. Zu erwähnen ist, dass die durch die beiden Listen definierte Relation nicht kommutativ ist. Dies ist absichtlich so festgelegt: Ein Benutzer mit einem ausgeprägten Sendungsbewusstsein ist nicht von vorne herein an den Annotationen aller anderen interessiert. Speziell trifft dies auch den Dozenten. Er gibt seine Annotationen eventuell direkt an alle Studenten frei, was aber noch lange nicht heißt, dass er auch an allen Annotationen aller Studenten interessiert ist.

Für das Propagieren einer neuen Annotation gibt es zwei Fälle:

- synchron
- asynchron

Sind zwei Benutzer in einer über die beiden Listen definierten Gruppe zeitgleich mit dem zentralen Server verbunden und erstellt der „liefernde“ Student eine Annotation, wird diese nach der Ablage im lokalen Annotationsspeicher des Benutzers direkt an den jeweiligen Server weitergegeben und somit repliziert. Anhand der Listen gibt der Server die Information über die Annotation dann auch sofort an alle, momentan verbundenen und interessierten Benutzer weiter. Dies entspricht der synchronen Meldung über eine neue Annotation.

Eine asynchrone Meldung entsteht, wenn entweder der Adressat zum Zeitpunkt der Erstellung der Annotation nicht mit der Serverkomponente in Verbindung stand oder der Absender der Meldung keine Verbindung zum Server hatte.

Annotationen werden beim Erstellen wie oben beschrieben zuerst beim Ersteller lokal abgelegt. Anschließend wird versucht, die Serverkomponente von der neuen Annotation zu unterrichten und die Annotation gegebenenfalls zur Replikation zu übergeben. Ist die Serverkomponente nicht erreichbar, wird die Annotation im lokalen Speicher als *nicht abgeglichen* markiert. Kann zu einem späteren Zeitpunkt eine Verbindung zu einer Serverkomponente erstellt werden, so werden die nicht abgeglichenen Annotationen zur Replikation übergeben. Erkennt die Serverkomponente anhand der Gesucht- und Erlaubt-Listen Informationsbedarf, erzeugt sie eine nun asynchrone Meldung über eine neue Annotation und gibt sie an die Interessenten weiter. Nur wenn die Annotation an die zentrale Serverkomponente übergeben wurde, wird die Markierung *nicht abgeglichen* entfernt.

Ist der Empfänger einer Meldung über eine neue Annotation nicht erreichbar, so geschieht nichts weiteres. Wenn dieser Empfänger sich dann zu einem späteren Zeitpunkt an der zentralen Serverkomponente anmeldet, werden alle Meldungen über neue Annotationen, an denen der Empfänger interessiert ist, nachgereicht. Um nicht bei jeder Meldung für jeden potentiellen Empfänger hinterlegen zu müssen, ob er die Meldung schon erhalten hat, wird bei der zentralen Serverkomponente wie folgt vorgegangen: Alle Annotationen werden beim Speichern auf Seite des zentralen Servers zusätzlich aufsteigend, eindeutig nummeriert (Zähler). Schließt ein Client eine Verbindung zum zentralen Server, erhält er den aktuellen Stand dieses Zählers und speichert diesen lokal ab. Anhand diesen alten Zählerstands können die neue hinzugekommenen Annotationen beim nächsten Anmelden erkannt werden. Diese Möglichkeit besteht nur beim zentralen Server, da er alle Annotationen sammelt. Ein netzlokaler Ser-

ver kann nicht nach diesem Prinzip vorgehen. Eine Lösung hierfür wäre wieder das Ergänzen jeder Annotation um die Benutzer, die eine Nachricht über diese Annotation erhalten haben. Dieser recht hohe Aufwand nur der proaktiven Meldung wegen ist kaum zu rechtfertigen. Außerdem wird davon ausgegangen, dass bei Bildung von Adhoc-Netzen mit netzlokalem Server das synchrone Zusammenarbeiten im Vordergrund steht. Synchrone Meldungen werden problemlos erstellt. Möchte ein Benutzer Informationen über alte Annotationen eines anderen Benutzers erhalten, kann er selbst aktiv werden und die später in der konkreten Architekturbeschreibung beschriebene Such- und Browsekomponente benutzen.

Neben den Gruppenannotationen werden auch kollaborative Annotationen angeboten. Bei kollaborativen Annotationen arbeiten mehrere Benutzer an einer gemeinsamen Annotation. Sie bilden damit ein *Team*. Zur Abgrenzung *Team* - *Gruppe* sei auf Teufel, Sauter, Mühlherr & Bauknecht (1995) verwiesen.

Auch für die Unterstützung kollaborativer Annotationen sind wieder zwei Schritte erforderlich:

- Das Team muss formiert werden.
- Die für die Zusammenarbeit notwendige Kommunikation muss ermöglicht werden.

Im Gegensatz zum obigen Vorgehen zur Formierung einer *Gruppe*, wird ein *Team* in der vorgestellten Architektur explizit angegeben. Dazu hat jeder Benutzer die Möglichkeit Teams anzulegen und andere Benutzer in ein Team einzuladen. Im Lehrumfeld ist es zudem denkbar, dass der Dozent Teams für eine Gruppenarbeit festlegt. Andererseits können sich auch Studenten über den Einladungsmechanismus zu eigenen Teams zusammenschließen, um beispielsweise den darüber entstehenden zusätzlichen Kommunikationskanal für Peer-Diskussionen zu nutzen. Teams haben bis zur expliziten Auflösung durch die Benutzer bestand.

In dem ersten Ansatz dieser Arbeit wird nur synchrone Zusammenarbeit über den zentralen Server unterstützt. Damit liegt die Verwaltung der Teams beim zentralen Server. Begründet wird dies damit, dass Adhoc-Gruppen mit *netzlokalem* Server meist bei *direktem* Zusammentreffen der Beteiligten entstehen. Somit stehen auch andere direkte Kommunikationswege zur Verfügung. Arbeiten die Beteiligten räumlich getrennt zusammen, nutzen sie sowieso ein Netzwerk, von dem aus mit hoher Wahrscheinlichkeit auch der zentrale Server erreichbar ist. Damit ist der Zusatzaufwand für die verteilte Datenhaltung der Teaminformation nicht gerechtfertigt.

Beim Starten einer konkreten, kollaborativen Annotation wird ein Team und eine Annotationsform gewählt. Für die Kommunikation bei der Erstellung der Annotation gibt es zwei Möglichkeiten:

- Die Kommunikation läuft über einen Abgleich des Annotationsobjektinhalts über die von der Architektur gestellte Infrastruktur

  Vorteil dieser Variante ist, dass prinzipiell jede Annotationsform für die Erstellung einer kollaborativen Annotation genutzt werden kann. Außerdem können verschiedene Teilnehmer unterschiedliche konkrete Erstellerobjekte benutzen, da diese nur den Annotationsinhalt interpretieren können müssen.
  Da die Kommunikation einzig über den Abgleich des Inhalts des Annotationsobjekts erfolgt, bedeutet dies aber auch, dass alle Komponenten der Infrastruktur, die eine Annotation weitergeben, auch in der Lage sein müssen, die Annotation in umgekehrter Richtung zurückzugeben. Außerdem müssen Transformationen, die dabei von einzelnen Kompo-

nenten während der Estellung vorgenommen werden, gegebenenfalls geeignet umgekehrt werden. Dieser Mehraufwand ist ein gewisser Nachteil.

- Die Erstellungskomponenten kommunizieren direkt (eventuell den zentralen Server als Proxy nutzend) miteinander.

  Vorteil dieser Alternative ist, dass der Abgleich nicht nur auf Ebene des Inhalts der Annotation möglich ist, sondern auch auf Ebene des Werkzeugs und der Erstellungshilfsmittel. Das von einem Studenten aktivierte Hilfsgitter würde zum Beispiel auch bei allen anderen erscheinen. Diese Variante kann bis zu einem Shared Window Ansatz ausgebaut werden.
  Als Nachteil kann gesehen werden, dass spezielle Annotationserstellungswerkzeuge entwickelt werden müssen. Außerdem müssten alle Teamteilnehmer das identische Annotationserstellungswerkzeug zur Verfügung haben.

Bei der konkreten Architektur wurde die erste Variante umgesetzt. Der zentrale Server dient dabei als Synchronisationsglied. Alle geänderten Annotationsinhalte der verschiedenen Benutzer laufen hier zusammen, werden hier bei zeitgleichen Veränderungen vereinigt und die resultierende Annotation an alle Teammitglieder verteilt. Die aktuelle Version einer Annotation ist also als diejenige auf dem Server definiert.

Die Vereinigung des Annotationsinhalts bei zeitlich überlappenden Veränderungen übernimmt eine eigene, bei Bedarf dynamisch geladene Klasse je Annotationsform. Damit ist die Erweiterbarkeit um neue Annotationsformen gewahrt.

5.1.4.7. Verbindung Annotation und Kursskript

Bevor der konkrete Architekturvorschlag präsentiert wird, bedarf es in jedem Fall noch der konzeptuellen Klärung einer Sache: Wie wird die Verbindung zwischen einer Stelle im Kursskript und einer Annotation hergestellt? Diese Verbindung ist notwendig, um beim Anzeigen einer Kursskriptseite auch die entsprechenden Annotationen anbieten zu können.

Um den Ergänzungsaufwand auf Seiten des bestehenden Kursskriptsystems gering zu halten, wird keinerlei Information über eine Annotation, auch nicht über die Platzierung, im Kursskriptsystem abgelegt. Damit der Benutzer trotzdem eine Kopplung der Systeme „erlebt“, sind folgende Funktionalitäten erforderlich (vgl. auch Kap. 5.1.2.):

- Methoden, die es der Annotationsunterstützung erlauben, auf die Kursskriptdarstellung Einfluß zu nehmen und den aktuellen Zustand abzufragen. Darunter fällt das inhaltliche Platzieren des Kursskripts, aber auch die Wahl der Darstellung für das Kursskript. Damit wird es möglich, von einer Annotation aus zur annotierten Stelle im Kursskript zu navigieren.
- Methoden, die die Annotationsunterstützung von Änderungen in der Darstellung beziehungsweise Platzierung des Kursskripts informiert. Dies entspricht also einem Ereignis, das vom Kursskriptdarstellungssystem im Annotationssystem ausgelöst wird. Mit Hilfe dieser Ereignisse kann das Annotationssystem dann die entsprechenden Annotationen zur im Kursskript präsentierten Information anbieten.

Aus diesen Ausführungen ergeben sich fünf konkrete Schnittstellenmethoden:

```
public void setPresentation( PresentationType type )
```

Das Kursskriptsystem auf eine bestimmte Darstellung einstellen.

`public void setScriptPosition( ScriptReference reference )`

Das Kursskriptsystem auf eine bestimmte Stelle im Kursskript positionieren

`public ScriptReference getScriptPosition( )`

Das Kursskriptsystem nach der aktuellen Position befragen. Wie bei den späteren Ausführungen zu `ScriptReference` erläutert, kann eine Position unterschiedlich große Bereiche referenzieren.

`public void presentationChanged( PresentationType type )`

Ändert sich die Darstellung des Kursskripts, informiert das Kursskript über diesen Aufruf im Annotationssystem das Annotationssystem.

`public void scriptPositionChanged( ScriptReference reference )`

Ändert sich die vom Kursskriptsystem angezeigte Position im Kursskript, beispielsweise durch Weiterblättern, so wird das Annotationssystem über den Aufruf dieser Methode darüber informiert.

Die ersten drei Methoden müssen vom bestehenden Kursskriptsystem implementiert werden. Die letzten beiden müssen bei entsprechenden Änderungen während der Laufzeit vom Kursskriptsystem im Annotationssystem aufgerufen werden, damit das Annotationssystem passend reagieren kann. Diese fünf Dinge bilden die mindestens notwendigen Ergänzungen im vorhandenen Kursskriptsystem, um mit der vorgestellten Architektur zusammenarbeiten zu können.

In `PresentationType` wird zum Einen das Darstellungssystem kodiert. Diese Information dient den in Kap. 5.1.4.3. vorgestellten Factory-Klassen zur Auswahl der passenden Tool-Klasse beim Erstellen und der passenden Presenter-Klasse bei der Darstellung. Über diesen Aspekt des `PresentationType`'s ergibt sich auch der Grad der Integration bei der Darstellungsfläche (vgl. Kap. 5.1.4.2.). Zum Anderen wird in `PresentationType` auch die aktuelle Größe der Darstellungsfläche und der Ursprung kodiert. Damit wird es den Presenter-Klassen möglich, gegebenenfalls entsprechende Skalierungen und Verschiebungen an den Annotationen vorzunehmen.

Der Parameter `SkriptReference` dient der Angabe einer Stelle im Kursskript. Trotz seiner klaren Aufgabe ist er recht komplex aufgebaut. Deshalb wird der Aufbau im Folgenden detailliert besprochen.

Konzeptuell sind für die Angabe einer Stelle im Kursskript zwei grundsätzlich verschiedene Formen möglich (vgl. Abb. 18):

- Eine layoutbasierte Referenz

  Hierunter fällt beispielsweise die Angabe einer Seitennummer und eventuell eine Koordinatenangabe innerhalb der Seite. Diese Angabe ist nicht an einen speziellen Inhalt des Kursskripts gekoppelt und kann zum Beispiel bei Änderung der Schriftgröße bei Proportionalschrift nicht praktikabel umgerechnet werden.

- Eine inhaltsbasierte Referenz

  Hier können abermals zwei grundsätzliche Varianten unterschieden werden: Eine Angabe auf Basis der inneren Struktur des Kursskripts. Dies entspräche beispielsweise der Angabe

„Abschnitt 5.3.1, 2. Absatz“. Die andere inhaltsbasierte Variante ist das Setzen einer Marke in den Inhalt. So könnte bei einer Definition zum Begriff „Relation“ zuvor die Marke „DEF_RELATION“ im Kursskript eingefügt werden. Zu erkennen ist, dass über strukturorientierte Kursskriptreferenzen auch ganze Bereiche im Kursskript referenziert werden können.

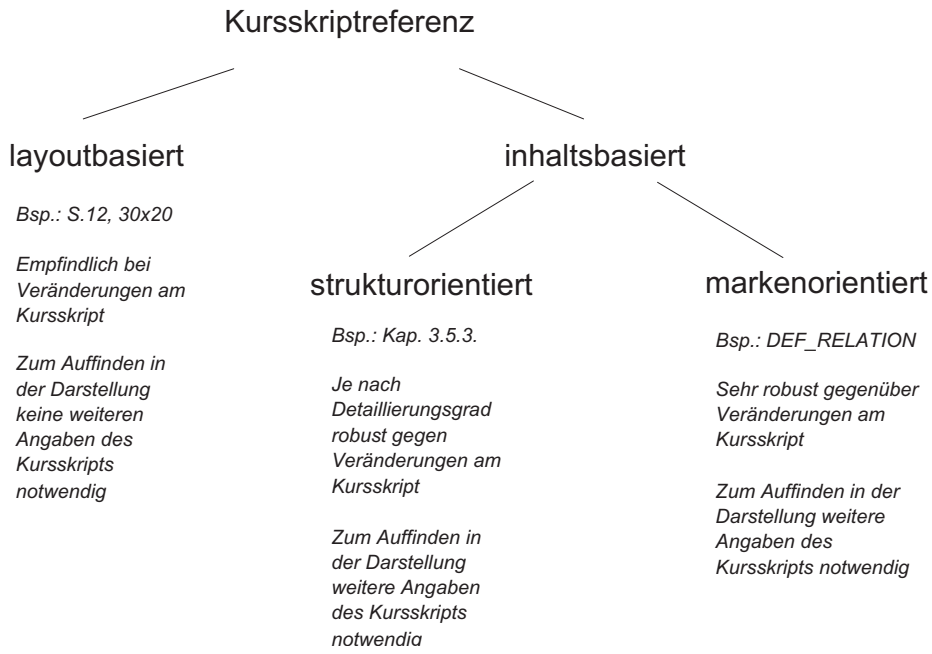


Abb. 18: Mögliche Formen von Referenzen ins Kursskript

Welche der beiden Formen technisch für die konkrete Anforderung besser geeignet ist, hängt zu einem hohen Prozentsatz von der Ausprägung der Integration der Darstellungsflächen der beiden Systeme ab (vgl. Kap. 5.1.4.2.). Die inhaltsbasierten Referenzen sind robuster gegen Veränderungen im Kursskript als die layoutbasierten. Andererseits erfordert eine inhaltsbasierte Referenz eine stärkere Integration der beiden Systeme, da das Annotationssystem zusätzliche Informationen vom Kursskriptsystem braucht, um eine exakte Koordinate für die Darstellung zu berechnen.

Da die vorgeschlagene Architektur beide Formen der Integration der Darstellungsflächen anbietet, bietet das Objekt ScriptReference die Möglichkeit, alle drei (Unter-)Formen parallel aufzunehmen. Die Angabe mindestens einer Form ist zwingend. Die jeweiligen Darstellungsklassen greifen dann den für sie notwendigen Eintrag heraus. Andererseits stehen dann alle Eintragungen auch für die geforderten Filter- und Suchmechanismen zur Verfügung.

Ein nicht offensichtlicher Aspekt, der vom Kursskriptsystem bei der Bereitstellung des ScriptReference Objekts beachtet werden muss ist folgender: Die Granularität der Angabe in der ScriptReference unterscheidet sich bei den Aufrufen `public ScriptReference getScriptPosition( )` und `public void scriptPositionChanged( ScriptReference reference )`.
Für die Rückgabe der Methode `getScriptPosition()` gilt so feingranular *wie möglich*.
Für den Parameter bei `scriptPositionChanged()` gilt so feingranular *wie nötig*.

Dies hängt mit der Nutzung dieser beiden Methoden zusammen. Beim Aufruf der Eventmethode `scriptPositionChanged()` durch das Kursskriptsystem entscheidet das Annotationssystem anhand der übergebenen `ScriptReference`, welche Annotationen für die aktuelle Ansicht des Kursskripts relevant sind. Die Angabe in `ScriptReference` sollte also den kompletten angezeigten Teil des Kursskripts möglichst gut beschreiben. Diese Information hat nur das Kursskriptsystem. Bei der layoutbasierten Angabe würde dann beispielsweise eine Angabe der Seite ohne Koordinaten reichen.

Die Methode `getScriptPosition()` hingegen ruft das Annotationssystem auf, wenn eine Annotation erstellt werden soll. Die hier vom Kursskriptsystem übergebene `ScriptReference` wird also zur Verankerung der Annotation benutzt. Sie sollte deshalb entsprechend genau sein. Hier bietet sich oftmals die Textcursorposition oder markierte Bereiche beim Starten der Annotation an.

Die Verbindung *Annotation - Kursskript* ist nicht die einzige Möglichkeit, eine Annotation zu verankern: Es besteht auch die Möglichkeit der Annotation einer Annotation. Beispiel dafür ist eine Fehlerkorrektur. Wurde eine fehlerhafte Annotation erstellt, so kann diese über eine Annotation der Annotation korrigiert werden. Ein direktes Editieren der Annotation ist nicht möglich, da andere Benutzer die Annotation eventuell schon übernommen haben und keine Veränderung wünschen (vgl. Anwendungsfall „Erstellen“, Kap. 5.1.3., S. 74). Für den Fall der Annotation einer Annotation bietet `ScriptReference` neben den oben angesprochenen Einträgen auch das Feld *VaterAnnotation*. Hier wird eine Referenz auf die annotierte Annotation hinterlegt.

5.2. Architektur

Nachdem die wichtigsten konzeptuellen Überlegungen vorgestellt sind, wird nun eine konkrete, in sich abgeschlossene Architektur präsentiert. Zu erwähnen ist, dass für diese Architektur im Zuge dieser Arbeit eine prototypische Referenzimplementierung in Java erstellt wurde. Für weiterführende konkrete Aspekte der Architektur, die in der vorliegenden Arbeit des Umfangs wegen offen bleiben, sei auf diese Referenzimplementierung verwiesen (vgl. Schütz, 2005b).

Der erste Abschnitt diesen Kapitels präsentiert die beteiligten Komponenten. Damit gewinnt der Leser einen ersten Überblick über die Gesamtarchitektur. Die anschließenden Abschnitte vertiefen schrittweise die Ausführungen über die Vorstellung der statischen Aspekte (Klassendiagramme, verwendete Datenstrukturen) hin zur Betrachtung einiger dynamischer Abläufe im System.

5.2.1. Beteiligte Komponenten

Abbildung 19 gibt einen Überblick über die Komponenten und ihre Platzierung in Teilsystemen.

Wie aus der Darstellung ersichtlich gliedert sich die vorgeschlagene Architektur in vier Teilsysteme:

- Kursskriptsystem

  Im weiteren Verlauf wird davon ausgegangen, dass dieser Teil schon vorhanden ist. Für die Integration der Annotationsfunktionalität müssen hier nur wenige Schnittstellen ergänzt werden.

- Lokales Annotationssystem

  Das lokale Annotationssystem umfasst alle Komponenten, die pro Benutzer nötig sind.

- Zentrales Annotationsverwaltungssystem

  Diese Teilsystem ermöglicht die Zusammenarbeit und den Austausch von Annotationen zwischen allen Benutzern eines Kurses.

- Netzlokales Annotationsverwaltungssystem

  Sollte das zentrale Annotationsverwaltungssystem nicht erreichbar sein, wird ein temporäres, netzlokales Annotationsverwaltungssystem gestartet. Damit können alle Benutzer, die dieses Teilsystem erreichen Annotationen tauschen und zusammenarbeiten. Im Gegensatz zum zentralen Annotationsverwaltungssystem legt das netzlokale keine Daten persistent ab.

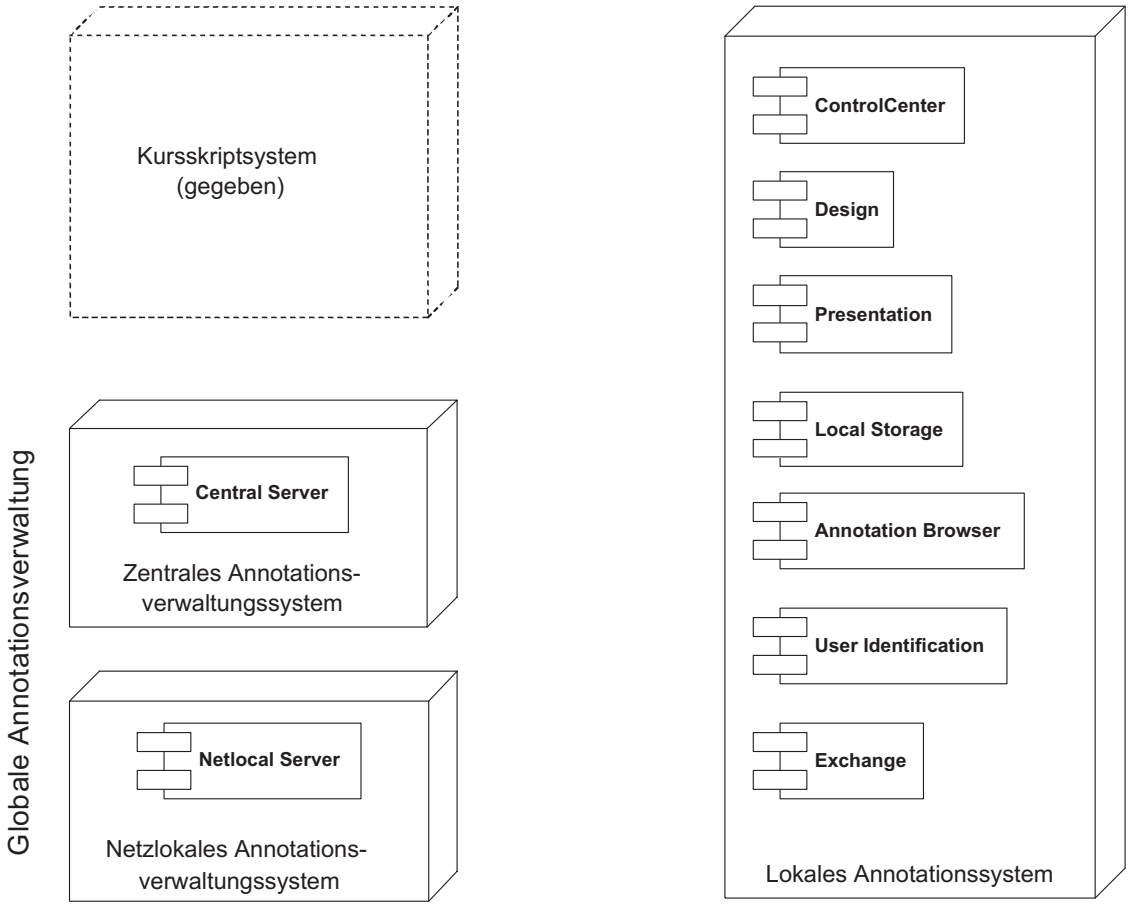


Abb. 19: Komponenten des Architekturvorschlags und ihre Platzierung in den Teilsystemen

Netzlokales und zentrales Annotationsverwaltungssystem bilden zusammen das globale Annotationsverwaltungssystem, wobei zu jedem Zeitpunkt in einem Netzabschnitt immer nur ein System aktiv ist.

Die einzelnen Komponenten der Teilsysteme und damit ihre jeweiligen Funktionen werden nun genauer erläutert. Ein detaillierter Gesamtüberblick in Form eines Klassendiagramms ist im Anhang S. 172 gegeben.

5.2.1.1. Kursskriptsystem

Der hier vorgestellte Konzeptvorschlag geht von der Ergänzung bestehender Systeme zur elektronischen Unterstützung der Lehre um Annotationsfunktionalität aus. Somit existiert eine Realisierung des Kursskriptssystems schon eigenständig. Trotzdem seien hier einige Anmerkungen für den Fall einer Neuentwicklung erlaubt:

Wird ein neues Gesamtsystem zur Unterstützung der Lehre erstellt, sollte auch hier eine strikte Trennung zwischen der Verwaltung des Kursskripts und der Annotationen eingehalten werden. Das Kursskript ist allgemein gehalten, über mehrere Kurse nutzbar und schon von daher „haltbarer“ als Annotationen, die eventuell genau auf eine spezielle Lehrsituation hin entstanden sind und um Sinn zu machen den jeweiligen Kontext benötigen.

Ein Kursskriptsystem sollte sich in die Teile Kursskriptinhalt und Kursskriptdarstellung zergliedern.

Trotz der großen Vielfalt an Systemen zur Verwaltung von Lerninhalten und Kursskriptinhalten ist eindeutig ein Trend zu XML-gestützten Systemen erkennbar. Gute Beispiele dafür sind LOM - Learning Object Metadata, (vgl. IEEE, 2002) und TargeTeam - TArgeted Reuse and GEneration of TEAching Materials) (vgl. Teege, 2002).

Ein besonderes Augenmerk wird bei Lerninhalten heute auf Wiederverwendbarkeit gelegt. Stichpunkte sind hier RLO - Reusable Learning Object (vgl. z.B. Wharrad & Leeder, 2003) und SCORM - Sharable Content Object Reference Model (vgl. ADLNET, 2005).

Neben dem Teilsystem, welches das Kursskript inhaltlich repräsentiert, ist ein weiteres Teilsystem zur Darstellung des Kursskripts notwendig. Da zu einem Kursskript mehrere Darstellungen gleichberechtigt nebeneinander existieren können, hat sich diese Trennung bewährt (vgl. hierzu auch die eben genannten Systeme zur Verwaltung des Kursskriptinhalts). Als Beispiel solcher verschiedener Darstellungen kann eine Druckversion des Skripts mit Seitenzahlen im Gegensatz zu einer Online-Version mit intensiver Verlinkung angeführt werden.

Für den weiteren Verlauf wird angenommen, dass von der im folgenden beschriebenen Architektur genau ein Kurs um Annotationsfunktionalität ergänzt wird. Sind mehrere Kurse zu erweitern, kann dies durch mehrmaliges Instanziieren des Annotationssystems, wo nötig mit geänderten Port-Angaben, problemlos erfolgen.

Nach diesem kurzen Einschub zu Kursskriptsystemen nun zurück zu den Komponenten des Annotationssystems.

5.2.1.2. ControlCenter (CC)

Die Komponente *ControlCenter* bildet die zentrale Schnittstelle für Annotationen nach außen hin. Alle Anfragen eines Benutzers laufen also vor einer Bearbeitung hier auf. Von hier aus werden die anderen Komponenten bei Bedarf gestartet. Die *ControlCenter*-Komponente setzt also auch die Steuerungselemente des Annotationssystems um.

Wie in Kap. 5.1.4.2. geschildert gibt es mehrere Arten der Integration auf Ebene der Steuerungselemente. Da diese Architektur die *parallel getrennte* und die *untergeordnet integrierte* Variante gleichsam unterstützt, wird als Modell hinter den Steuerungselementen ein Menüsystem mit Iconleiste implementiert. Dieses Modell ist eigenständig lauffähig, also problemlos zur Umsetzung von *parallel getrennt* nutzbar. Andererseits bieten heute fast alle grafischen Benutzungsoberflächen dieses Bedienungsmodell. Eine untergeordnete Integration erfordert somit auf Seiten des Kursskriptsystems nur Erweiterungen des schon vorhandenen Bedienungsmodells und ist damit meist einfach umzusetzen.

Alle Komponenten des lokalen Annotationssystems, die dem Benutzer Menüpunkte zur Verfügung stellen, registrieren sich beim *ControlCenter*. Dabei werden die notwendigen Menüpunkte erstellt. Eine Registrierung ist dauerhaft und wird wie alle anderen Einstellungen des Benutzers in einer vom *ControlCenter* global zur Verfügung gestellten Registry persistent abgelegt. Die Registrierung der Komponenten wird sinnvollerweise automatisch bei der Installation durchgeführt (vgl. hierzu auch Plugin-Mechanismus bei Eclipse, Clayberg & Beck, 2004).

Das *ControlCenter* ist die einzige Komponente, die direkt mit dem Kursskriptsystem interagiert. Zum Einen nimmt das *ControlCenter* bei Bedarf Einfluß auf die Darstellung und die Platzierung des Kursskripts. Zum Anderen wertet es Informationen aus, welche es vom Kursskriptsystem erhält. Dazu zählt in erster Linie das Navigieren des Benutzers im Kursskript.

Abbildung 20 zeigt das Klassendiagramm für diese Komponente.

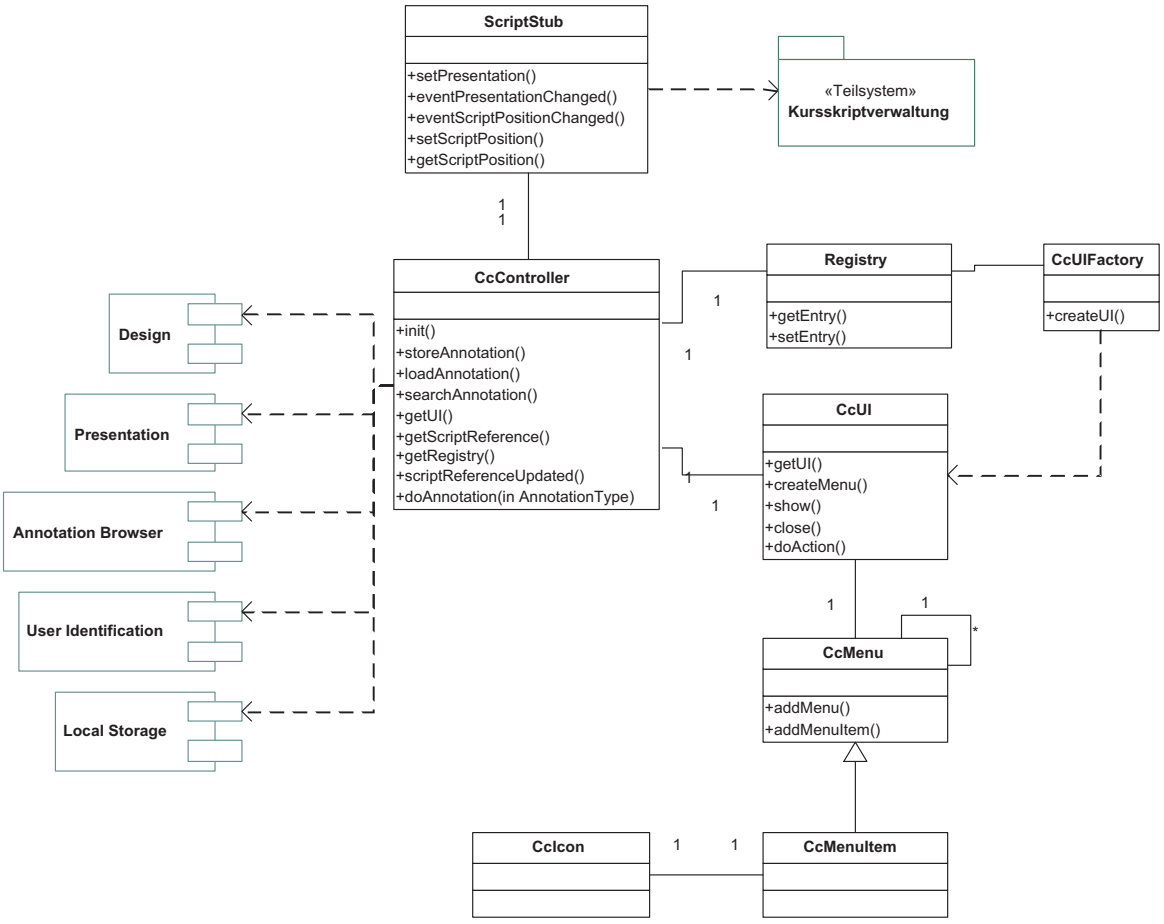


Abb. 20: Klassendiagramm für das ControlCenter

Zentraler Punkt der Komponente ist die Klasse *CcController*. Alle Anforderungen anderer Komponenten des Annotationssystems an Dienste der *ControlCenter*-Komponente laufen hier auf. Eine direkte Kommunikation zwischen anderen Objekten der Komponente und Objekten anderer Komponenten ist nicht vorgesehen und der klaren Festlegung von Schnittstellen wegen auch strikt zu vermeiden (vgl. Oesterreich, 1998).

Die Klasse *ScriptStub* stellt die Verbindung zur Kursskriptverwaltung her. Hier werden die Schnittstellenmethoden des Kursskripts geeignet aufgerufen beziehungsweise die Ereignismethodenaufrufe des Kursskriptsystems entgegengenommen. Die Anpassung an Kursskriptsysteme erfolgt über die Benutzung von Unterklassen von *ScriptStub*.

Die Referenzimplementierung des Annotationssystem erlaubt zwei verschiedene Arten Starts des Annotationssystems: Entweder das Annotationssystem wird über den ScriptStub, also aus dem Kursskriptsystem heraus, fremd gestartet oder aber der CcController wird als eigenständige Anwendung mit eigener Hauptfunktion gestartet. Je nach Grad der Integration in das Kursskriptsystem wird die eine oder andere benutzt.

Die *ControlCenter*-Komponente bietet über ein *Registry*-Objekt die Möglichkeit, Einstellungen dauerhaft abzulegen und abzurufen. Die Methoden `getEntry()` und `setEntry()` sind `static` definiert und somit direkt von jedem anderen Objekt des Annotationssystems erreichbar.

Die restlichen Klassen der *ControlCenter*-Komponente dienen der Umsetzung der Benutzungsoberfläche nach dem oben angesprochenen Bedienungsmodell eines Menüs mit Iconleiste.

5.2.1.3. Design (Ds)

Mit Hilfe dieser Komponente werden vom Benutzer Annotationen erstellt. Die hierfür geltenden Aspekte wurden schon vorab in Kap. 5.1.4.3. behandelt. Das konkrete Klassendiagramm dieser Komponente zeigt Abbildung 21.

Der Befehl zur Erstellung einer Annotation wird im *CcController*-Objekt in der *ControlCenter*-Komponente angestoßen. Das *CcController*-Objekt ruft daraufhin die Methode *doAnnotate()* der beim Initialisieren erstellten *DsController*-Instanz auf. Dabei übergibt es den gewünschten Annotationstyp (z.B. „Stichpunkt“) und die aktuelle Kursskriptdarstellung. Über die Kursskriptdarstellung werden im weiteren Verlauf Komponenten gewählt, die mit dem entsprechenden Kursskriptdarstellungssystem kooperieren können. Wird hier nichts angegeben, wird automatisch „generic“ gesetzt. Damit werden Komponenten gewählt, die eigenständig, ohne direkte Verbindung zum Kursskriptdarstellungssystem arbeiten können. Die Integration verläuft dann also *vertikal geschichtet*. Die Methode `doAnnotate()` der *DsController*-Instanz kreiert über die *DsDesignerFactory*-Methode `createDesigner()` ein passendes *DsDesigner*-Objekt. Der Factory werden dazu die Parameter Annotationstyp und Kursskriptdarstellung übergeben. Die Entscheidung darüber, von welcher konkreten *DsDesigner*-Klasse eine Instanz erstellt werden soll, wird folgendermaßen getroffen: Die Factory benutzt „<Annotationstyp>/<Kursskriptdarstellung>“ als Schlüssel in der Registry des Annotationssystems. Dort ist der Name der dafür auf diesem Benutzersystem zu instanziierenden Klasse hinterlegt. Dieser Eintrag wurde bei der Installation der jeweiligen *Designer*-Klassen auf dem Benutzersystem erstellt. Wird kein Eintrag für das angegebene Kursskriptdarstellungssystem gefunden, wird unter „<Annotationstyp>/generic“ nachgesehen und die entsprechende Klasse instanziiert.

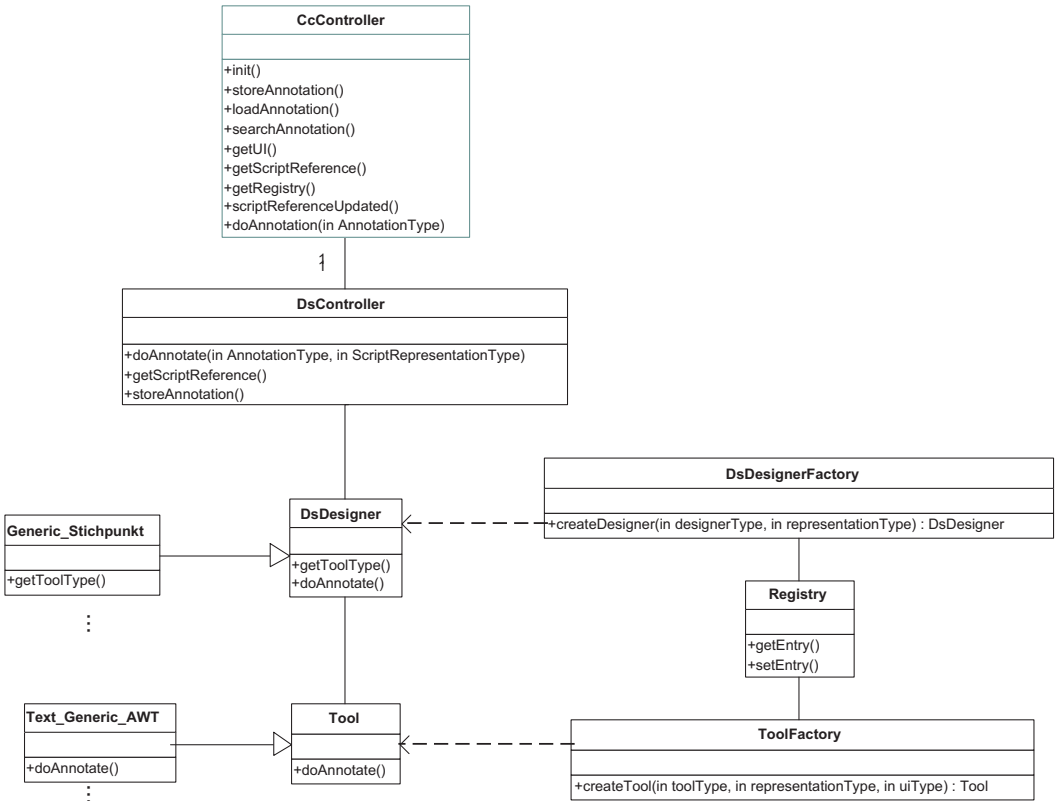


Abb. 21: Klassendiagramm der Design-Komponente

Mit dem *DsDesigner*-Objekt steht nun also ein Objekt zur Verfügung, welches den Annotationstyp und damit die semantische Ausrichtung der Annotation kapselt. In diesem Objekt wird auch gekapselt, wie der Inhalt der Annotation für die Persistenzschicht aufbereitet, sprich serialisiert werden soll.

Nachdem das *DsDesigner*-Objekt erstellt wurde, wird von der Methode `doAnnotate()` der *DsController*-Instanz die Methode `doAnnotate()` des *DsDesigner*-Objekts aufgerufen.

Da das *DsDesigner*-Objekt nur für die semantische Ausrichtung der Annotation steht, fehlt immer noch ein konkretes Werkzeug - eine Benutzerschnittstelle - für die Erstellung der Annotation. Das passende Werkzeug-Objekt erstellt das *DsDesigner*-Objekt über die Methode `createTool()` der *DsToolFactory* in der Methode `doAnnotate()`. Der Factory wird dabei neben dem Parameter Kursskriptdarstellung die Parameter Werkzeugtyp und Benutzungsoberflächentyp übergeben. Auch hier werden die Parameter wieder zu einem Schlüssel in die Registry zusammengesetzt („<Benutzungsoberflächentyp>/<Kursskriptdarstellung>/<Werkzeugtyp>“). Der dort hinterlegte Klassenname gibt die Klasse an, die instanziiert werden muss.

Die DsDesigner-Klasse stellt in der vorgestellten Architektur entsprechend einer MVC (Modell - View - Controller) Architektur, den Controller für die Annotationserstellung. Sie bestimmt also auch, aus welchen Objekten die View, also die Ansicht bestehen soll und instanziiert deshalb gegebenenfalls mehrere verschiedene *DsTool*-Objekte über die Factory und koordiniert ihre Zusammenarbeit. Als Beispiel kann folgende Kombination genannt werden:

Ein *DsTool*-Objekt für das Freihandzeichnen, ein *DsTool*-Objekt zur Farbwahl und ein *DsTool*-Objekt zur Darstellung eines Hilfsgitters.

Neben dem Beginn einer Annotation ist auch das Beenden einer Annotation ein interessanter Aspekt. Das Vorgehen der hier vorgeschlagenen konkreten Architektur zeigen folgende Codefragmente:

```
public class DsController {
   ...
   public void doAnnotate(String annotationType,
                          String scriptRepresentationType) {
      if (_designer != null) {
         _designer.finishAnnotation();
      }

      _designer=DsDesignerFactory
                  .createDesigner( this,
                                   annotationType,
                                   scriptRepresentationType);

      _designer.init( annotationType, scriptRepresentationType );

      _designer.doAnnotate();
   }
   ...
}

public abstract class DsDesigner {
   ...
   public void doAnnotate() {
      _reference = _dsController.getScriptReference();

      _dsTool.doAnnotate();

      if (_dsTool.isModal()) {
         finishAnnotation();
      } // otherwise, DsController will call finishAnnotation()
   }

   public void finishAnnotation() {
      if (!_annotationFinished) {
         _reference = _dsTool.prepareReference(_reference);
         _reference = prepareReference(_reference);

         byte[] blob = _dsTool.getAnnotationBlob();

         Annotation annotation = new Annotation(_reference,
                                                blob,
                                                getToolType(),
                                                _annotationType);

         _dsController.storeAnnotation(annotation);

         _annotationFinished = true;
      }
   }
   ...
}
```

Wie aus dem ersten Codestück ersichtlich, beendet das DsController-Objekt eine Annotation spätestens, sobald eine neue Annotation über doAnnotate() begonnen werden soll. Dies erfolgt bei einem Wechsel der Annotationsform.

Das DsDesigner-Objekt hingegen prüft in der Standardimplementierung, ob das Erstellungswerkzeug _dsTool modal arbeitet. Modal bedeutet, dass das Erstellungswerkzeug alle Eingaben des Benutzers in das Annotationssystem abfängt und somit kein Weiterarbeiten an anderer Stelle erlaubt. Die Methode doAnnotate() des _dsTool-Objekts kehrt dann erst nach Beendigung der Annotation zurück.

Über die Festlegung, ob ein DsTool-Objekt modal arbeitet, kann also gesteuert werden, ob zum Beispiel aus mehreren einzelnen Strichen bei einer Freihandannotation eine einzige Gesamtannotation (⇒ nicht modal) oder pro Strich eine eigene Annotation entstehen und damit verwaltet wird (⇒ modal).

5.2.1.4. Presentation (Ps)

Die Annotationserstellung und die Annotationsdarstellung wird in diesem Architekturvorschlag aus mehreren Gründen in zwei Komponenten *Design* und *Presentation* aufgeteilt (vgl. Kap. 5.1.4.3.).

Ein weiterer Grund, die beiden Teile Erstellung und Darstellung zu trennen liegt in der Ergonomie: Bei der Erstellung von Annotationen wird jeweils auf die zu erstellende Annotation fokussiert. Dies erlaubt während der Erstellung andere Teile auszublenden und Hilfsmittel und Werkzeuge einzublenden. Bei allen anderen Arbeiten stehen das Kursskript und eventuell mehrere Annotationen gleichzeitig gleichberechtigt im Vordergrund. Dies erfordert eine überlegte Umsetzung der Darstellung. So sollte die Darstellungskomponente um eine Informationsüberflutung zu verhindern bei hoher Annotationsdichte Konzepte zur Reduktion der visuellen Komplexität einsetzen. Ihrer Intuitivität wegen im Lehrumfeld hervorragend geeignete Konzepte sind unter anderem *Semantic Fisheye Views* (vgl. Furnas, 1981) und *Perspective Wall* (vgl. Mackinlay, Robertson, & Card, 1991). Zur Umsetzung dieser sinnvollen Konzepte ist es erforderlich, auch verkleinerte Darstellungen und symbolische Ersatzdarstellungen einer Annotation über die bloße Reproduktion hinaus anzubieten.

Das Klassendiagramm für die Komponente *Presentation* zeigt Abbildung 22.

Wie aus dem Klassendiagramm ersichtlich, bedient sich auch die *Annotation Browser*-Komponente zur Darstellung der Annotationen der Klassen der *Presentation*-Komponente. Dies entspricht der konsequenten Umsetzung des Wiederverwendungsgedanken.

Prinzipiell arbeitet die *Presentation*-Komponente ähnlich der *Design*-Komponente. Bekommt die *Presentation*-Komponente durch den Aufruf von `presentAnnotation(Annotation)` der *PsController*-Klasse, den Befehl eine Annotation darzustellen, wird abhängig vom der Annotation entsprechenden Annotationstyp, der Kursskriptdarstellung und der Benutzungsoberfläche über die Methode `createFactory()` der *PsPresenterFactory*-Klasse ein entsprechendes *Presenter*-Objekt instanziiert. Dabei bedient sich die *PsPresenterFactory* wieder dem den Parametern entsprechenden Klassenamen in der Registry. Hier ist kein zweistufiges Konzept notwendig, da für die reine Darstellung keine Kombination von mehreren Werkzeugen (z.B. Werkzeug, Farbwahl und Hilfsgitter) notwendig ist.

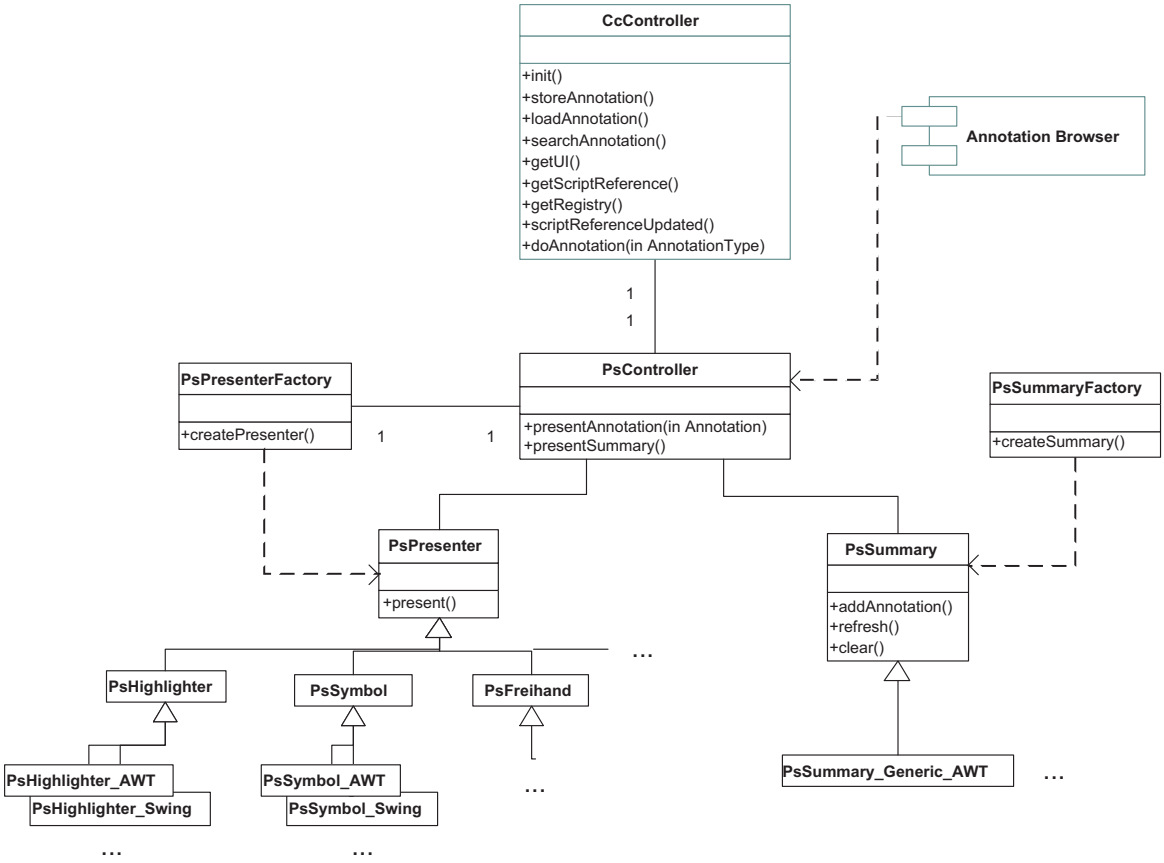


Abb. 22: Klassendiagramm der Presentation-Komponente

Die Methode `presentAnnotation(Annotation)` der *PsController*-Klasse wird von der *ControlCenter*-Komponente aufgerufen, wenn über das Kursskriptsystem eine Veränderung des dargestellten Kursskriptteils signalisiert wird.

Die *Presentation*-Komponente erlaubt neben der Darstellung einer einzelnen Annotation, mehrere Annotationen unabhängig vom Kursskript zusammengefasst in ihrer verkleinerten Darstellung anzuzeigen. Dafür wurde in der konkreten Implementierung die Klasse *PsSummary* eingeführt. *PsSummary* kann eine Menge von Annotationen zur Darstellung übergeben werden. Konkret eingesetzt wird dies in der Architektur, um alle Annotationen zum im Kursskript angezeigten Teil aufzulisten. Um hierbei wieder auf eine unterschiedlich enge Integration der Darstellungsflächen des Kursskripts und des Annotationssystems reagieren zu können, wird die konkrete Instanz von *PsSummary* zur Laufzeit von der Factoryklasse *PsSummaryFactory* erstellt.

5.2.1.5. Annotation Browser (Ab)

Dementsprechend eingesetzt sind Annotationen eigenständig im Lehrumfeld auch ohne Kursskript gewinnbringend nutzbar. Hier kann etwa noch einmal auf den Einsatz im Sinne einer Ordnungsstrategie verwiesen werden (vgl. 2.3.3., S. 27). Wurden zum Beispiel wichtige Punkte des Kursskripts mit dem Annotationstyp *Stichpunkt* annotiert, so erlaubt es ein Mechanismus zum Suchen und Filtern von Annotationen auf Knopfdruck einen Überblick über alle Stichpunkte zu generieren, was in der Regel einer ersten Zusammenfassung des Stoffes entspricht. Aus diesem Grund umfaßt der vorgestellte Architekturvorschlag eine eigene

Komponente zum „Browsen“, also Durchstöbern der Annotationen. Diese Komponente erlaubt das Suchen und Filtern von Annotationen nach gewissen Kriterien und den Aufruf der jeweiligen Darstellung unabhängig vom Kursskript. Somit wird eine Verdichtung des Kursinhalts auf die Annotationen erreicht. Als Suchkriterien vorstellbar sind neben Verfasser, Zeitpunkt und Annotationstyp durchaus weitere. Ist es möglich, auf den Annotationsinhalt zuzugreifen, wie zum Beispiel bei Textannotationen, kann auch dieser mit Suchmustern durchsucht werden.

Das Klassendiagramm der Annotation Browser Komponente ist in Abbildung 23 dargestellt.

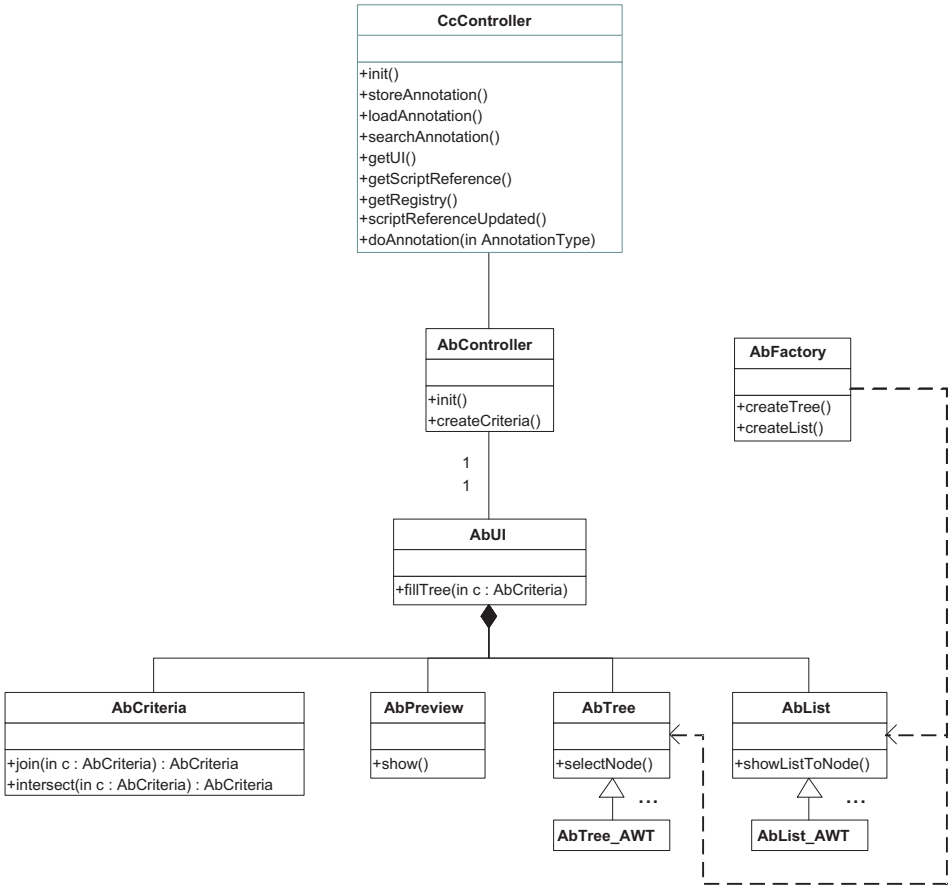


Abb. 23: Klassendiagramm der Annotation Browser-Komponente

Aus technischer Sicht kann die *Annotation Browser*-Komponente mit ihren Möglichkeiten zur Suche und Anzeige von Annotationen auch als Benutzungsoberfläche für den Annotationsspeicher angesehen werden. Je nachdem, ob dafür der lokale Annotationsspeicher oder der zentrale zugrunde gelegt wird, lassen sich unterschiedliche Dinge erreichen: Im Falle des lokalen Annotationsspeichers kann der Annotationsbrowser als Ausgangspunkt für das Wiederholen des Stoffes benutzt werden. Dabei ist es dann möglich, nur mit den Annotationen alleine zu wiederholen, oder aber von den Annotationen ausgehend, an Stellen im Kursskript zu springen. Im Falle des zentralen Annotationsspeichers können über kumulierende Funktionen auch Statistiken erstellt und Aussagen zum allgemeinen Annotierverhalten getroffen werden. Dies ergibt eine Quelle für Feedback für den Dozenten.

Auch die *Annotation Browser*-Komponente besitzt ein eigenes Controller-Objekt, den *AbController*. Das *AbController*-Objekt stellt die Verbindung zum restlichen Annotationssys-

tem her und steuert die Benutzungsoberfläche der *Annotation Browser*-Komponente. Als Modell für die Benutzungsoberfläche der *Annotation Browser*-Komponente wird eine Baumansicht mit zugeordneter Listendarstellung gewählt. Die Baumansicht wird über die Klasse *AbTree*, die Listendarstellung über die Klasse *AbList* umgesetzt. Dabei wird die Interaktion beider Klassen vollständig in diesen Basisklassen verwirklicht. Erst in den abgeleiteten Klassen wird die Darstellung in der jeweiligen zur Verfügung stehenden Benutzungsoberfläche thematisiert (z.B. *AbTree_AWT* und *AbList_AWT*). Welche konkreten Klassen instanziiert werden, entscheidet die *AbFactory* anhand der eingestellten Benutzungsoberfläche und Einträgen in der Registry.

Die in der *Annotation Browser*-Komponente möglichen Filter und Suchanfragen werden jeweils über *AbCriteria*-Objekte repräsentiert. Neben den Parametern der Filter und Suchanfragen bietet *AbCriteria* auch einen begrenzten Zwischenspeicher für Annotationen im Lösungsraum. Mehrere *AbCriteria*-Objekte können über `join()` (Vereinigung) oder `intersect()` (Schnitt) zu neuen *AbCriteria*-Objekten zusammengeschaltet werden.

Wird die *Annotation Browser*-Komponente ohne Kursskriptsystem benutzt, müssen Annotationen trotzdem dargestellt werden können. Dies ist die Zuständigkeit der Klasse *AbPreview*.

5.2.1.6. Local Storage (Ls)

Laut Anforderung muss das erarbeitete Annotationssystem das Arbeiten ohne Netzwerkverbindung erlauben. Deshalb ist ein lokaler Speicher für Annotationen und einige Verwaltungstabellen bei jedem Benutzer unumgänglich. Diese werden in der Komponente *Local Storage* verwirklicht.

Die Art der technischen Umsetzung des Annotationsspeichers ist relativ frei zu gestalten, da verschiedene Benutzer des Systems in ihrer eigenen Umgebung unterschiedliche Möglichkeiten haben. Vorgaben das Annotationssystems, die andere Produkte zur Voraussetzung machen, sollten der Akzeptanz wegen in jedem Fall vermieden werden. Eine solche Vorgabe wäre etwa die Forderung nach einem SQL-Datenbanksystems.

Die Umsetzung aller Annotationsspeicherkomponenten basiert deshalb auf einer Persistenzschicht, welche den eigentlichen, technischen Speichermechanismus kapselt. Damit kann unter anderem eine dateibasierte Verwaltung, ein Anbindung an ein Datenbanksystem oder eine Anbindung an ein Semantic Web jedem Benutzer austauschbar angeboten werden.

Zu erwähnen ist, dass die Komponente zur Verwaltung des Kursskripts schon selbst auf einen Speicher zur Bereitstellung der Kursdaten zurückgreift. Es sollte in jedem Fall geprüft werden, ob sich dieser Speicher auch für Annotationen eignet. Ist der Kursinhalt speziell in einem Semantic Web abgelegt, so bietet sich eine Ablage der Annotationen im selben Web als natürlichste Lösung an.

Die Art und der Aufbau der abzulegenden Daten für Annotationen wird in 5.2.2. ab S. 111 genauer beschrieben. Abbildung 24 zeigt das Klassendiagramm zur *Local Storage*-Komponente.

5.2.1.7. User Identification (Uid)

Werden Annotationen auch anderen Benutzern zur Verfügung gestellt, sollte unbedingt der Autor der Annotation mit abgelegt werden. Aus diesem Grund ist es notwendig, dass jeder Verfasser von Annotationen sich vor dem Weitergeben einer Annotation identifiziert. Ist au-

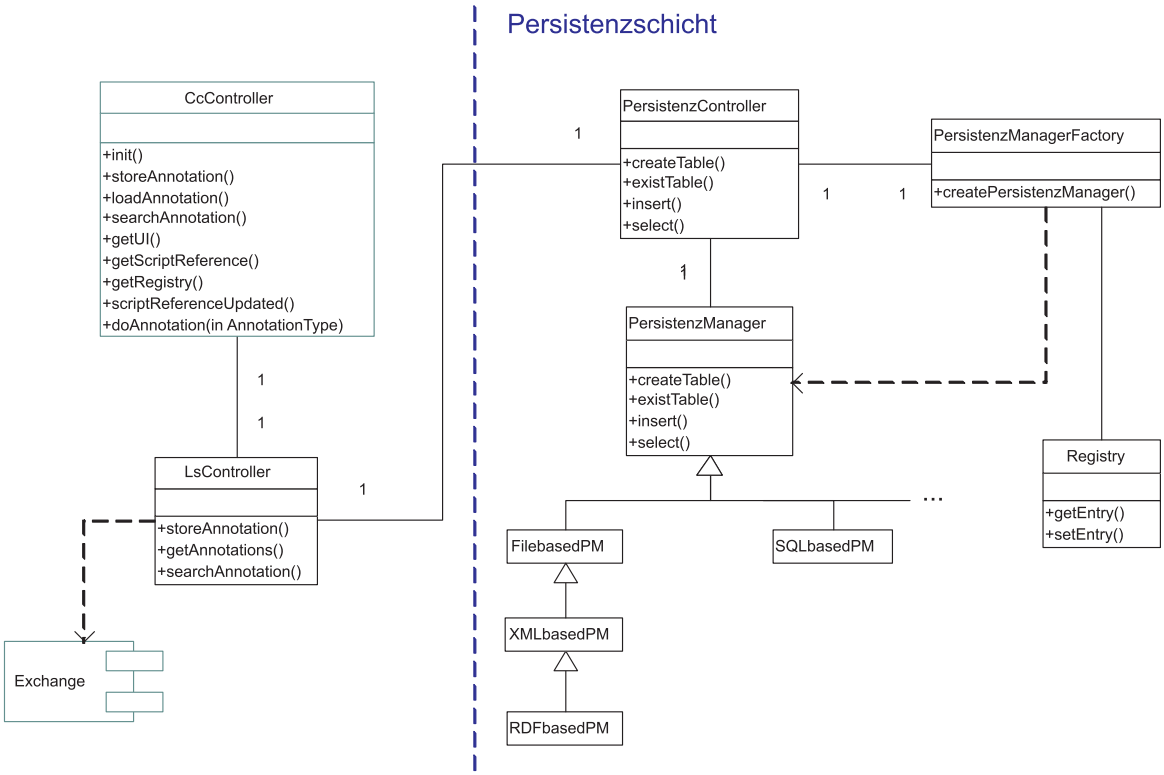


Abb. 24: Klassendiagramm zur Local Storage-Komponente

ßerdem gefordert, den Personenkreis der eine Annotation betrachten darf einzuschränken, so müssen sich Benutzer die Annotationen anderer betrachten wollen ebenfalls identifizieren. Für das in Kap. 5.1.4.6. vorgestellte Konzept zur Umsetzung der Gruppen- und kollaborativen Annotationen, ist eine Benutzeridentifikation in jedem Fall notwendig.

Die Benutzeridentifikation kann auch automatisiert über das lokale System durch zum Beispiel den Login, ein Cookie oder Benutzerzertifikate im HTML-Umfeld erfolgen. Damit ist technisch eine Identität festgestellt.

Praktisch kann es bei Nutzung des lokalen Logins im Gesamtsystem aller Benutzerrechner noch zu doppelten Logins kommen. Dies kann teilweise durch Ergänzung des Rechnernamens umgangen werden. Dann ist eine Anmeldung für einen Benutzer aber nur immer von einem Rechner aus möglich. Eine Lösung ist ein Identifikationsverfahren, welches über den zentralen Annotationsserver arbeitet. Sollte dieser nicht erreichbar sein und mit einem netzlokalen Server gearbeitet werden, sind die Benutzerangaben als vorläufig zu markieren, später beim Anmelden am zentralen Annotationsserver zu überprüfen und gegebenenfalls zu ersetzen. Eine weitere Lösung besteht in der Nutzung zentral ausgestellter Benutzerzertifikate. Das lokale System kann aus dem genutzten Benutzerzertifikat die Identität ermitteln. Über die zentrale Vergabe der Zertifikate sind Duplikate in den Benutzeridentifikatoren ausgeschlossen. Beide Varianten sind über die Komponente *User Identification* abgedeckt.

Das Klassendiagramm zu dieser Komponente ist in Abbildung 25 dargestellt. Auch in dieser Komponente bildet die Schnittstelle zum restlichen Annotationssystem mit dem *UsrController*-Objekt ein Controller. Dieser erstellt für die konkrete Umsetzung der Identifikationskom-

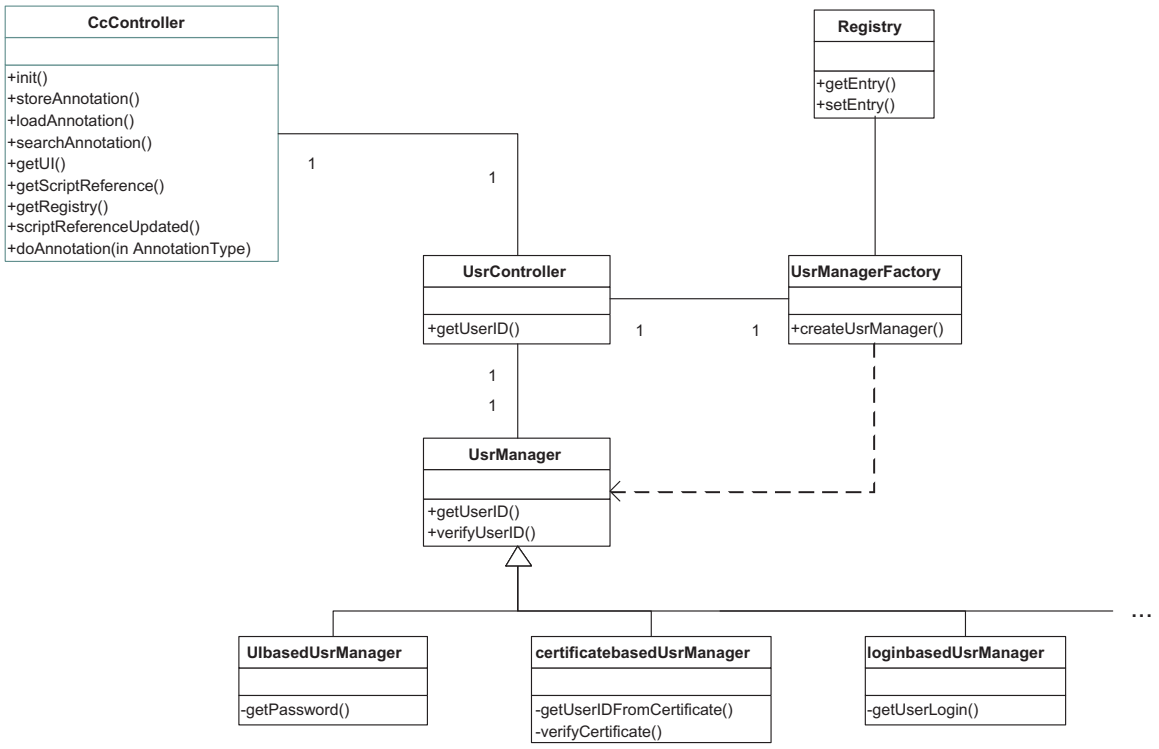


Abb. 25: Klassendiagramm zur User Identification-Komponente

ponente über das *UsrManagerFactory*-Objekt ein spezialisiertes *UsrManager*-Objekt. Welches Objekt erstellt werden soll, wird in der *Registry* hinterlegt.

5.2.1.8. Central Server (Cs)

Die *Central Server*-Komponente hat die Aufgabe, über die Zeit alle Annotationen die während eines Kurses von allen Benutzern gemacht werden zu sammeln, zu speichern und gegebenenfalls einzelne Annotationen automatisch an Benutzer weiterzugeben. Für Statistiken werden verschiedene Aggregatsfunktionen zur Verfügung gestellt.

Dazu ist neben der Umsetzung des schon oben beschriebenen Mechanismus zum Abgleich von Annotationen weiteres eher administratives notwendig:

Damit ein Benutzer auswählen kann, wessen Annotationen er übernehmen möchte und wem er eigene private Annotationen zur Verfügung stellen will, muss er wissen, wer überhaupt mit dem System arbeitet. Deshalb enthält die *Central Server*-Komponente ein Verzeichnis aller Benutzer, von denen schon Annotationen abgelegt wurden, beziehungsweise die sich anderweitig an der *Central Server*-Komponente mindestens einmal angemeldet haben. Ein Benutzer kann nun jederzeit im Betrieb am Netz über die *Annotation Browser*-Komponente diese Liste anfordern und Benutzer, auf deren Annotationen er gerne Zugriff hätte, in die in Kap. 5.1.4.6. beschriebene *Gesucht-Liste* übernehmen. Die *Gesucht-Liste* wird über die *Local Storage*-Komponente lokal beim Benutzer abgelegt. Das eben gesagt gilt in gleichem Maße für die *Erlaubt-Liste*.

Beim Anmelden eines Benutzers bei der *Central Server*-Komponente werden nun nicht nur die ohne Netzwerkverbindung gemachten Annotationen übermittelt, sondern auch diese beiden Listen. Die Central Server-Komponente führt eine eigene globale Liste - die Notifika-

tionsliste-, in der je Benutzer angeführt ist, an wen aufgrund aller gerade bekannten *Erlaubt*- und *Gesucht-Listen* eine Benachrichtigung über eine neue Annotation versendet werden muss. Meldet sich ein Benutzer ab, wird er komplett aus der Notifikationsliste entfernt. Damit ist diese Liste in jedem Moment so kurz als möglich. Es wird also keine Zeit für das Überspringen nicht mit dem zentralen Server verbundener Einträge oder durch Senden von Benachrichtigungen ohne Empfänger vergeudet.

Die globale Liste hat folgenden Aufbau: Je angemeldetem Benutzer enthält sie einen Eintrag, der jeweils aus zwei Teilen besteht. Der erste Teil ist der Benutzeridentifikator, der zweite der Kopf einer einfach verketteten Liste. In dieser Liste sind die Identifikatoren aller Benutzer aufgeführt, die beim Eintreffen einer neuen Annotation des Benutzers benachrichtigt werden müssen (vgl. Abb. 26).

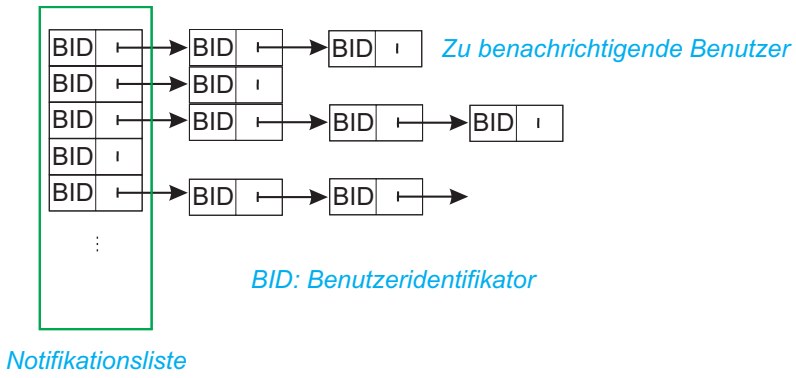


Abb. 26: Schematischer Aufbau der Notifikationsliste

Am natürlichsten lässt sich die Notifikationsliste als Hashmap umsetzen. Der Benutzeridentifikator dient dabei als Schlüssel.

Meldet sich nun ein Benutzersystem an der *Central Server*-Komponente an, wird folgender Algorithmus zur Integration der Information der Erlaubt und Gesucht-Liste ausgeführt:

```
// leeren Benutzereintrag in Notifikationsliste erstellen
neuID = getUserID();
notificationList.insertUserID(neuID);

// Alle Einträge der Erlaubtliste des Benutzers durchgehen ...
neuErlaubtListe = getErlaubtListeTo(neuID);
while( neuErlaubtListe.hasNext() ) {
   // ... und prüfen, ob der eingetragene Benutzer, falls er gerade
   // angemeldet ist, auch an Annotationen interessiert ist ...
   altID = neuErlaubtListe.next();
   if (isConnected( altID ) ) {
      altGesuchtListe = getGesuchtListe(altID);
      // ... falls Interesse besteht, in die Notifikationsliste aufnehmen
      if ( altGesuchtListe.hasElement(neuID) ) {
         notificationList.insertReceiverAtUser(altID, neuID);
      }
   }
}

// Nun noch neuen Benutzer, wo nötig, in die Notifikationsliste anderer
// Benutzer eintragen. Dazu seine Gesucht-Liste durchgehen...
```

```
neuGesuchtListe = getGesuchtListeTo(neuID);
while( neuGesuchtListe.hasNext() ) {
   // ... und prüfen, ob der eingetragene Benutzer, falls er gerade
   // angemeldet ist, auch seine Annotationen weitergeben will …
   altID = neuGesuchtListe.next();
   if (isConnected( altID ) ) {
      altErlaubtListe = getErlaubtListe(altID);
      // ... falls Erlaubt, in die Notifikationsliste aufnehmen
      if ( altErlaubtListe.hasElement(neuID) ) {
         notificationList.insertReceiverAtUser(neuID, altID);
      }
   }
}
```

Damit ist die Notifikationsliste wieder aktuell. Der Code für das Durchgehen der Listen in den unteren beiden Abschnitten wurde der Übersichtlichkeit wegen nicht zusammengefasst.

Meldet sich ein Benutzersystem ab, wird die Notifikationsliste über folgende Routine aktualisiert:

```
// BenutzerID des sich abmeldenden Studenten ermitteln
wegID = getUserID();

// Den direkten Eintrag des Benutzers aus der Notifikationsliste
// entfernen.
notificationList.removeEntry(wegID);

// Diesen Benutzer aus den Listen der anderen Benutzer in der
// Notifikationsliste entfernen. Dazu die Gesucht-Liste des
// sich abmeldenden Benutzers durchgehen.
wegGesuchtListe = getGesuchtListeTo(wegID);
while( wegGesuchtListe.hasNext() ) {
   id = wegGesuchtListe.next();
   notificationList.removeReceiverAtUser(wegID, id);
}
```

Meldet ein System eine neue Annotation an die *Central Server*-Komponente, werden die entsprechenden Benutzersysteme anhand der Notifikationsliste durch folgenden Algorithmus ermittelt:

```
// aus der Annotation den Autor ermitteln
autorID = annotation.getID();

// die Liste der zu informierenden Benutzer aus der Notifikationsliste
// ermitteln
toInformList = notificationList.getToInformList(autorID);
while (toInformList.hasNext()) {
   // Jeden Benutzer in der Liste informieren.
   toInformID = toInformList.next();
   notify(toInformID, annotation);
}
```

Das Klassendiagramm zur *Central Server*-Komponente zeigt Abbildung 27. Die *Central Server*-Komponente benutzt die Persistenzschichtlogik, die auch von der *Local Storage*-Komponente benutzt wird (natürlich in einer eigenen Instanz auf dem eigenen Rechner).

Neben dieser Wiederverwendung im Bereich der Persistenz wird auch ein anderer erheblicher Teil wiederverwendet. Die *CsController*-Klasse erbt den Großteil seiner Funktionalität

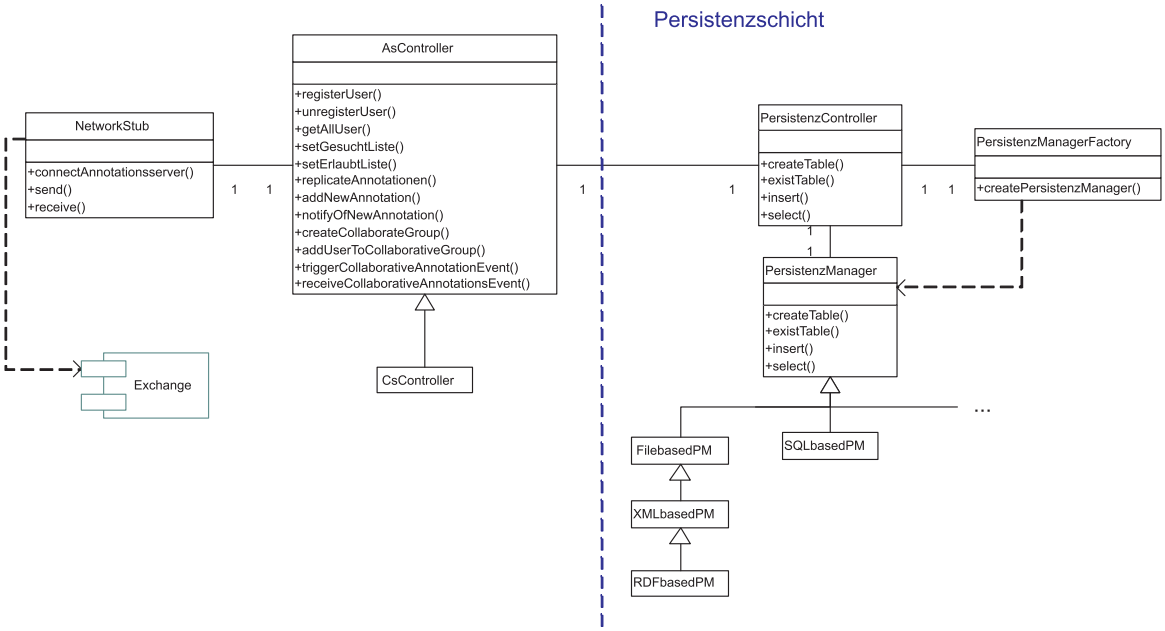


Abb. 27: Klassendiagramm der Central Server-Komponente

von der *AsController*-Klasse. Sie beinhaltet die Logik zur Verwaltung der Annotationen und der verschiedenen Benutzerlisten. Die *AsController*-Klasse bildet auch die Oberklasse der *NsController*-Klasse in der *Netlocal Server*-Komponente (vgl. folgenden Abschnitt). Der große Unterschied zwischen beiden Server-Komponenten liegt zum Einen in der benutzten *PersistenzManager*-Klasse. Die *Netlocal Server* Komponente benutzt der geringeren zu erwartenden Benutzerzahl und der nicht notwendigen dauerhaften Speicherung wegen eine hauptspeicherbasierte *PersistenzManager*-Klasse. Der andere Unterschied liegt darin, dass an die *NetLocal Server*-Komponente übergebene Annotationen weiterhin als abzugleichen geführt werden und die diesbezügliche Markierung im lokalen Benutzersystem nicht entfernt wird.

Die Klasse *NetworkStub* kapselt das Protokoll, den Verbindungsaufbau und die Verbindungsart bei Netzwerkverbindungen zwischen den einzelnen Komponenten. Auf jeder Seite der Verbindung sitzt eine Instanz der Klasse *NetworkStub*. Somit ist es durch Austausch der konkret zu verwendenden *NetworkStub*-Klassen möglich, zum Beispiel ohne weitere Veränderungen im restlichen System von einer normalen TCP-IP Verbindung auf eine SSL-Verbindung zu wechseln.

5.2.1.9. Netlocal Server (Ns)

Die *Netlocal Server*-Komponente hat die Aufgabe, temporär die Verteilung von Annotationen zu übernehmen. Konzeptuell gesehen unterscheidet sich die Komponente von der *Central Server*-Komponente nur darin, dass die *Central Server*-Komponente eine persistente Speicherung ermöglicht, die *Netlocal Server*-Komponente nicht. Dies wird dadurch erreicht, dass die Persistenzschicht ein *PersistenzManager*-Objekt für den Hauptspeicher benutzt.

Da keine persistente Speicherung notwendig ist, kann diese Komponente bei Bedarf auch auf jedem Benutzersystem gestartet werden oder von einem Benutzersystem zu einem anderen

migrieren. Die Umstände hierfür wurden in Kap. 5.1.4.4. beschrieben. Die Routine für das migrieren läuft folgendermaßen ab:

```
// ... vom CcController kam der Aufruf zum Beenden
// Prüfen ob hier ein Netlocal Server läuft
if (startedNetlocalServer()) {
   // muss beendet werden. Falls noch andere Benutzer angemeldet sind
   // ist ein Migrieren notwendig
   nls = getNetlocalServer();
   nls.migrate();
   nls.stop();
}
// restlichen lokalen Annotationsdienst beenden

...
class NsController extends AsController {
   ...

   public void migrate() {
      // ist ein Benutzer verbunden oder ist unser System sowieso
      // das einzig aktive
      if (!userSystemsConnected()) {
         // Wenn wir das einzige System sind, braucht nicht migriert
         // zu werden.
         return;
      }

      // Einem anderen verbundenen Benutzersystem die Aufforderung
      // schicken, einen neuen NsController zu starten
      someSystem = getConnectedUserSystem();
      sendMessage( ”startNsController“, someSystem );

      // Mit Timeout auf die Rückmeldung des neuen NsControllers warten
      try {
         newNsController = someSystem.getNsController();
      } catch ( TimeOutException ex ) {
         // Benutzer benachrichtigen, dass migrieren fehlgeschlagen ist.
         // Evtl. wegen erneutem Versuch fragen
         // Ansonsten das Migrieren abbrechen.
         return;
      }

      // (1)
      // Nach erfolgreichem Start eines weiteren NsControllers
      // keine Nachrichten von Benutzersystemen mehr annehmen.
      stopAcceptingMessages();

      // Zuerst die verschiedenen Listen zu den verbundenen
      // Benutzersystemen übertragen
      sendNotificationList( newNsController );
      sendErlaubtListen( newNsController );
      sendGesuchtListen( newNsController );

      // Jetzt kann der neue Netzlokale Server seine Arbeit beginnen.
      sendStartAcceptingMessages( newNsController );

      // Nun die verbundenen Benutzersysteme vom Wechsel in
      // Kenntnis setzen
```

```
        allSystems = getAllConnectedUserSystems();
        while( allSystems.hasNext() ) {
            system = allSystems.next();
            sendChangeNsAddress( system, newNsController );
        }

        // von (1) bis hierher sollte der Timeout einer normalen
        // Benutzeranfrage ausreichen. Damit geschieht der Wechsel im
        // Hintergrund. Ein Langer Timeout ist bei den Benutzersystemen
        // beim Benachrichtigen über neue Annotationen kein Problem

        // Jetzt noch die gesammelten Annotationen übertragen
        sendCollectedAnnotations( newNsController );
    }

    ...
}
```

Das Klassenmodell zur *Netlocal Server*-Komponente zeigt die Abbildung 28.

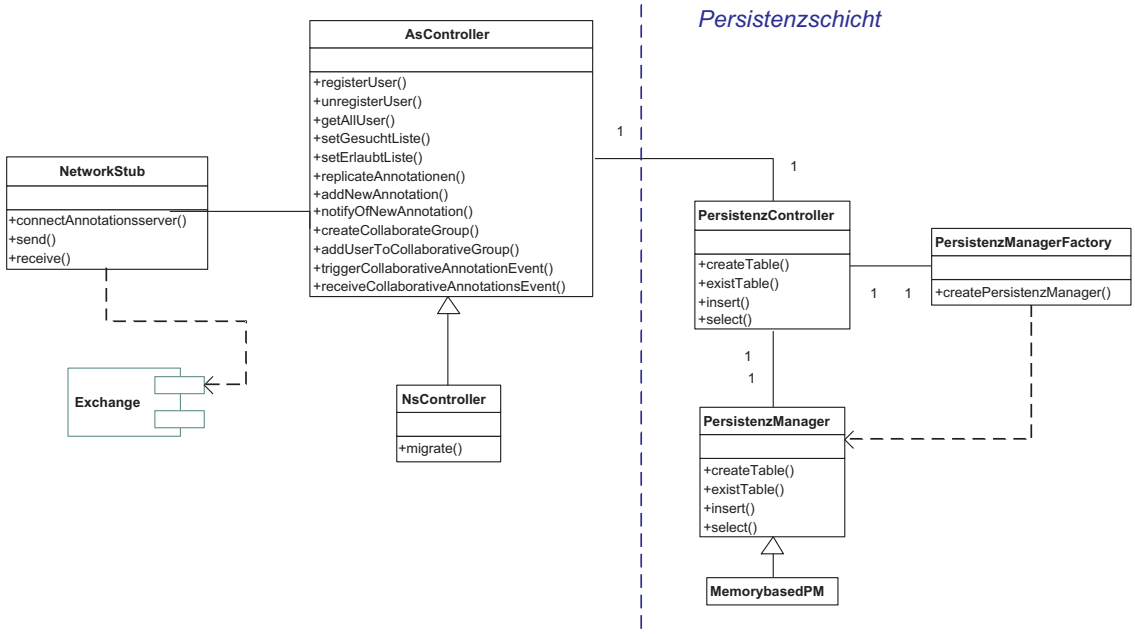


Abb. 28: Klassendiagramm zur Netlocal Server-Komponente

5.2.1.10. Exchange(Ex)

Diese Komponente kümmert sich auf Seite des lokalen Benutzersystems um die Kommunikation mit den Serverkomponenten.

Dazu gehört zum Einen die Implementierung des in Kap. 5.1.4.4. und Kap. 5.1.4.6. geschilderten Replikationsmechanismus für die eigenen Annotationen. Zum Anderen geben die Serverkomponenten die Information über neue Annotationen anderer Benutzer über einen Push-Mechanismus an die *Exchange*-Komponente weiter. Wünscht der Benutzer eine Übernahme einer Annotation eines anderen Benutzers, wird diese von der *Exchange* -Komponente gezielt über einen Pull-Mechanismus vom Server geholt.

Eine weitere Aufgabe der *Exchange*-Komponente ist die Unterstützung der kollaborativen Annotationen. Die dafür notwendigen Nachrichten werden zwischen den *Exchange*-Komponenten der Benutzer im Team über die *Central Server*-Komponente als Mittler hinweg ausgetauscht.

Abbildung 29 zeigt das Klassendiagramm der *Exchange*-Komponente.

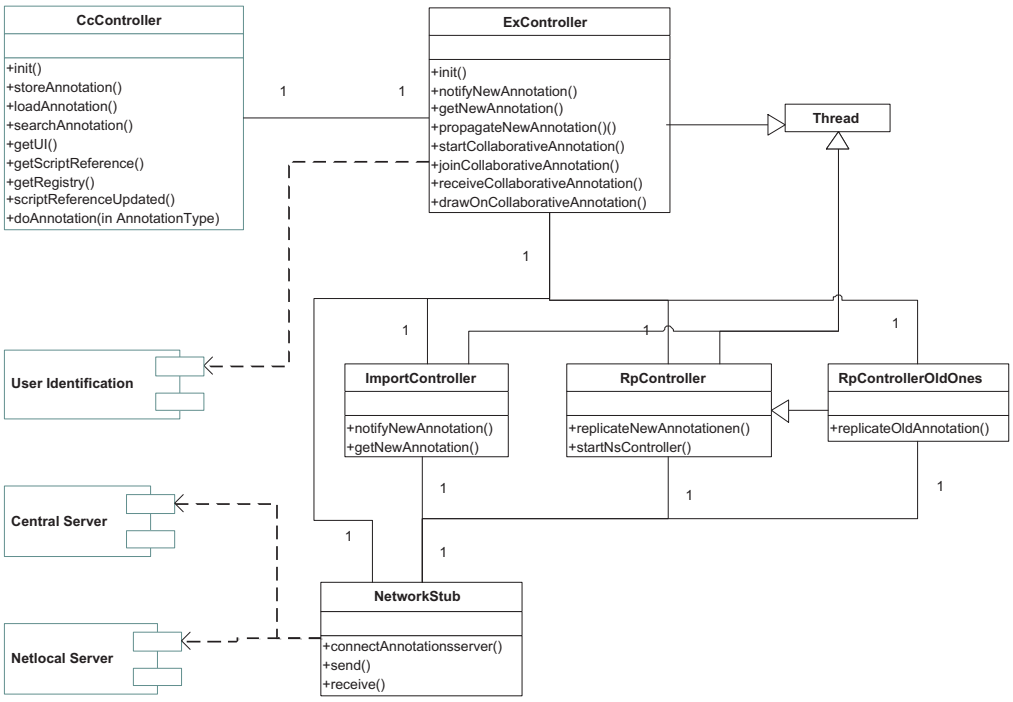


Abb. 29: Klassendiagramm der Exchange-Komponente

Ein *ExController*-Objekt stellt die Schnittstelle zum restlichen lokalen Annotationssystem dar. Innerhalb der *Exchange*-Komponente delegiert das *ExController*-Objekt die, da weitgehend nicht direkt den eigenen Arbeitsablauf des lokalen Annotationssystem betreffend, hintergrundfähigen Aufgaben an drei asynchron, parallel laufende Objekte: das *ImportController*-Objekt, das *RpController*-Objekt und das *RpControllerOldOnes*-Objekt. Alle drei Objekte stehen über ein jeweils eigenes *NetworkStub*-Objekt mit der Serverkomponente in Verbindung. Nur den Abgleich für kollaborative Annotationen übernimmt das *ExController*-Objekt, da dieser direkt auf den eigenen Arbeitsablauf einwirkt. Für kollaborative Annotationen nutzt auch das *ExController*-Objekt eine eigene Instanz eines *NetworkStub*-Objekts.

Das *ImportController*-Objekt gibt beim Aufbauen der Verbindung zur Serverkomponente die Erlaubt- und Gesucht-Liste weiter. Somit meldet es das Benutzersystem an der Server-Komponente an. Nun wartet es auf Benachrichtigungen über neue Annotationen, holt einzelne Annotationen anderer nach Aufforderung durch den Benutzer auf das lokale System und bietet Zugriff auf die Suchdienste der Serverkomponente.

Das *RpController*-Objekt repliziert eigene, neue Annotationen nach ihrer Erstellung quasi synchron auf die Serverkomponente. Die Serverkomponente kann dann anhand der Notifkationsliste entscheiden, an welche Benutzer eine Benachrichtigung erfolgen muss (vgl. Kap. 5.2.1.8.).

Das *RpControllerOldOnes*-Objekt überträgt im Gegensatz dazu, die eigenen Annotationen, die während einer Verbindung zu einem Netzlokalen Server erstellt wurden (zur Erinnerung: auch der Fall, wenn das System alleine war) und somit noch nicht auf dem zentralen Server zur Verfügung stehen, auf den zentralen Server. Das Erstellen eines *RpControllerOldOnes*-Objekt ist also nur notwendig, wenn eine Verbindung zum zentralen Server erstellt wird. Die Übertragung wird zu Beginn einer Verbindung zum zentralen Server angestoßen und läuft asynchron, parallel im Hintergrund ab. War die Übertragung einer Annotation erfolgreich, wird sie im lokalen System als abgeglichen markiert. Außerdem entscheidet der zentrale Server über eine Meldung an andere Benutzersysteme anhand der Notifikationsliste (vgl. wieder Kap. 5.2.1.8.).

Die beiden Objekte *RpController* und *RpControllerOldOnes* setzen somit den Replikationsmechanismus um.

Die Klasse *NetworkStub* entspricht der bei den Serverkomponenten, wird also wiederverwendet (siehe Kap. 5.2.1.8.).

Das *ExController*-Objekt kümmert sich, neben dem Anstoßen der anderen, eben besprochenen Objekte, um die im Rahmen von kollaborativen Annotationen notwendige Kommunikation. Kollaborative Annotationen werden lokal über den Benutzer eingeleitet und dann über normale *Designer*-Objekte umgesetzt (vgl. Kap. 5.2.1.4. und Kap. 5.2.1.3.).

Wird vom Benutzer das Erstellen einer kollaborativen Annotation angestoßen, wird vor der Erstellung eines entsprechenden *Designer*-Objekts über den geschilderten Standardablauf, eine kollaborative Annotation eingeleitet. Dazu bietet das *CcController*-Objekt dem Benutzer zuerst eine Auswahl von bestehenden Teams für die Zusammenarbeit bei dieser Annotation an. Die Liste, der zur Verfügung stehenden Teams, erhält es vom *ExController*-Objekt. Dazu fragt das *ExController*-Objekt bei der Serverkomponente die verfügbaren Teams ab, an denen der Benutzer beteiligt ist. Entweder der Benutzer wählt nun ein vorhandenes Team aus, oder er gibt den Namen für ein neues Team an.

Wird ein neuer Name eingegeben, übergibt das *CcController*-Objekt diesen dem *ExController*-Objekt zur Neuanlage bei der Serverkomponente. Bei Neuanlage eines Teams bietet das *CcController*-Objekt anschließend dem Benutzer eine Liste, der gerade mit dem Server verbundenen Benutzer an. Diese Liste erhält es vom *ExController*-Objekt, das die Liste wiederum von der Serverkomponente erfragt. Aus der Liste wählt der Benutzer die Benutzer aus, die er zur Zusammenarbeit an der kollaborativen Annotation einladen möchte. Diese Auswahl wird abermals über das *ExController*-Objekt an die Serverkomponente übergeben und dem neu geformten Team zugeordnet. Jeder eingeladene Benutzer wird dann von der Serverkomponente informiert. Dazu sendet die Serverkomponente eine Benachrichtigung an das *ExController*-Objekt der eingeladenen Benutzer. Diese erhalten eine Anzeige und können die Einladung annehmen. Wird eine Einladung angenommen, wird dies in der Liste zum Team bei der Serverkomponente vermerkt. Das neue Team ist somit formiert. In der momentanen Implementierung kann nur der Benutzer, der eine Gruppe anlegt, Benutzer einladen. Außerdem kann er dies nur direkt beim Anlegen des Teams. In konkreten Einsatzszenarien sind auch andere Vorgehensweisen denkbar. So könnte auch jedes andere Mitglied eines Teams andere Benutzer einladen dürfen.

Nach der Wahl oder Neuaufstellung eines Teams, wird mit der Erstellung der eigentlichen Annotation begonnen. Jeder Benutzer, der die Teilnahme am Team bestätigt hat, bekommt das entsprechende *Designer*-Objekt für diese Form der Annotation, über ExController und

CcController angestoßen, angezeigt. Nach jeder lokalen Manipulation der kollaborativen Annotation durch einen Benutzer, durchläuft diese lokal die Schritte, die vor einer lokalen Ablage notwendig sind. Dies entspricht der Serialisierung des augenblicklichen Inhalts des Annotationsobjekts. Diese Serialisierung wird nun über das *ExController*-Objekt an den zentralen Server übermittelt. Dieser verteilt die Serialisierung an die *ExController*-Objekte aller Teammitglieder. Die jeweiligen *ExController*-Objekte kümmern sich jetzt darum, dass der Inhalt des *Designer*-Objekts durch die übermittelte serialsisierte Annotation ersetzt wird. Die lokale Version der Annotation wird also durch die Version vom zentralen Server ersetzt. Der zentrale Server bekommt das Ersetzen der lokalen Annotation von allen Benutzersystemen im Team bestätigt. Meldet ein Benutzersystem eine lokale Manipulation einer kollaborativen Annotation durch den Benutzer, bevor eine Bestätigung der Änderung eines anderen Systems erfolgt, besteht ein Konflikt, der durch die Serverkomponente erkannt wird. Die Serverkomponente erstellt dynamisch für die entsprechende Annotationsform ein Objekt für das Zusammenführen beziehungsweise Abgleichen der beiden in Konflikt stehenden Inhalte. Das Ergebnis des Abgleichs wird an alle Teammitglieder als nun wieder neuer Inhalt verteilt.

Für einen Gesamtüberblick über den Aufbau und Zusammenhang aller Klassen sei auf das Klassenmodell im Anhang S. 172 verwiesen.

5.2.2. Verwendete Datenstrukturen

Für eine weitere Betrachtung der vorgeschlagenen Gesamtarchitektur werden nun die zu verwendenden Datenstrukturen auf konzeptueller Ebene besprochen. Der Zusammenhang zwischen den Datenstrukturen ist als ER-Modell dargestellt (vgl. Abb. 30). Fremdschlüssel sind über das Kürzel *FK* markiert. Es sei explizit erwähnt, dass die Darstellung als ER-Modell nicht bedeutet, dass die Strukturen in einem relationalen Datenbanksystem abgelegt werden müssen.

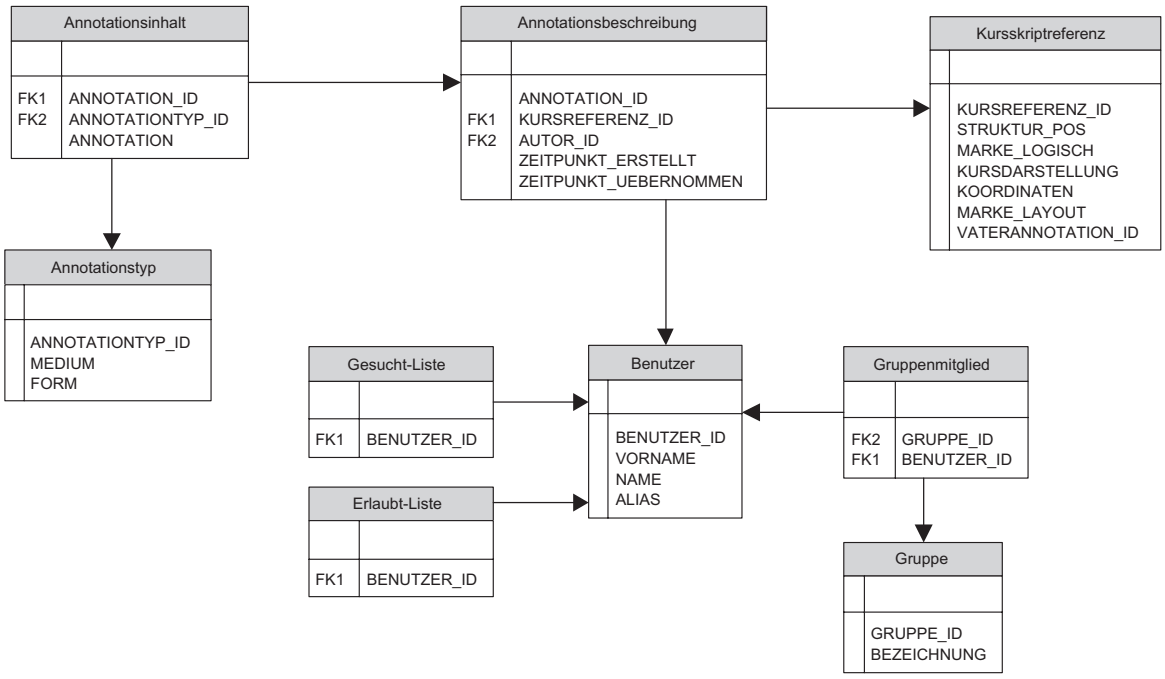


Abb. 30: ER-Modell für die verwendeten Datenstrukturen

5.2.2.1. Annotation

Die Daten zu einzelnen Annotationen werden in zwei getrennten Strukturen abgelegt: die *Annotationsbeschreibung* und der *Annotationsinhalt*. Diese Trennung ist eine logische Konsequenz aus der Vielfalt der Annotationsmöglichkeiten. In der Datenstruktur *Annotationsbeschreibung* werden die Daten zusammengefasst, die für alle Annotationen gleich welchen Typs vorhanden sind. In der Annotationsstruktur *Annotationsinhalt* werden nur die speziellen Daten zum jeweiligen Annotationstyps abgelegt. Außerdem umfasst die Datenstruktur *Annotationstyp* eine Aufzählung von erlaubten Annotationstypen.

Alle drei Datenstrukturen für Annotationen sind auf dem lokalen Benutzersystem und auf den Serversystemen (netzlokal oder zentral) strukturell identisch. Inhaltlich unterscheiden sie sich in einigen Feldern.

Annotationsbeschreibung

Die Datenstruktur *Annotationsbeschreibung* umfasst die Felder wie in Tabelle 5 angegeben.

Name	Beschreibung	Typ	Schlüssel
ANNOTATION_ID	Global eindeutiger Identifikator	guid	✓
KURSREFERENZ_ID	Identifikator einer Kursskriptreferenz	reference	
AUTOR_ID	Die BENUTZER_ID des Autors der Annotation	reference	
ZEITPUNKT_ ERSTELLT	Zeitpunkt, wann die Annotation erstellt wurde	time	
ZEITPUNKT_ UEBERNOMMEN	Zeitpunkt, wann die Annotation übernommen wurde	time	

Tab. 5: Datenstruktur Annotationsbeschreibung

Eine Annotation wird über die global eindeutige ANNOTATION_ID identifiziert. Die Thematik global eindeutiger Identifikator wurde in Kapitel 5.1.4.5. besprochen.

Eine Annotation bezieht sich immer auf genau eine Stelle im Kursskript. Eine solche Stelle wird über eine Kursskriptreferenz spezifiziert (vgl. 5.2.2.1.) In der Datenstruktur *Annotationsbeschreibung* wird der Identifikator einer Kursskriptreferenz im Feld KURSREFERENZ_ID abgelegt. Somit ist es möglich, eine Referenz für mehrere Annotationen zu benutzen.

Der Autor einer Annotation wird in der *Annotationsbeschreibung* festgehalten. Dazu wird die BENUTZER_ID des Benutzers im Feld AUTOR_ID eingetragen (vgl. 5.2.2.2.). Da neue Benutzer erst eine BENUTZER_ID nach der ersten Anmeldung bei der zentralen Annotationsverwaltungskomponente erhalten, wird folgendermaßen vorgegangen: Im lokalen Annotationsspeicher bleibt das Feld AUTOR_ID bei eigenen Annotationen prinzipiell immer leer. Erst wenn die Annotationen im Rahmen des Annotationsabgleichs auch auf den zentralen Annotationsspeicher repliziert werden, wird dort die AUTOR_ID gesetzt.

In das Feld ZEITPUNKT_ERSTELLT wird die jeweilige lokale Systemzeit eingetragen. Wenn die dabei potentiell vorhandene Differenz kritisch ist, kann die lokale Systemzeit über

Zeitserver synchronisiert werden. Kritisch wird eine Differenz, wenn zum Beispiel über die Feedback-Mechanismen aufgrund dieses Feldes eine Auswertung erstellt wird und aus dieser weitere Schlüsse gezogen werden.

Das Feld ZEITPUNKT_UEBERNOMMEN bleibt bei eigenen Annotationen im lokalen Annotationsspeicher leer. Beim Übernehmen einer fremden Annotation in den eigenen lokalen Annotationsspeicher wird dieses Feld lokal gesetzt. Außerdem setzt der zentrale Annotationsserver dieses Feld in seiner Kopie der Annotation auf die aktuelle Zeit, wenn er eine Annotation im Zuge der Replikation übernimmt.

Eigene Annotationen können also im lokalen Annotationsspeicher über eine leere AUTOR_ID oder über den leeren ZEITPUNKT_UEBERNOMMEN erkannt werden.

Annotationsinhalt

Neben der Annotationsbeschreibung ist die Datenstruktur *Annotationsinhalt* natürlich von zentraler Bedeutung. Ihr Aufbau ist in Tabelle 6 dargestellt.

Name	Beschreibung	Typ	Schlüssel
ANNOTATION_ID	Identifikator der zugehörigen Annotationsbeschreibung	reference	✓
ANNOTATIONTYP_ID	Identifikator des Annotationstyps	reference	✓
ANNOTATION	Spezielle Daten der Annotation	Large binary object	

Tab. 6: Datenstruktur Annotationsinhalt

Das Feld ANNOTATION_ID entspricht dem Feld in *Annotationsbeschreibung* und stellt die Relation zwischen den beiden Strukturen her. Die vorgestellte Architektur sieht vor, dass es zur gleichen Annotation unterschiedliche Annotationsinhalte geben kann.

Wie ist das zu begründen?

Im Zusammenhang mit den multimedialen Möglichkeiten erweitert sich die Palette der Annotationen. Aber auch wenn es die Möglichkeit gibt, eine Videosequenz als Annotation anzuhängen, heißt dies nicht, dass der Student immer die komplette Videosequenz ansehen will. Normalerweise wird er sogar vergleichsweise selten eine Videosequenz abspielen. Damit er aber trotzdem eine Vorstellung vom Inhalt der Annotation bekommt, kann es neben der Darstellung als Video auch eine Ersatzrepräsentation dafür geben. Zum Beispiel ist das erste Bild des Videos denkbar. Somit stehen für ein und dieselbe Annotation zwei verschiedene Inhalte zur Wahl.

Dieses Szenario ist durchaus auch für Textannotationen sinnvoll. Hier kann es eine Langversion mit mehreren Absätzen und einen kurzen Titel als Zusammenfassung geben. Beides entspricht der selben Annotation. Für verschiedene Darstellungen werden aber verschiedene Inhalte angezeigt.

Eine weitere Anwendung ist die Texterkennung bei handschriftlichen Annotationen. Der eine Annotationsinhalt enthält die handschriftliche Originalannotation, der andere die von einem Erkennungssystem gelieferte textuelle Repräsentation. Dieser Fall entspricht auch dem

in dieser Architektur geplanten Verwendungszweck für mehrere Inhalte innerhalb einer Annotation. Alternative Inhalte werden demnach durch automatische Transformationen erzeugt. Beispiele sind die angesprochene Texterkennung, automatische Zusammenfassungen, sprachliche Übersetzungen, automatisch erstellte Sprachausgaben und anderes mehr.

Die Aufspaltung von verschiedenen Inhalten zur gleichen Annotation wird über die ANNOTATIONTYP_ID erreicht. ANNOTATIONTYP_ID stellt eine Relation zur Datenstruktur *Annotationstyp* her. Wichtig ist hier zu betonnen, dass es sich um eine einzelne Annotation handelt, für die mehrere Inhalte zur Verfügung stehen und nicht um verschiedene Annotationen. Welcher Annotationsinhalt benutzt wird, wird von der zur Darstellung dieses Annotationstyps zugeordneten *Presenter*-Klasse entschieden. Vorteil an dieser Vorgehensweise ist, dass eine Transformation nur einmal und nicht bei jedem Darstellen durchgeführt wird. Außerdem kann die Tranformation auch auf dem zentralen Server und dort über spezielle Hard- und Software erfolgen.

Das Feld ANNOTATION stellt einen Container für jegliche Form von Inhalt zur Verfügung. Der Typ *Large Binary Object* erlaubt es hier beliebig aufgebaute und beliebig lange Objekte abzulegen. Es bleibt der konkreten Implementierung überlassen, ob bei sehr großen multimedialen Inhalten, wie zum Beispiel einem Video, der Inhalt direkt in diesem Feld gespeichert wird, oder ob hier nur ein Link zu einer anderen Quelle für den Inhalt abgelegt wird. Zu betonen ist, dass auch der Inhalt von textuellen Annotationen in diesem Feld abgelegt werden.

Annotationstyp

Tabelle 7 zeigt den Aufbau der Datenstruktur Annotationstyp.

Name	Beschreibung	Typ	Schlüssel
ANNOTATIONTYP_ID	Identifikator für einen Annotationstyp	guid	✓
MEDIUM	Medium der Annotation	enum	
FORM	Erscheinungsform des Annotation	enum	

Tab. 7: Datenstruktur Annotationstyp

Das Feld ANNOTATIONTYP_ID ist ein Identifikator für einen möglichen Typen einer Annotation. Dabei ist ein gewisser Satz von Typen fest implementiert. Neue Typen werden über den zentralen Annotationsserver abgeglichen. Probleme entstehen, wenn jeder Benutzer eigene Annotationstypen anlegt und es somit, ohne den notwendigen Überblick über die schon vorhandenen Annotationstypen, zu Redundanzen kommt. Als Lösung hierfür sieht der Architekturvorschlag vor, dass neue Annotationstypen vom Administrator des zentralen Annotationsserver (beispielsweise dem Dozenten) bestätigt werden müssen.

Im Feld MEDIUM wird die Syntax für die Interpretation des abgelegten Annotationsinhalts festgelegt. Darunter ist zu verstehen, dass hier zum Beispiel ein MIME-Type (vgl. RFC2046, 1996) angegeben werden kann.

Im Feld FORM wird abgelegt, welche Erscheinungsform beim Erstellen dieser Annotation vom Benutzer gewählt wurde. Dies entspricht einer semantischen Einordnung einer Annotation. Mögliche Werte sind: Stichpunkt, Erläuterung, Veranschaulichung, und andere (vgl. Kap. 2.2., S. 11).

Nach den vorgestellten Datenstrukturen, besteht eine Annotation also aus einem beschreibenden Kopfelement (Annotationsbeschreibung), zu dem es mindestens ein Inhaltselement (Annotationsinhalt) gibt. Die Inhaltselemente selbst sind über eine Typangabe (Annotationstyp) genauer spezifiziert und unterscheidbar. Liegt also eine Annotation inhaltlich als Text und über eine automatische Transformation auch als Sprachdatei vor, wählt die, die Annotation präsentierende Presenter-Klasse, über Angabe des Annotationstyp, den von dieser Klasse verarbeitbaren Annotationsinhalt aus und präsentiert ihn entsprechend.

Kursskriptreferenz

Im vorhergehenden Abschnitt wurden die Datenstrukturen zur Ablage der eigentlichen Annotation vorgestellt. Dies ist aber für die Verwaltung der Annotationen nicht genug, da eine Annotation immer einen Bezugspunkt besitzt. Dieser Bezug wird über Einträge in der Datenstruktur *Kursskriptreferenz* aufgebaut (vgl. Tab. 8). Jeder Annotation wird in der Datenstruktur *Annotationsbeschreibung* über das Feld KURSREFERENZ_ID genau ein solcher Eintrag zugeordnet.

Name	Beschreibung	Typ	Schlüssel
KURSREFERENZ_ID	Identifikator für eine Referenz in den Kurs	guid	✓
STRUKTUR_POS	Bezug zum Skript über eine logische Gliederung (z.B. 1.1.5§2)	string	
MARKE_LOGISCH	Bezug zu einer logischen Marke, z.B. Überschrift	string	
KURSDARSTELLUNG	Aktuelle Kursskriptdarstellung	string	
KOORDINATEN	Koordinaten bezüglich Darstellung	string	
MARKE_LAYOUT	Bezug zu einer Marke in der Darstellung, z.B. URL oder Seite	string	
VATERANNOTATION_ID	Identifikator einer Annotation, die von dieser Annotation annotiert wird.	reference	

Tab. 8: Datenstruktur Kursskriptreferenz

Wie die Datenstrukturen bisher definiert sind, kann sich eine Annotation immer nur genau auf eine Stelle - über deren Ausdehnung hier noch nichts festgelegt ist - im Kursskript beziehen. Alleine dies ist aber schon auf mindestens zweierlei Arten möglich: logisch konzeptuell und punktgenau layoutspezifisch. Mit logisch konzeptuell sind Bezüge wie *Kapitel 1 - Abschnitt 2 - Absatz 1* oder *Definition Rechter Winkel* gemeint. Punktgenau layoutspezifisch hingegen baut die Relation zum Beispiel über *Seite 10 - Abstand oberer Rand 10 cm, Abstand linker Rand 5 cm* auf (vgl. Kap. 5.1.4.7.). Für ein und dieselbe Annotation gibt es also verschiedene Möglichkeiten, mit dem Kursskript verknüpft zu sein. Da sich die Verknüpfungsmöglichkeiten aber auf wenige reduzieren lassen, werden alle zusammen in eine einzige Datenstruktur *Kursreferenz* abgelegt.

Die Felder STRUKTUR_POS und MARKE_LOGISCH erlauben die Definition einer Verknüpfung auf logischer Ebene. STRUKTUR_POS wird dabei genutzt, um eine in der logischen Struktur des Kursskript hierarchische Angabe einer Position abzulegen. Der Vorteil der hierarchischen Struktur liegt darin, dass sich eine Annotation explizit, aber auch implizit auf überordnende Einheiten beziehen kann. Wird hier nur *Kapitel 1* angegeben, bezieht sich die Annotation explizit allgemein auf das erste Kapitel. Wird hier *Kapitel 1 - Abschnitt 2* angegeben, bezieht sich die Annotation explizit auf den zweiten Abschnitt im ersten Kapitel, implizit aber auch auf Kapitel 1. Als Syntax für die Angabe empfiehlt sich folgender Aufbau:

[(<*Abschnitt Ebene*>)*][A<*Absatznummer*>][S<*Satznummer*>][W<*Wortnummer*>]

Dabei wird jede Zählung innerhalb eines Punktes beim Beginn des übergeordneten Punktes bei 1 begonnen.

Wird die erste Ebene jeweils fix mit 2 Stellen, die Absatznummer mit 6, die Satznummer mit 7 und die Wortnummer mit 11 Stellen angegeben (jeweils mit führenden Nullen), lassen sich, im für dieses Feld vorgesehenen Datentyp (char(40)), also 8 Abschnittsebenen darstellen

Beispiel:
03 05 05 00 00 00 00 00 000002 0000003 00000000026,

steht also für:
Kapitel 3.5.5. - 2. Absatz - 3. Satz - 26. Wort.

Die vorgeschlagene Form der Notation erlaubt dann eine elegante Suche nach Annotationen über einfache Präfixtextvergleiche. Werden alle Annotationen gesucht, deren Referenz in den Abschnitt 1.2 des Kursskripts zeigt, so werden zuerst alle Kursskriptreferenzen gesucht, deren Feld STRUKTUR_POS mit 0102 beginnt. Dabei werden alle Kursskriptreferenzen auf Abschnitt 1.2 und darunter liegende, also auch 1.2.2., gefunden Über die KURS-REFERENZ_ID können anschließend die zugehörigen Annotationen in der Datenstruktur *Annotationsbeschreibung* ermittelt werden.

Damit diese Form eines hierarchisch logischen Verweises möglichst robust gegen Veränderungen im Inhalt bleibt, sollte ein solcher Verweis immer so genau wie nötig angegeben werden.

Ist eine exakte und gegen Veränderungen im Kursskript einschließlich einer Umstrukturierung robuste logische Referenzierung gewünscht, steht das Feld MARKE_LOGISCH zur Verfügung. Hier kann eine logische Marke, die von der Komponente zur Verwaltung des Kursskripts unterstützt wird, eingetragen werden. Als Beispiel ist „Definition Rechter Winkel“ oder „Überschrift Rekursion“ denkbar. Da hier keine Strukturinformation enthalten ist, stellen auch Strukturänderungen kein Problem dar. Soll der Benutzer eigene Marken setzen können, so muss dies durch das Kursskriptsystem unterstützt werden. Im Annotationssystem werden die verfügbaren Marken über das ScriptStubObjekt vom Kursskriptsystem angefordert. Der Benutzer kann dann eine Marke beim Erstellen der Annotation auswählen. Es ist auch denkbar, dass das Kursskriptsystem Marken automatisch für spezielle Elemente, wie Überschriften und Definitionen generiert. Diese sind dann von der konkreten Nummerierung und damit Struktur unabhängig.

Die beiden besprochenen Felder schließen sich nicht gegenseitig aus. Werden beide Einträge angegeben, besteht eine breitere Datenbasis für Filter- und Suchfunktionen. Wegen der Forderung, dass sich das zugrundeliegende Kursskript während eines Kurses nicht verändert, sind keine Inkonsistenzen zu erwarten.

Neben den beiden besprochenen Feldern zur Festlegung einer logischen Referenzierung, stehen die drei Felder KURSDARSTELLUNG, MARKE_LAYOUT und KOORDINATEN zur Angabe einer layoutspezifischen Referenzierung zur Verfügung.

Im Feld KURSDARSTELLUNG wird dabei angegeben, auf welche konkrete Darstellung sich die beiden anderen Felder beziehen. Da es sich um eine layoutspezifische Referenz handelt ist dies eine entscheidende Information. Wird das Kursskript als HTML-Seite oder als PDF-Seite dargestellt, bedingt dies sicher andere Koordinaten und Seitenumbrüche. Wie der Eintrag im Feld KURSDARSTELLUNG für eine spezielle Darstellung genau aussieht, wird hier nicht spezifiziert. Dies bleibt der konkreten Umsetzung überlassen, da die notwendigen Informationen sehr stark vom Kursskriptsystem abhängen. Wird das Kursskript als HTML-Seiten in einem Browser dargestellt, sind erheblich mehr Parameter zu beachten, als in einem abgeschlossenen System. In jedem Fall muss sich aus dieser Angabe, die Darstellung zum Zeitpunkt der Annotationserstellung ermitteln lassen. Somit spielt die Auflösung, das Ausgabemedium und das Kursskriptmedium (z.B. PDF oder HTML) eine Rolle.

Die Angabe im Feld MARKE_LAYOUT gibt eine grobgliedrige Referenzierung innerhalb der Darstellung an. Hierunter fällt die Angabe einer Seitenzahl. Bei der Darstellung des Kursskripts als HTML-Seite kann dies die entsprechende URL sein. Es wird also der, dem Benutzer in einer bestimmten Kursdarstellung präsentierte Teil des Kursskripts, referenziert.

Innerhalb dieses Teils kann über das Feld KOORDINATEN eine weitere, verfeinernde Angabe für die Referenzierung gemacht werden. Hier kommen unter anderem der Abstand zum linken Rand und der Abstand zum oberen Rand in Frage.

Über die vorgeschlagenen Datenstrukturen gibt es nur eine Möglichkeit, eine Annotation über eine layoutspezifische Kursskriptreferenz mit einer konkreten Stelle im Kursskript in Beziehung zu setzen. Wenn es aber mehrere Darstellungen für das Kursskript gibt, wieso nur eine darstellungsgebundene Referenzierung? Der Grund dafür liegt im Ablauf des Gesamtsystems. Der Benutzer macht eine Annotation immer genau in einer Darstellung. Wechselt er die Darstellung und erstellt die Annotation noch einmal, ist dies eine neue Annotation, da der Benutzer aktiv wurde und einen eigenen Annotationserstellungsvorgang gestartet hat. Damit ist sicher gestellt, dass eine layoutspezifische Referenzierung immer genau dorthin zeigt, wo die Annotation auch vom Benutzer erstellt wurde.

Andererseits ist es in vielen Fällen jederzeit möglich, Referenzierungen direkt zwischen Darstellungen umzurechnen. Dann müssen aber auch keine zwei Koordinaten gespeichert werden. Ein Beispiel hierfür ist die Umrechnung von Koordinaten bezüglich der Darstellung eines Kursskripts, welches als Bilddatei gegeben ist, in eine andere Bildschirmauflösung.

Die beiden bisher angesprochenen Referenzierungsvarianten bezogen sich auf das Kursskript: Einmal auf die Struktur und einmal auf die Darstellung. Es gibt in der Architektur noch eine dritte Form der Referenzierung. Dabei bezieht sich eine Annotation nicht direkt auf das Kursskript, sondern auf eine andere Annotation. Somit ist es möglich, eine Annotation zu annotieren. Die entstehende Referenzierungskette endet immer in einer Annotation, die sich auf das Kursskript bezieht. Damit entspricht auch diese Form der Referenz indirekt einem Verweis ins Kursskript. Für Referenzierungen dieser Art steht das Feld VATERANNOT_ID zur Verfügung. In dieses Feld wird der Annotationsidentifikator der zu referenzierenden Annotation eingetragen.

Damit sind alle von dieser Architektur zur Verfügung gestellten Varianten zur Referenzierung von Stellen im Kursskript vorgestellt. Die verschiedenen Varianten stehen dabei nicht in

einer „entweder oder“ Beziehung. Es können vielmehr in einer Referenz, beliebig viele Felder gleichzeitig ausgefüllt sein.

Die Datenstruktur *Kursskriptreferenz* wird, wie die drei Datenstrukturen zu Annotationen, sowohl auf dem lokalen Benutzersystem, als auch über den Abgleichmechanismus auf dem zentralen Serversystem abgelegt.

5.2.2.2. Benutzer

Neben den besprochenen Datenstrukturen zur Speicherung der eigentlichen Annotationen mit ihrer Referenz sind noch weitere Datenstrukturen eher administrativer Art zur Sicherstellung eines geregelten Ablaufs notwendig. Eine dieser Datenstrukturen ist *Benutzer* (vgl. Tab. 9).

Name	Beschreibung	Typ	Schlüssel
BENUTZER_ID	Identifikator für einen Benutzer	guid	✓
NAME	Name des Benutzers	string	
VORNAME	Vorname des Benutzers	string	
ALIAS	Pseudonym für den Benutzer	string	

Tab. 9: Datenstruktur Benutzer

Die Datenstruktur *Benutzer* repräsentiert einen Benutzer des Systems. Die beiden Felder NAME, VORNAME werden für persönliche Angaben genutzt. Es können beliebige weitere Felder ergänzt werden. Im Feld ALIAS wird ein systemweit eindeutiger Alias für die Person vergeben. Der Alias hat zwei Aufgaben: Zum Einen können über den Alias zwei Personen mit gleichem Vornamen und Namen unterschieden werden. Zum Anderen lässt sich das System so konfigurieren, dass aus datenschutzrechtlichen und auch psychologischen Gründen anderen Benutzern gegenüber nur der Alias angezeigt wird.

Eine vollständige Liste der Benutzer wird auf dem zentralen Annotationsserver geführt. Dort können auch vom Administrator (z.B. Dozenten) generische Benutzer wie *Alle* und *Studenten* angelegt werden.

Das lokale System des Benutzers enthält nur den Benutzer selbst. Immer wenn sich ein Benutzersystem beim zentralen Annotationsserver anmeldet und Annotationen abgleicht, übergibt es seine lokale Benutzerstruktur. Der zentrale Annotationsserver prüft, ob der Benutzer schon in der zentralen Liste enthalten ist. Wenn nicht, wird der Benutzer neu eingetragen. Dabei wird dann auch eine neue BENUTZER_ID vergeben. Außerdem wird überprüft, ob der Alias systemweit einzigartig ist. Gegebenenfalls wird ein neuer Alias beim Benutzersystem angefordert.

Die Datenstruktur *Benutzer* stellt die Grundlage für die weiteren aufgeführten Datenstrukturen dar.

5.2.2.3. Gruppen

Gesucht-Liste

Anforderung A5 (vgl. Kap. 5.1.1.) fordert die Möglichkeit, Annotationen austauschen zu können. Dazu ist es notwendig, Kenntnis über Annotationen anderen zu erlangen. Damit keine Überflutung mit solchen Informationen auftritt, legt jeder Benutzer fest, für welche anderen Benutzern er einen Hinweis auf eine neue Annotation wünscht. Diese Benutzer werden jeweils auf dem lokalen Benutzersystem in der so genannten *Gesucht Liste* geführt (vgl. Tab. 10). Hierzu werden nur die BENUTZER_ID's der gewünschten Benutzer in die Liste eingetragen.

Name	Beschreibung	Typ	Schlüssel
BENUTZER_ID	Identifikator für einen anderen Benutzer	reference	✓

Tab. 10: Datenstruktur Gesucht-Liste

Meldet sich ein Benutzersystem beim Start bei einem zentralen Annotationsserver oder einem netzlokalen Annotationsserver an, übermittelt es seine eigene *Gesucht-Liste*. Damit hat die jeweilige Serverkomponente die Information, wessen Annotationen für diesen Benutzer interessant sind (vgl. Kap. 5.2.1.8.).

Erlaubt-Liste

Um die Privatheit der Annotationen sicherzustellen (vgl. Anforderung A6, Kap. 5.1.1.), werden Informationen über neue Annotationen nicht nur aufgrund eines Eintrags in einer *Gesucht-Liste* weitergegeben. Es wird ebenfalls geprüft, ob der jeweilige Autor auch einverstanden ist, dass ein spezieller anderer Benutzer seine Annotationen einsehen darf. Dazu wird bei jedem Benutzer, also bei jedem potentiellen Autor einer Annotation, eine *Erlaubt-Liste* geführt (vgl. Tab. 11). In diese, bei jedem Benutzersystem geführte Liste, wird eingetragen, wem der Benutzer Einsicht in seine Annotationen gestatten will. Auch diese Liste wird beim Anmelden des Benutzersystems an dem zentralen Annotationsserver oder einem netzlokalen Annotationsserver übermittelt. Nun hat der jeweilige Annotationsserver die Möglichkeit, Informationen über das Erstellen einer Annotation, im Einklang mit den Wünschen der Benutzer, automatisiert an andere Benutzer weiterzugeben (vgl. Kap. 5.2.1.8.).

Name	Beschreibung	Typ	Schlüssel
BENUTZER_ID	Identifikator für einen anderen Benutzer	reference	✓

Tab. 11: Datenstruktur Erlaubt-Liste

Das Führen einer *Erlaubt-Liste* entspricht einer restriktiven Variante. Nur Benutzer, die explizit die Erlaubnis erhalten, dürfen über Annotationen informiert werden und diese übernehmen. Hier steht also die Privatheit der Annotationen im Vordergrund.

Eine offene Variante ist das Führen einer *Bann Liste*. In eine solche Liste werden Benutzer eingetragen, denen kein Zugirff auf die eigenen Annotationen gewährt werden soll. Damit

wird implizit jedem anderen der Zugriff automatisch gestattet. Hier ist also der freie Austausch von Annotationen vorrangig.

Von der Umsetzung her unterscheiden sich die beiden Varianten nur in der logischen Verknüpfung der Information der *Gesucht-Liste* mit der Information der *Bann-* oder *Erlaubt-Liste*. Es hängt vom jeweiligen Benutzerkreis und den kulturellen Gegebenheiten ab, welche Variante im speziellen Fall geeigneter ist.

5.2.2.4. Teams

Die in Abschnitt 3.3.3. aufgeführten Gruppenannotationen, werden über den Mechanismus der *Gesucht-*, *Bann-* oder *Erlaubt-Liste* unterstützt. Gruppen entstehen hier implizit durch Verknüpfen zweier Listen. Für kollaborative Annotationen (vgl. 3.3.4.) werden Gruppen von Benutzern explizit zusammengestellt, sogenannte Teams. Eine solche Teamzusammenstellung hat also nicht gezwungenermaßen, etwas mit den Benutzerwünschen zu tun. Es kann sich auch um eine vom Dozenten vorgegebene Teambildung handeln (vgl. Gruppenarbeit).

Zur Verwaltung der Teams werden zwei Datenstrukturen benötigt, die im Folgenden beschrieben werden. Beide Datenstrukturen werden nur auf dem zentralen Annotationsserver abgelegt. Wie in Kapitel 5.1.4.6. begründet, werden im aktuellen Ansatz nur kollaborative Annotation über den zentralen Annotationsserver unterstützt.

Team

Jedes Team wird als eigenständiges Objekt betrachtet. Jedes solches Objekt ist durch einen systemweit eindeutigen Identifikator und eine Bezeichnung charakterisiert. Beide Informationen werden in der Datenstruktur *Team* in entsprechenden Feldern abgelegt (vgl. Tab. 12).

Name	Beschreibung	Typ	Schlüssel
TEAM_ID	Identifikator für ein Team	guid	✓
BEZEICHNUNG	Bezeichnung für das Team	string	

Tab. 12: Datenstruktur Team

Die Datenstruktur *Team* wird von der *Central Server*-Komponente verwaltet.

Teammitglieder

Ist ein Team angelegt, können ihm über die Datenstruktur *Teammitglieder* Benutzer als Mitglieder zugeordnet werden (vgl. Tab. 13).

Dazu wird in der Datenstruktur *Teammitglieder* der Identifikator des Teams und der Identifikator des Benutzers abgelegt. Über die somit entstehende *n zu m* - Zuordnung können also zu einem Team mehrere Benutzer, aber auch mehrere Gruppen zu einem Benutzer verknüpft werden. Das Feld BESTAETIGT gibt an, ob der Benutzer die Einladung in das Team angenommen hat. Erst dann gehört er offiziell zum Team.

Prinzipiell können alle, dem zentralen Annotationsserver bekannte BENUTZER_ID's, in die Struktur *Teammitglieder* eingetragen werden. Somit können auch generische Benutzer wie

Alle und *Studenten* in ein Team aufgenommen werden. Die Bestätigung erfolgt dabei direkt beim Eintragen.

Name	Beschreibung	Typ	Schlüssel
TEAM_ID	Identifikator für ein Team	reference	✓
BENUTZER_ID	Identifikator eines Benutzers	reference	✓
BESTAETIGT	Hat der Benutzer die Einladung zum Team angenommen?	boolean	

Tab. 13: Datenstruktur Teammitglieder

5.3. Komponenteninteraktion

Nachdem nun die Komponenten und die Datenstrukturen der Annotationsarchitektur benannt sind, werden einige dynamische Aspekte auf technischer Ebene vorgestellt. Dazu werden vier Standardvorgänge, welche das geforderte Leistungsspektrum weitgehend abdecken und während der vorangegangenen Betrachtungen noch nicht in sich abgeschlossen, über Komponentengrenzen hinweg, besprochen wurden, vorgestellt:

- Ein Benutzer macht eine Annotation
- Ein Benutzer übernimmt eine Annotation
- Ein Student wiederholt den Stoff
- Der Dozent reflektiert seine Vorlesung

Die Diskussion dieser vier Standardvorgänge stützt sich auf die Klassenmodelle der einzelnen Komponenten (vgl. Kap. 5.2.1.). Außerdem ist der Gesamtüberblick über das Klassengerüst des Annotationssystem für ein Verständnis der Ausführungen empfehlenswert (vgl. Anhang S. 172).

Der Detaillierungsgrad der Ausführungen schwankt. Ziel der Ausführungen ist, den Leser von der Funktionsfähigkeit der Architektur zu überzeugen, sich dabei aber nicht in Selbstverständlichkeiten zu verlieren. Weniger offensichtliche, aber konzeptuell interessante Dinge werden hervorgehoben. Wo möglich, werden verschiedene Varianten von Benutzungsschnittstellen in Abbildungen begleitend vorgestellt.

5.3.1. Ein Benutzer macht eine Annotation

Ausgangspunkt dieses Standardvorgangs ist die *ControlCenter*-Komponente (vgl. Abb. 20). Um eine Annotation zu erstellen, wählt der Benutzer in der Benutzungsoberfläche der *ControlCenter*-Komponente eine Erscheinungsform einer Annotation aus. Wie bei den Ausführungen zur *ControlCenter*-Komponente beschrieben, handelt es sich bei der Benutzungsoberfläche um ein Menü, wahlweise mit Iconleiste (vgl. Kap. 5.2.1.2.). Die zur Verfügung stehenden Erscheinungsformen und somit die dynamisch erzeugten Menüeinträge richten sich nach der aktuellen Darstellung des Kursskripts und sind in der Registry der *ControlCenter*-Komponente hinterlegt. In Abbildung 31 werden verschiedene Benutzungsoberflächen beispielhaft präsentiert.

Vorteil dieses Vorgehens ist, dass eine spätere Erweiterung der Palette von Annotationstypen, oder aber auch das Hinzufügen neuer Kursskriptdarstellungen, problemlos - in Sprachen

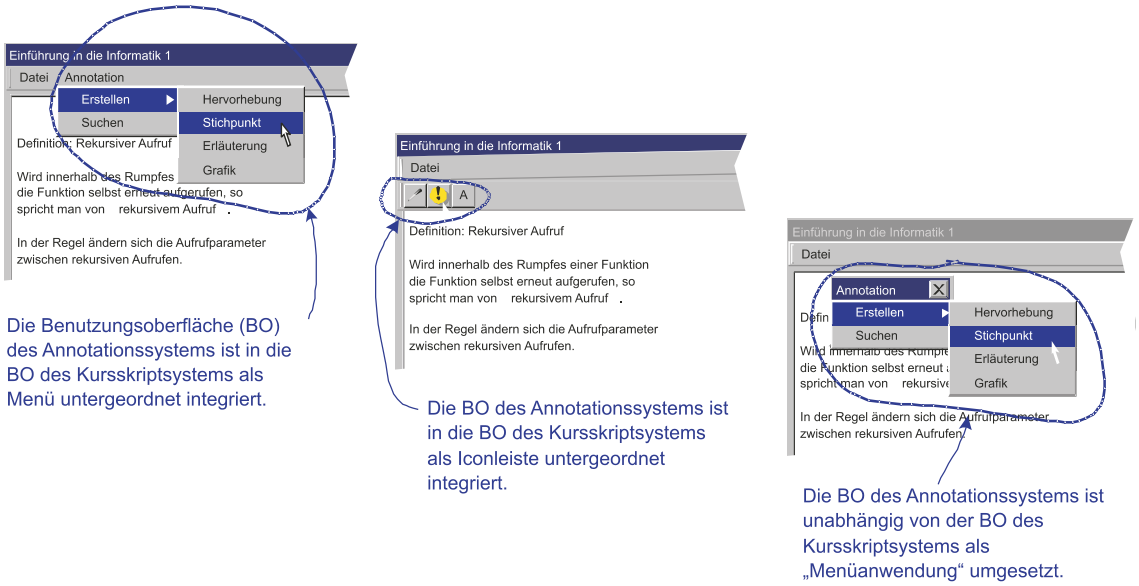


Abb. 31: Beispielhafte Benutzungsoberflächen der ControlCenter-*Komponente*

wie Java sogar ohne komplette Neuübersetzung - möglich ist. Es müssen nur die entsprechenden Klassen implementiert und die hinzugekommenen Optionen über Einträge in der Registry bekannt gegeben werden. Somit kann jeder Benutzer problemlos, die für ihn ergonomisch angenehmste Menüführung zusammenstellen.

Je nach aktueller Darstellung und gewählter Erscheinungsform der Annotation erstellt die *Design*-Komponente ein passendes Benutzungsoberflächenobjekt und ein passendes *DsDesigner*-Objekt. Was passende Objekte sind, bestimmen entsprechende Factory-Klassen anhand von Einträgen in der Registry.

Dieses Vorgehen bietet abermals den Vorteil, dass neue Erscheinungsformen sehr einfach hinzugenommen werden können. Dies ist insbesondere deshalb interessant, da so auch äußerst effizient neue ergonomisch angepasste Varianten von *DsDesigner*-Objekten, zum Beispiel über Vererbung, implementiert werden können. Das System lässt sich damit also einfach an Benutzerwünsche und Lehrsituationen anpassen (vgl. Abb.32). Im Extremfall kann sich jeder Benutzer eigene Klassen für die Erstellung von Annotationen schreiben (vgl. Plugin-Entwicklung für Eclipse, Bolour, 2003). Durch die starke Kapselung sind dazu nur geringe Kenntnisse über die Gesamtarchitektur notwendig. Deshalb und des geringen Aufwands wegen, ist eine Erweiterung durch den Benutzer keineswegs utopisch.

Hat der Benutzer das Erstellen der Annotation beendet, wandelt das *DsDesigner*-Objekt noch, die über die vermittelnden Controller-Objekte vom ScriptStub-Objekt erhaltene, aktuelle Kursskriptreferenz, entsprechend der Anforderungen der Erscheinungsform und Darstellung ab. Die neue Annotation mit der angepassten Referenz übergibt das *DsDesigner*-Objekt über das *DsController*-Objekt dem *CcController*-Objekt zur Speicherung. Nun beendet sich die *Design*-Komponente. Das Anpassen der Kursskriptreferenz direkt durch das *DsDesigner*-Objekt ist eine konsequente Umsetzung des Kapselungsgedankens (vgl. z.B. Designregeln und -heuristiken, S. 195, Oesterreich, 1998). Nur das *DsDesigner*-Objekt selbst entscheidet, welche Attribute der Kursskriptreferenz (entsprechen den Feldern der Datenstruktur Kursskriptreferenz) dieser Annotation Sinn machen und wie sie anzupassen sind.

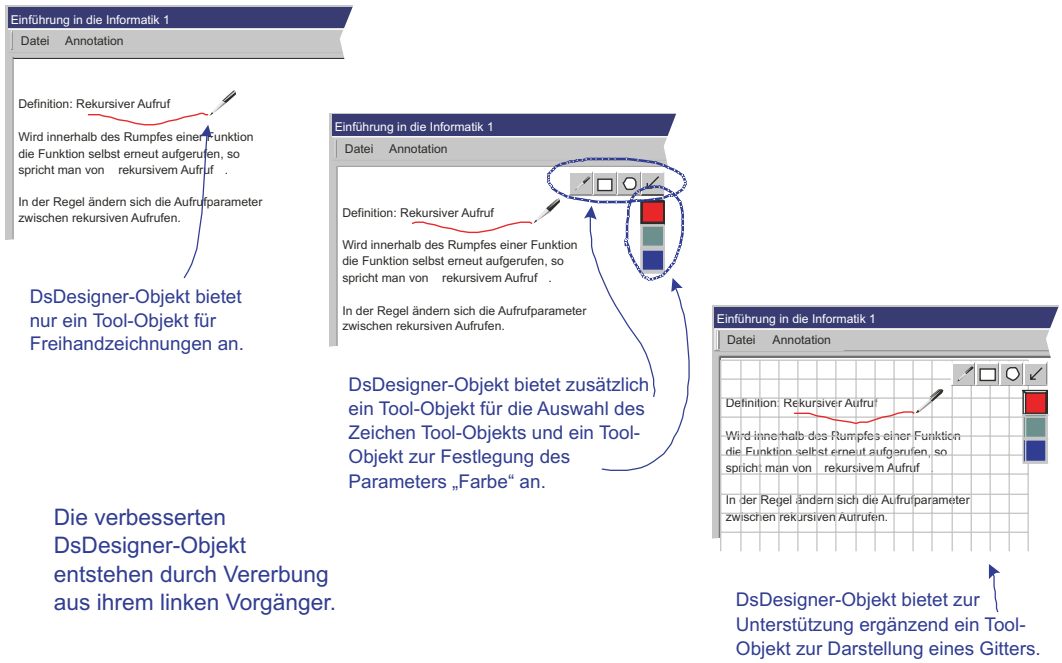


Abb. 32: Angepasste Benutzungsoberflächen verschiedener DsDesigner-*Objekte*

Hat das *CcController*-Objekt die Annotation mit Referenz erhalten, gibt es diese nach Anreicherung um Benutzer- und anderen allgemeinen Informationen (z.B. Zeit) an den Controller des lokalen Annotationsspeichers weiter. Das Einfügen dieser Zusatzinformationen übernimmt das *CcController*-Objekt, um sicherzustellen, dass alle Annotationen diesen Satz an Attributen besitzen.

Die eigentliche Speicherung übernimmt die Persistenzschicht im lokalen Annotationsspeicher. Das *PersistenzController*-Objekt erstellt einmal beim Start des lokalen Annotationsspeichers über ein *PersistenzManagerFactory*-Objekt ein *PersistenzManager*-Objekt. Welches *PersistenzManager*-Objekt zu erstellen ist, entnimmt das *PersistenzManagerFactory*-Objekt der Registry der *ControlCenter*-Komponente.

Das *PersistenzManager*-Objekt übernimmt die Annotation mit Referenz und legt sie entsprechend des gewählten Speicherverfahrens ab. Hierbei kapselt es etwaige Umformungen struktureller oder inhaltlicher Art (vgl. *Marshalling* bei verteilten Anwendungen, oder *Serialisieren* von Objekten).
Vorteil dieses Vorgehens ist, dass alle für die jeweilige Speicherarchitektur notwendigen speziellen Operationen im jeweiligen *PersistenzManager*-Objekt gekapselt sind. Über die Festlegung des gewünschten Speicherverfahrens durch einen Eintrag in der Registry der *ControlCenter*-Komponente kann so jeder Benutzer, die für ihn beste Variante wählen. Hat er auf seinem Computersystem eine SQL-Datenbank zur Verfügung, wird dies die bevorzugte Wahl für die Speicherung sein. Trotzdem erfordert das entstehende Annotationssystem nicht grundsätzlich eine SQL-Datenbank. Als einfache, auf jedem System verfügbare Variante steht immer die dateibasierte Ablage zur Verfügung.

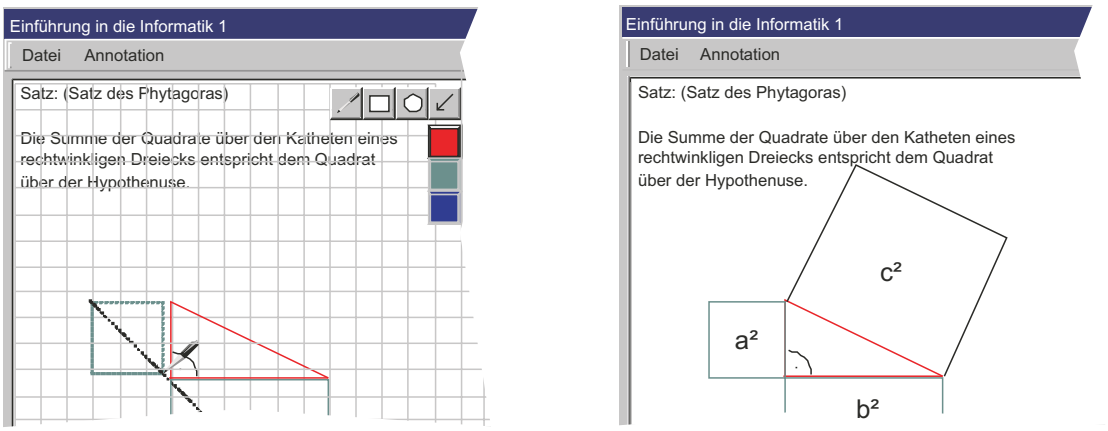
Ist die Annotation nun erstellt und lokal abgelegt, informiert die *ControlCenter*-Komponente das *ExController*-Objekt in der *Exchange*-Komponente mittels eines entsprechenden Methodenaufrufs über die neue Annotation. Diese Information wird nun asynchron über das *RpController*-Objekt der *Exchange*-Komponente an den jeweiligen Annotationsserver wei-

tergegeben. Für die lokale *ControlCenter*-Komponente im erstellenden System ist damit die Erstellung der Annotation abgeschlossen.

Auf Seiten des asynchron benachrichtigten Annotationsspeichers erfolgen zwei Dinge:

Zum Einen überprüft der von *AsController* erbende Controller der Serverapplikation (zentral oder netzlokal) anhand der Gesucht- und Erlaubt-Liste, ob andere Benutzer an dieser Annotation interessiert sind. Ist dies der Fall, wird das *ImportController*-Objekt in der *Exchange*-Komponente des jeweiligen, interessierten Benutzers über das *NetzwerkStub*-Objekt von der neuen Annotation in Kenntnis gesetzt. Dieses *ImportController*-Objekt gibt die Information zur Anzeige an den Benutzer an die *ControlCenter*-Komponente weiter.

Zum Anderen fordert die Serverkomponente die Annotation, die zugehörige Kursskriptreferenz und den Benutzer an. Vor der Ablage der Annotation und der Kursskriptreferenz ergänzt das *AsController*-Objekt die Benutzerinformation in der Annotation.



Anzeige der Annotation zur Zeit der Erstellung mit Hilfe der Design-Komponente

Anzeige der fertigen Annotation über die Presentation-Komponente

Abb. 33: Anzeige einer Annotation beim Erstellen und bei der Darstellung

Zu erwähnen bleibt, dass die sichtbare Annotation mit Beendigung des *DsDesigner*-Objekts in der Anzeige vorerst verschwindet. Durch einen abschließenden, vom *CcController*-Objekt über das ScriptStubObjekt in der Kursskriptkomponente ausgelösten Neuaufbau (refresh) der Kursskriptdarstellung wird die neue Annotation aber erkannt und über die Presenter-Komponente entsprechend präsentiert (vgl. Abb. 33). In diesem Ablauf spiegelt sich also abermals die konsequente Trennung der Erstellung und der Darstellung wieder.

5.3.2. Ein Benutzer übernimmt eine Annotation

Jeder Benutzer hat die Möglichkeit, Annotationen anderer zu übernehmen. Ein solcher Vorgang teilt sich in zwei Abschnitte: Die Auswahl, der zu übernehmenden Annotation und die eigentlichen Übernahme in den lokalen Annotationsspeicher.

Für die Auswahl der Annotation gibt es mehrere Möglichkeiten. Die Übernahme in den lokalen Annotationsspeicher erfolgt technisch aber immer in gleicher Weise und wird am Ende dieses Abschnitts erläutert.

Ein Benutzer erfährt von einer neuen Annotation eines anderen Benutzers durch die aktive Benachrichtigung durch seine ControlCenter-Komponente (vgl. Abb. 34).

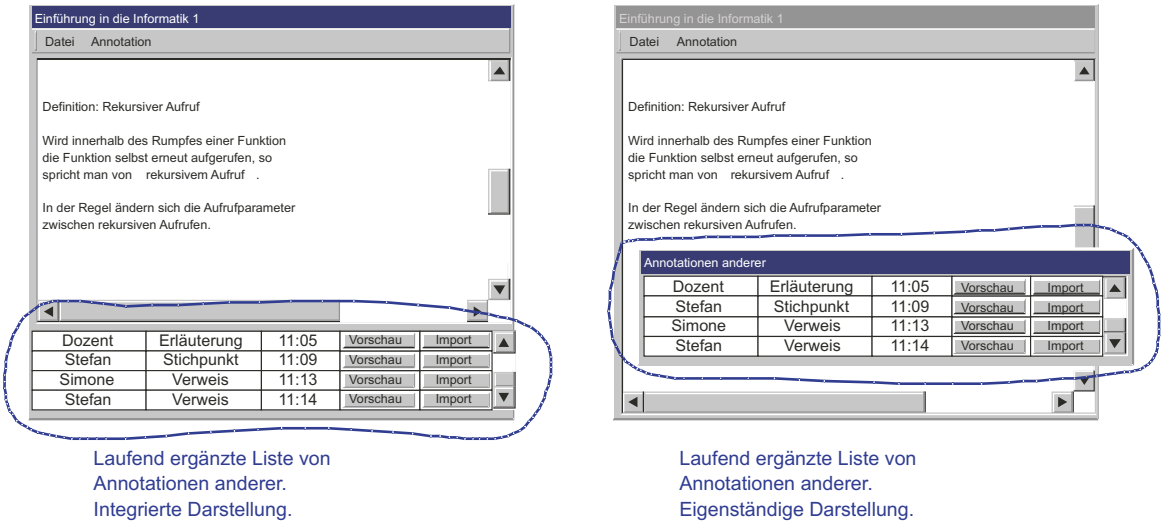


Abb. 34: Beispiele für die Darstellung der Benachrichtigung über Annotationen anderer

Die Umstände, die eine solche Benachrichtigung bewirken, wurden unter Abschnitt 5.3.1. beschrieben. Erkennt der Benutzer eine Annotation alleine anhand der Notifikation, kann er sie ohne weitere Betrachtung übernehmen.
Der Student kennt beispielsweise die Annotation, die der Dozent gerade vor seinen Augen verfasst hat. Oder Studenten sitzen in einer Lerngruppe zusammen. Auch hier kann er das Erstellen der Annotation schon beim anderen Studenten verfolgt haben. Solche Annotationen können sinnvoll direkt nach der Benachrichtigung übernommen werden.

Bei allen anderen aktiv vorgeschlagenen Annotationen ist eine direkte Übernahme unwahrscheinlich. Deshalb hat jeder Benutzer die Möglichkeit, bei der Benachrichtigung über eine neue Annotation, sich diese, als Vorschau anzeigen zu lassen. Dazu wird die *Annotation Browser*-Komponente verwendet. Hat sich der Benutzer in der *ControlCenter*-Komponente für eine Vorschau einer neuen Annotation entschieden, holt das *CcController*-Objekt über das *ExController*-Objekt der *Exchange*-Komponente die Annotation von der Serverkomponente auf das lokale Benutzersystem. Die Annotation wird dann über das *AbController*-Objekt der *Annotation Browser*-Komponente dem *AbPreview*-Objekt übergeben. Überzeugt die Vorschau den Benutzer, übernimmt er die Annotation. Davon wird das *CcController*-Objekt unterrichtet. Ansonsten beendet der Benutzer die Vorschau.

Neben der aktiven Benachrichtigung über eine neue Annotation durch das System kann der Benutzer auch selbst Initiative ergreifen und eine Annotation suchen. Dazu öffnet er über die *ControlCenter*-Komponente die *Annotation Browser*-Komponente. Dort kann er im Benutzungsschnittstellenobjekt *AbCriteria* eine Suchspezifikation zusammenstellen. Diese Suchspezifikation wird vom *AbController*-Objekt über das *CcController*-Objekt an das *ExCon-*

troller-Objekt weitergegeben. Das *ExController*-Objekt leitet die Suchspezifikation über das Netz an das zuständige *AsController*-Objekt. Dieses startet über das *PersistenzController*-Objekt die Suchfunktionalität des aktuellen *PersistenzManager*-Objekts. Dabei wird die Notifikationsliste der Serverkomponente mit berücksichtigt. Somit ist sichergestellt, dass nur Annotationen, bei denen Gesucht- und Erlaubt-Liste zusammenpassen, gefunden werden.

Als Ergebnis liefert das *PersistenzManager*-Objekt eine Liste von Annotationsbeschreibungen über den Aufrufweg zurück an das *AbController*-Objekt. Dieses lässt die Liste von dem Benutzungsschnittstellenobjekt *AbUI* über ein *AbTree*-Objekt mit verknüpftem *AbList*-Objekt anzeigen. Zu jeder Annotation kann der Benutzer dort eine Vorschau anfordern. Der damit angestoßene Ablauf zum Holen, der für die Vorschau notwendigen Daten von der Serverkomponente, entspricht dem oben aufgeführten. Ist eine Annotation interessant, übernimmt der Benutzer diese (eventuell auch ohne Vorschau). Auch hier wird das *CcController*-Objekt informiert, da es die eigentliche Übernahme koordiniert.

Soll eine Annotation also schlussendlich in den lokalen Annotationsspeicher übernommen werden, prüft das *CcController*-Objekt zuerst, welche Daten der Annotation noch fehlen. Gab es zum Beispiel eine Vorschau, sind somit weniger Daten notwendig. Dies hängt damit zusammen, dass eine Annotation je Annotationstyp einen anderen Annotationsinhalt besitzen kann (vgl. Kap. 5.2.2.1.). Eine Vorschau greift nur auf einen Annotationstyp und damit Annotationsinhalt zurück. Bei der Übernahme einer Annotation werden alle Annotationsinhalte übernommen. Anschließend fordert das *CcController*-Objekt über das *ExController*-Objekt die fehlenden Daten von der Serverkomponente an. Hat das *CcController*-Objekt alle Daten der Annotation erhalten, trägt es in der Annotationsbeschreibung den Übernahmezeitpunkt und eventuell weitere Felder ein. Nun übergibt es die Annotation dem *LsController*-Objekt zur Speicherung im lokalen Annotationsspeicher.

Diese Umsetzung des Standardvorgangs über auf den ersten Blick lang wirkende Delegationsketten erscheint eventuell etwas umständlich. Bei genauerem Hinsehen lässt sich aber zum Einen recht schnell eine gute Wiederverwendung von Programmteilen erkennen. Zum Anderen wird durch die strikte Einhaltung der Kapselung von Teilaspekten, wie zum Beispiel der Netzwerkzugriffe, eine gute Wartbarkeit des Gesamtsystems erreicht.

5.3.3. Ein Student wiederholt den Stoff

Die Wiederholung des Stoffes entspricht dem, sich mit dem Kursskript und den Annotationen auseinander setzen. Daraus ergeben sich unmittelbar zwei mögliche Szenarien: Der Student betrachtet das Kursskript einschließlich der Annotationen, oder aber der Student arbeitet nur mit den Annotationen. Beide Szenarien werden im Folgenden angesprochen.

Geht der Student das Kursskript mit den Annotationen durch, so stellt die Kursskriptkomponente den Ausgangspunkt dar. Die Kursskriptdarstellung erlaubt sinnvollerweise ein lineares Durchlaufen des Kursskripts, das direkte Anspringen von Stellen im Kursskript anhand eines Inhaltsverzeichnisses oder eines Index und eventuell eine Freitextsuche über das Kursskript hinweg. Da diese Arbeit den Fokus auf die Annotationen setzt und unter anderem auch von der Erweiterung eines bestehenden Systems zur Kursskriptdarstellung ausgeht, werden hierzu keine genaueren Ausführungen zu den internen Vorgängen gegeben.

Von besonderer Wichtigkeit ist dahingegen die Schnittstelle zu den Annotationen hin. Immer wenn die Kursskriptkomponente den angezeigten Teil des Kursskripts ändert, auf welche der oben genannten Arten auch immer, informiert sie das ControlCenter-Objekt mit Hilfe einer

asynchronen Meldung an das ScriptStub-Objekt. Inhalt der Meldung ist eine aktuelle Kursskriptreferenz. Diese Kursskriptreferenz bereitet das SkriptStub-Objekt auf und bildet daraus eine Suchanfrage an die *Local Storage*-Komponente. Die Suchanfrage wird über das *CcController*-Objekt an das *LsController*-Objekt weitergegeben und dort mit Hilfe der Persistenzschichtklassen ausgeführt. Als Ergebnis der Suche erhält das *CcController*-Objekt eine Liste von Annotationen, die zur momentan angezeigten Kursskriptstelle in der aktuellen Darstellung relevant sind. Diese Annotationen werden nun durch entsprechende *Presenter*-Objekte repräsentiert. Dies kann bedeuten, die Darstellung wird direkt in die Kursskriptdarstellung integriert. Es kann aber auch bedeuten, die Annotationen werden in einer Liste neben dem Kursskript aufgeführt, was zum Beispiel für Videosequenzen oder Audio sinnvoll ist (vgl. Abb. 35).

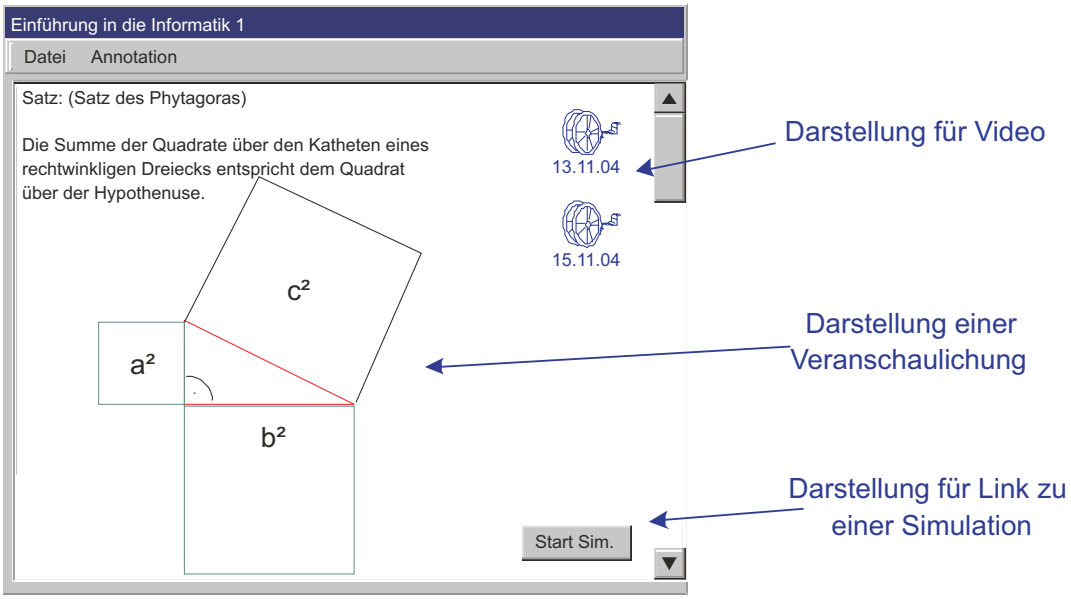


Abb. 35: Unterschiedliche Darstellungen bei verschiedenen Erscheinungsformen von Annotationen

Diese große Flexibilität in der Darstellung von Annotationen ist eine Stärke dieser Architektur. Damit wird es möglich, verschiedenartigste Annotationen innerhalb der Architektur über weite Strecken gleich zu behandeln. Erst in der Darstellung unterscheiden sich die möglicherweise multimedialen Annotationen. Dieses Vorgehen ist mit erfolgreichen Konzepten wie OLE im Bereich von Microsoft (vgl. Brockschmidt, 1995) oder auch Plugins in Webbrowsern (vgl. Netscape, 2004) vergleichbar.

Welches *Presenter*-Objekt nun für welche Annotation benutzt wird, legt das *PsPresenterFactory*-Objekt anhand von aktueller Kursskriptdarstellung, Typ der Annotation und Einträgen in der Registry der *ControlCenter*-Komponente fest. Damit kann auch die Palette der Annotationsdarstellungen ohne großen Aufwand geändert oder ergänzt werden. Im Extremfall kann wieder jeder Benutzer seine eigene Darstellung wählen oder auch eine neue entwerfen.

Erwähnt werden muss, dass von der Architektur vorgegeben wird, dass jedes *Presenter*-Objekt eine minimierte und eine fokussierte Darstellung der Annotation anbietet. Damit können global Methoden zur Reduzierung der visuellen Komplexität verwirklicht werden. Die konkrete Umsetzung der minimierten Darstellung ist insoweit festgelegt, als dass ein Icon fester Größe geliefert werden muss.

Die andere Möglichkeit den Stoff zu wiederholen, liegt in der Arbeit direkt mit den Annotationen. Hierfür ist die *Annotation Browser*-Komponente zuständig. Über das *ControlCenter*-Objekt gestartet, erlaubt es dem Benutzer, die über die *Local Storage*-Komponente abgelegten Annotationen, aufzulisten. Dabei kann über die Angabe von Suchkriterien eine Filterung der Annotationen erreicht werden. Als Attribute stehen alle Felder der Annotationsbeschreibung zur Verfügung. Somit kann also unter anderem auch, die im allgemeinen Teil der Arbeit vorgeschlagene Reduktion auf Stichpunkte zum Kursskript, tatsächlich durchgeführt werden (vgl. 4.2.2.1.). Außerdem kann für die Suche auch Text im Annotationsinhalt herangezogen werden.

Zur Darstellung der Liste von Annotationen benutzt dieser Architekturvorschlag eine Baumansicht mit angegliedertem Listenfenster. Dies erfolgt aus zweierlei Gründen: Zum Einen hat diese Repräsentation aus Sicht des Benutzers offensichtliche Vorteile. Sie ist intuitiv und erlaubt, Aufrufe von Sortier-und Gruppierfunktionalität ergonomisch in die Benutzungsoberfläche einzubinden. Zum Anderen birgt diese Darstellung auch weniger augenfällige Vorteile für die technischen Umsetzung im Hintergrund. Durch die Baumstruktur werden die Daten gegliedert, was in Kombination mit der untergeordneten Listendarstellung, die auf einmal anzuzeigenden und somit zu ermittelnden Ergebnismengen, von vorne herein einschränkt. Dies kommt der Performanz eines solchen Systems sehr zu gute.

Selektiert der Benutzer eine Annotation in der Liste der Annotationen im Annotationsbrowser, kann er die Annotation als Vorschau innerhalb des Annotationsbrowsers anzeigen, oder aber direkt an die zugehörige Stelle im Kursskript springen. Für letzteres wird die entsprechende Kursskriptreferenz vom ABController an das CcController-Objekt übergeben. Dieses benutzt das SkriptStub-Objekt, um die Kursskriptdarstellungskomponente entsprechend zu positionieren.

Jedem Benutzer stehen somit umfassende Möglichkeiten zur Wiederholung des Stoffes anhand der Annotationen zur Verfügung.

5.3.4. Der Dozent reflektiert seinen Kurs

Da der Dozent einen Kurs meist mehrere Male hält, ist er in der Regel an einer Weiterentwicklung seines Kursskripts interessiert. Dazu reflektiert er einen gehaltenen Kurs. Im Folgenden sollen zwei Schritte, die dem Dozenten dafür speziell in dieser Architektur zur elektronischen Unterstützung zur Verfügung gestellt werden, näher betrachtet werden.

Zum Einen hat der Dozent schon während eines Kurses die Möglichkeit, Annotationen zu verfassen, die Bemerkungen oder Fehlerkorrekturen für weitere Kurse auf Basis dieses Kursskripts enthalten. Dazu wählt der Dozent für die Erstellung eine spezielle Erscheinungsform für Annotationen aus. Ansonsten entspricht der Vorgang des Erstellens dem unter Kapitel 5.3.1. beschriebenen. Der Vorteil dieses Vorgehens wird erst deutlich, wenn der Dozent später während der Reflexion seines Kurses, die Möglichkeit nutzt, nach Annotationen zu suchen. Dazu öffnet der Dozent die *Annotation Browser*-Komponente und sucht komfortabel nach allen Bemerkungen oder Fehlerkorrekturen, wie schon unter Kapitel 5.3.3. beschrieben. Als Einschränkungsmerkmal gibt er dafür den entsprechenden Annotationstypen an.

Zum Anderen hat der Dozent in der Annotationsarchitektur die Möglichkeit, die Annotationen anderer Benutzer auszuwerten und daraus Feedback zu seinem Kurs zu generieren. Da es sich dabei um statistisch verdichtete Daten handelt, wird die Privatheit der einzelnen Annotation gewahrt. Konkrete Beispiele für solches Feedback werden in Abbildung 36 gegeben.

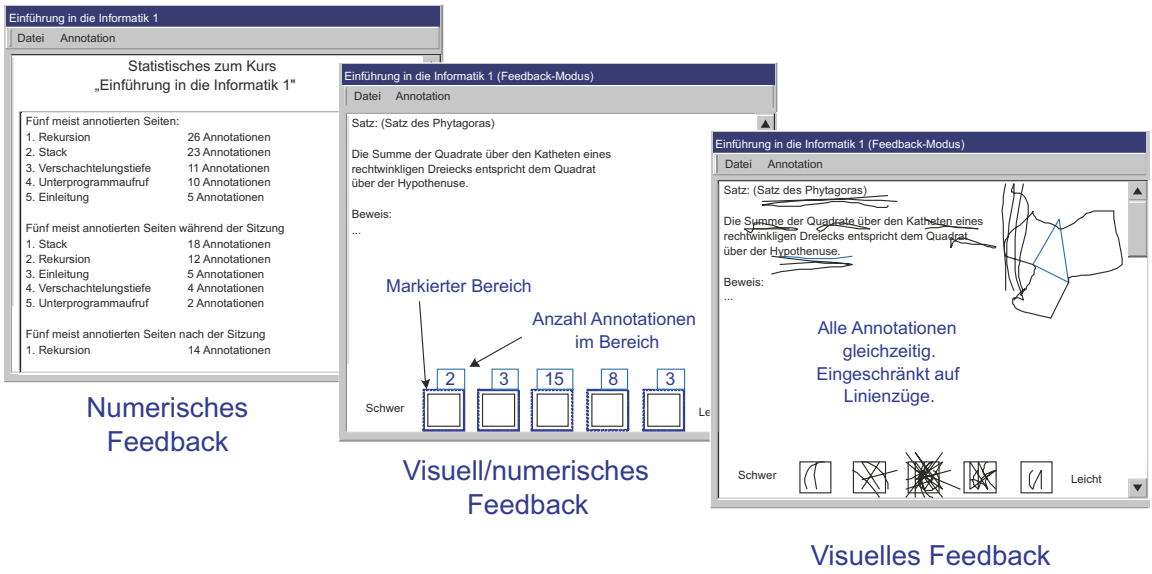


Abb. 36: Beispiele für Feedback

Diese Art des Feedbacks ist, vom Betrachten von Annotationen einzelner Studenten durch den Dozenten abzugrenzen. Ein direktes Einsehen der Annotationen Einzelner ist nur möglich, wenn dies von den Studenten über ihre Erlaubt-Liste für den Dozenten freigegeben ist. Im Gegensatz dazu werden die Gesucht- und Erlaubt-Listen der einzelnen Benutzer bei den statistischen Auswertungen nicht berücksichtigt.

Ausgangspunkt für die Funktionalität ist wieder die *ControlCenter*-Komponente. Sie stellt drei Klassen von statistischen Funktionalitäten zur Verfügung:

- Numerisches Feedback

  Der Dozent hat die Möglichkeit, Aggregatsfunktionen, wie etwa Anzahl, Summe, Maximum, Minimum, Durchschnitt auf Felder der Annotationsbeschreibungen anzuwenden. Dabei kann er die Menge, der jeweils zu berücksichtigenden Annotationen, über die Angabe von Filterkriterien einschränken. Außerdem sind Kombinationen von Abfragen über mathematische Operationen möglich. Als Ergebnis erhält der Dozent Attribut-Wert Paare die er bei Bedarf zu Diagrammen weiterverarbeiten kann. Ein Attribut entspricht dabei einem Anfrageaspekt. Der Wert ist numerisch und gibt in der Regel eine Anzahl oder einen Prozentsatz wieder.

- Visuelles Feedback

  Der Dozent bekommt alle Annotationen aller Benutzer zur aktuellen Kursskriptseite gleichzeitig angezeigt. Wahlweise wird der Erstellungszeitpunkt mit in die Darstellung, zum Beispiel über die Farbgebung oder eine zusätzliche Zeitlinie, einbezogen.

  Diese Form der Darstellung liefert visuelles Feedback. Charakteristisch für visuelles Feedback ist, dass es vom menschlichen Gehirn extrem schnell erfasst und verarbeitet wird (vgl. U.a. Weidenmann, 1994). Eine Interpretation der Darstellung bleibt vorerst Sache des Dozenten. Da diese Form der Darstellung schon aufgrund der fehlenden Annotationsinformation bisher nicht möglich war, gibt es noch keine allgemein anerkannten, klassifizierbaren Annotationsschemata mit entsprechender Interpretation. Dies ist ein identifizierbares Forschungsfeld für Pädagogen und Psychologen.

- Visuell/numerisch gemischtes Feedback

  Der Dozent gibt Bereiche auf der aktuellen Kursskriptseite an. Das Annotationskontrollzentrum liefert zu jedem Bereich die Anzahl der Benutzer, die innerhalb des Bereichs eine Annotation gemacht haben.

  Über diese Funktionalität kann der Dozent Umfragen durchführen. Wegen der einfachen und intuitiven Umsetzung dieser Funktionalität, ist es denkbar, jede Skriptseite um eine eigene Umfrage, wie etwa: „Unterrichtsstoff Stoff war: schwer, mittel, leicht.“, zu ergänzen. So können die Studenten laufend und gut zuordenbar Feedback geben.

  Neben der Angabe von Bereichen, können zusätzlich auch andere Attributwerte eingeschränkt werden. So ist der Zeitpunkt der Annotation sicherlich eine wertvolle Ergänzung. Die vollständige Definition der Abfrage wird gespeichert werden und steht somit wiederholt zur Verfügung.

Technisch werden alle drei Klassen von Feedback über die *Control Center*-Komponente auf gleiche Weise gelöst: Generierte numerische Werte werden als temporäre Annotationen verpackt und können somit entweder direkt in der Kursskriptdarstellung (vgl. Visuell/numerisch gemischtes Feedback) oder im Annotationsbrowser (vgl. Numerisches Feedback) ohne aufwendige zusätzliche Programmierung dargestellt werden. Als Quelle für die temporären Annotationen dient nicht wie beim Nachbereiten des Stoffes der lokale Annotationsspeicher, sondern der Speicher der Serverkomponente. Da dieser schon die Benutzerinformation verwaltet, ist eine Einschränkung, des zu statistischen Auswertungen autorisierten Personenkreises, problemlos realisierbar.

Insgesamt werden die angesprochenen Mechanismen zur Reflexion also durch eine vorbildliche Wiederverwendung bestehender Komponenten realisiert. Durch die überlegte Gesamtarchitektur ist dies in natürlicher Weise möglich.

5.4. Wesentliche Unterschiede zu anderen Systemen

Folgende Punkte fassen wichtige Eigenheiten der vorgestellten Annotationsarchitektur zusammen, wobei schon jeder für sich alleine, in dieser Ausprägung in keinem anderen betrachteten System Eingang gefunden haben:

- Einfache Erweiterbarkeit um neue Annotationsformen

  Da die vorliegende Architektur für sich in Anspruch nimmt, universell im Umfeld elektronisch unterstützter Lehre einsetzbar zu sein, kann es nicht statthaft sein, eine abgeschlossene Menge von Annotationsformen vorzugeben. Deshalb wurde bei der Entwicklung der Architektur explizit Wert darauf gelegt, die Architektur für Erweiterungen der Annotationsformen offen zu halten. So wird das Angebot an verfügbaren Annotationsformen aus der Registry gelesen. Entsprechende Factoryklassen sorgen für die Instanziierung der umsetzenden Klassen.

- Viele Erweiterungen auch ohne Neuübersetzung des Gesamtsystems möglich

  Zum Einen ist es für die Akzeptanz eines elektronischen Systems zur Unterstützung der Lehre allgemein notwendig, dass der Dozent einmal beim Einrichten des Systems Anpassungen an sein eigenes Verhalten vornehmen kann. Spätestens für die Studenten stellen solche Vorgaben oft eine starre Einschränkung dar. Wird das System von einem Administrator eingerichtet, ist auch der Dozent schon eingeschränkt.

Durch konsequenten Einsatz von Factory-Klassen und zur Laufzeit gelesenen Einträgen in einer lokalen Registry ist es bei Umsetzung mit heutigen Mechanismen (vgl. Z.B. dynamic link libraries (DLL), ClassLoader, ...) in diesem Architekturvorschlag technisch möglich, dass jeder Benutzer eigene Erweiterungen programmiert und ohne Neuübersetzung des Gesamtsystems einbindet. Die bei erforderlicher Neuübersetzung des Gesamtsystems notwendige Freigabe von Sourcecode und lizenzrechtliche Überlegungen gefährden die Einbindung solcher Funktionalität schon auf Anbieterseite.

Damit diese Erweiterungsmöglichkeiten nicht nur technisch vorhanden, sondern auch für Anwender praktizierbar sind, wurde auf starke Kapselung geachtet. Dadurch wird erreicht, dass für solche Erweiterungen kein, eventuell überforderndes, Gesamtverständnis über die Gesamtarchitektur notwendig ist. Es müssen lediglich vergleichsweise begrenzte Anforderungen erfüllt werden. Über objektorientierte Vererbung ist auch diese Hürde noch einmal entschärft. Trotzdem werden sicherlich nicht alle Benutzer eigene Erweiterungen schreiben. Andererseits ist an der Vielzahl kleiner Insellösungen, zum Beispiel im Bereich Physik, ein Bedarf für die Anpassung an das Lehrverhalten abzulesen. Diese Architektur ermöglicht einen Einstieg in solche Anpassungen auf einer höheren Schicht. Die gegebene Infrastruktur übernimmt viele, der oft abschreckenden Standardarbeiten.

- Konsequente Trennung zwischen der Erstellung und der Darstellung einer Annotation

  Die anderswo kaum anzutreffende Trennung zwischen Erstellung und Darstellung von Annotationen wurde im vorliegenden Architekturvorschlag als ein zentrales Ziel definiert. Für diese Trennung sprechen zwei fundamentale Punkte: Erstens erfordern manche multimedialen Annotationen schon vom Wesen her unterschiedliche Schnittstellen für das Erstellen und für das Darstellen (vgl. Audio). Zweitens ermöglicht gerade diese Aufteilung eine ergonomische Umsetzung jeder dieser Phasen. Während bei der Erstellung verschiedenste unterstützende Hilfsmittel vorstellbar und für qualitative Annotationen geradezu notwendig sind (vgl. Raster, Hilfslinien, Ankerpunkte), würden diese eine Darstellung nur unnötig aufblähen und der Effizienz des Gesamtsystems schaden.

- Mehrere Annotationsinhalte für die gleiche Annotation möglich

  Die vorgestellte Architektur sieht vor, dass eine Annotation über verschiedene Inhalte repräsentiert werden kann. Dies öffnet das System für Drittanwendungen in folgender Weise: Werkzeuge Dritter, die eine Repräsentation einer Annotation in eine andere Repräsentation überführen, haben so die Möglichkeit, über genau eine Stelle an den Annotationsinhalt heranzukommen und das Ergebnis der Umwandlung wieder zur Weiterverarbeitung im System abzulegen. Da nur Annotationsinhalte ergänzt werden, bleibt die Zuordnung zur ursprünglichen Annotation erhalten.

  Ein wünschenswertes Werkzeug Dritter ist zum Beispiel Texterkennung. Sie erstellt zu einem handschriftlichen Annotationsinhalt einen textuellen. Weil das Ergebnis der Texterkennung wieder als Annotationsinhalt abgelegt ist, steht es sofort den Standardmechanismen der Architektur, wie etwa Suche, zur weiteren Verarbeitung zur Verfügung.

- Unterschiedliche Darstellungen für Annotationen möglich

  Im Gesamtsystem ist die Darstellung für eine Annotation nirgends direkt festgelegt. Somit kann ein und dieselbe Annotation bei verschiedenen Benutzern unterschiedlich angezeigt

werden. Von der Architektur her ist es also problemlos möglich, für geeignete Annotationstypen auch Darstellungen für behinderte Benutzer anzubieten.

- Schrittweise Umsetzung möglich

  Interessant am vorliegenden Architekturvorschlag ist auch, dass er sich in mehreren Phasen schrittweise umsetzen lässt (vgl. Abb. 37). Für eine erste Umsetzung der Annotationen reicht die Implementierung der Schnittstellen der Kursskriptkomponente, die *Design*-Komponente, die *Presentation*-Komponente und die *Local Storage*-Komponente. In dieser Ausbaustufe ist es dann möglich, Annotationen zu erstellen, zu speichern und anzuzeigen. Dies entspricht dem Stand der Annotationsunterstützung der meisten Systeme zur elektronischen Unterstützung der Lehre, falls überhaupt eine Unterstützung angeboten wird..

  Wird nun in einer weiteren Ausbaustufe die *Annotation Browser*-Komponente ergänzt, stehen automatische Zusammenfassungen und Suche zur Verfügung. Bis zu dieser Ausbaustufe ist das Arbeiten mit Annotationen nur auf eigene Annotationen beschränkt.

  Werden schließlich in der dritten Phase alle restlichen Komponenten umgesetzt, ist ein Austausch von Annotationen unter den Benutzern möglich.

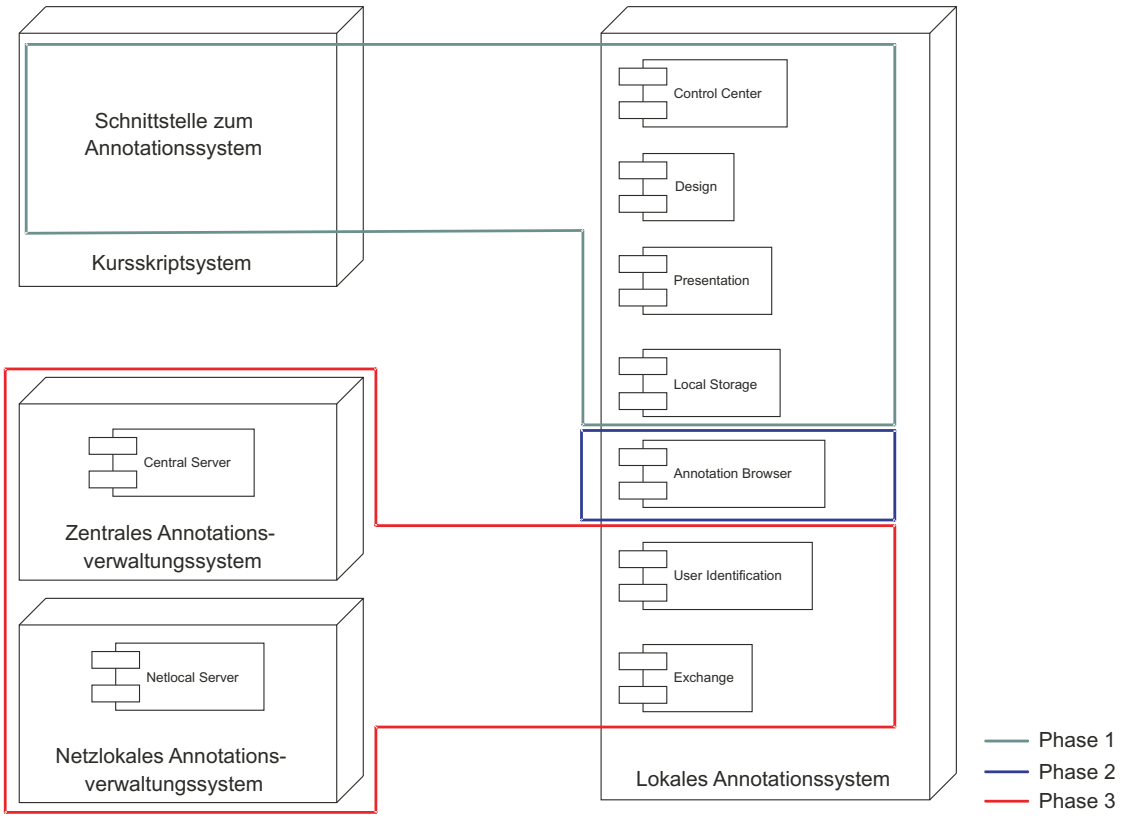


Abb. 37: Schrittweise Umsetzung

Ein weiterer Vorteil der vorgeschlagenen Architektur ist sicher auch, dass sie schon mehrere Iterationen des Entwicklungsprozess durchlaufen hat. Viele der Anforderungen wurden während des praktischen Einsatzes am Informatiklehrstuhl für Angewandte Informatik und Kooperative Systeme der Technischen Universität München identifiziert und über prototypische Umsetzungen studiert. Auf diese Erfahrungen wird im nächsten Kapitel eingegangen.

6. Konkrete Anwendung der Annotationsarchitektur

Als Beleg für die Tragfähigkeit der vorgestellten Architektur, werden nun zwei darauf basierende Umsetzungen vorgestellt.

6.1. Das CAL-System

6.1.1. Überblick

CAL steht in diesem Zusammenhang für „Call A Lecture“ und bezeichnet die am Lehrstuhl für Angewandte Informatik und Kooperative Systeme der Fakultät für Informatik der Technischen Universität München entwickelte Umgebung zur elektronischen Annotation von Lehrangeboten (vgl. Schütz, 2001). CAL ist aus dem Projekt Lecture 2000 (vgl. Schlichter, 2000) hervorgegangen und dient seit mehreren Jahren unter anderem als Experimentierplattform für eine elektronische Annotationsunterstützung. Einige frühe Bereiche des CAL-Systems basieren auf einem Redesign der Arbeit von Daniel Ovadya (1999).

In weiten Teilen entspricht CAL der vorgestellten Annotationsarchitektur und hat diese auch entscheidend mitgeprägt. Eine direkte Referenzumsetzung der Annotationsarchitektur wird im späteren Teilkapitel zu konkreten Anwendungen vorgestellt.

Der erwähnte Lehrstuhl nutzt TargeTeam (Teege, 2002), um aus einzelnen Lernobjekten einen Kurs zusammenzustellen und dafür automatisiert verschiedene Kursskripte zu generieren. Das CAL-System wurde entwickelt, um basierend auf diesen Kursskripten vielfältige Annotationsformen anzubieten. So erlaubt das CAL-System beispielsweise eine Ergänzung des Kursskripts um Audio- und Videoannotationen. Außerdem sind Freihandannotationen, Annotationen mit vordefinierten Symbolen und Textannotationen möglich. Letztere Annotationsmöglichkeiten stehen den Dozenten und den Studenten zur Verfügung. Die Audio- und Videoannotationen können momentan nur vom Dozenten erstellt werden.

Die vom CAL-System zur Verfügung gestellten Annotationsmöglichkeiten werden am oben genannten Lehrstuhl in nicht auf den ersten Blick offensichtlicher Art und Weise genutzt: Das Annotationssystem dient der Aufzeichnung von Audio-, Video- und Notizannotationen während Präsenzvorlesungen. Dabei ergibt sich durch den Annotationsmechanismus eine intensive Verbindung der entstehenden Annotationen zum Kursskript. Ein Effekt, welcher das System gerade für Dozenten attraktiv macht ist, dass diese „Konservierung einer Unterrichtseinheit“ von einer Person alleine und ohne Nachbereitung machbar ist, wenn dafür geringe Abstriche in der redaktionellen Qualität des Videos (nur eine Kameraeinstellung) hingenommen werden (vgl. Schütz, 2003). Die so „on the fly“ erstellte elektronische Variante des Kurses wird den Studenten zusätzlich zum Selbststudium angeboten.

Ungeachtet der Nutzung des Dozenten, hat jeder Student über das CAL-System die Möglichkeit, eigene private Annotationen im Kursskript zu erstellen. Ein Austausch der Studentenannotationen untereinander ist noch nicht integriert.

Voraussetzung für die Annotationsunterstützung durch das CAL-System ist ein aus HTML-Dateien bestehendendes Kursskript. Sollten innerhalb des Kursskripts HTML Fra-

mes benutzt werden, ist darauf zu achten, dass keine absoluten Frameangaben wie „_top“ genutzt werden. Weitere Voraussetzungen gibt es keine.

6.1.2. Vorgaben bei der Entwicklung

Zu Beginn der Erstellung des CAL-Systems wurde festgelegt, dass die erste Ausbaustufe der Annotationsunterstützung (vgl. Kap. 5.4.) erreicht werden muss. Die Annotationen müssen also erstellt, gespeichert und wieder dargestellt werden können. Insbesondere ergeben sich daraus folgende Anforderungen:

- **CAL1:** Da das Präsentationskursskript des Dozenten als HTML-Seiten vorliegt, müssen alle Erweiterungen, insbesondere auch das Erstellen der Annotationen in einer Browserumgebung stattfinden. Eine Festlegung auf einen speziellen Browser, ist auf konzeptueller Ebene in jedem Fall zu vermeiden. Auf Implementierungsebene notwendige Spezialisierungen, sind in eigenen Klassen zu kapseln und konsequent auswechselbar zu gestalten.
- **CAL2:** Alle Kursartefakte - sprich Kursskript, Audio-, Video- und Dozentenannotationen - sind so abzulegen und in sich zu verweben, dass ein direktes, komplettes Kopieren auf andere Medien, insbesondere CD oder DVD ohne Nacharbeiten möglich ist.
- **CAL3:** Da das System auch von Studenten genutzt werden soll, darf es keine speziellen Hardware- und Systemanforderungen für das Erstellen der Annotationen (ausgenommen Audio- und Videoannotationen) geben. Eine heute als vorhanden postulierbare Maus und Tastatur müssen prinzipiell ausreichend sein. Andere Eingabemedien, wie zum Beispiel ein Stifteingabemedium, sollen ausdrücklich zusätzlich genutzt werden können.
- **CAL4:** Jeglicher administrativer Zusatzaufwand für den Dozenten, ganz besonders während der Sitzung, ist in jedem Fall zu minimieren. Eine Nachbearbeitung darf nicht grundsätzlich notwendig sein.

Die Anforderung *CAL1* ergab, dass die Umsetzung als Java-Applet durchgeführt wurde. Java-Script als Sprache innerhalb der meisten Browser bietet keine Unterstützung von Zeichenroutinen, Audio oder Video und schied deshalb aus. Die Marktdurchdringung von Macromedia Flash war zu Beginn der Umsetzung (Mitte 2000) noch zu gering beziehungsweise der Funktionsumfang noch nicht ausreichend. Die Ausführung von Java-Applets wurde von allen marktüblichen Browsern unterstützt.

Des Weiteren ergeben die Anforderungen *CAL1* und *CAL2*, dass eine *dateibasierte* Persistenzkomponente für die Annotationen notwendig ist. Die eventuell denkbare Vorbedingung an die Benutzer, ein SQL Datenbanksystem zu installieren, hätte Anforderung *CAL3* klar widersprochen.

Aus der Anforderung *CAL3* erwuchs die Entscheidung, für die Annotationserstellung, die von Java standardmäßig unterstützten Keyboard- und Mausevents zu nutzen. Eine Programmierung einer speziellen Schnittstelle für Stifteingabegeräte war damit nicht notwendig, da diese ebenfalls Mausevents erzeugen.

Aus Anforderung *CAL4* wurde abgeleitet, dass die Benutzungsoberfläche für Annotationen mit einer minimalen Anzahl von Mausklicks bedienbar sein muss. Verschachtelte Menüstrukturen wurden deshalb von vorne herein vermieden.

Damit stellt sich die Frage des Integrationsgrades des Annotationssystems in das Kursskriptsystem (vgl. Kap. 5.1.4.2.). Auf Ebene der Steuerungselemente wurde ein parallel, ge-

trennter Ansatz gewählt, der später der besseren Bedienbarkeit wegen, um einige zusätzliche untergeordnet, integrierte Steuerelemente ergänzt wurde. Abbildung 39 zeigt die Steuerungsoberfläche des Annotationssystems für den Dozenten.

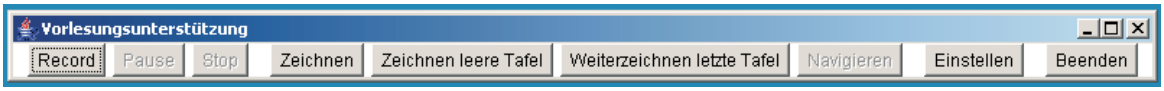


Abb. 39: Die Steuerungsoberfläche des CAL-Annotationsystems als Dozenten

Auf Ebene der Darstellungsintegration wurde eine vertikal geschichtete Integration durchgeführt. Eine horizontal verschränkte Integration auf HTML-Ebene würde eine Veränderung der HTML-Seiten des Kursskripts notwendig machen. Während sich dies bei Einsatz eines Webservers recht elegant transparent erledigen lässt, bedingt dies der geforderten reinen dateibasierten Ablage (vgl. Folgerung aus *CAL2* und *CAL3*) wegen eine echte Veränderung der HTML- Kursskriptdateien. Dies wurde, der entstehenden engen Verbindung zwischen Kursskript und Annotation wegen, vermieden.

6.1.3. Das erstellte System

6.1.3.1. Die CAL Applets

Die Integration des CAL-Systems in das HTML-Kursskript erfolgt auf Ebene von HTML-Frames, unterstützt durch JavaScript (vgl. Abb. 38).

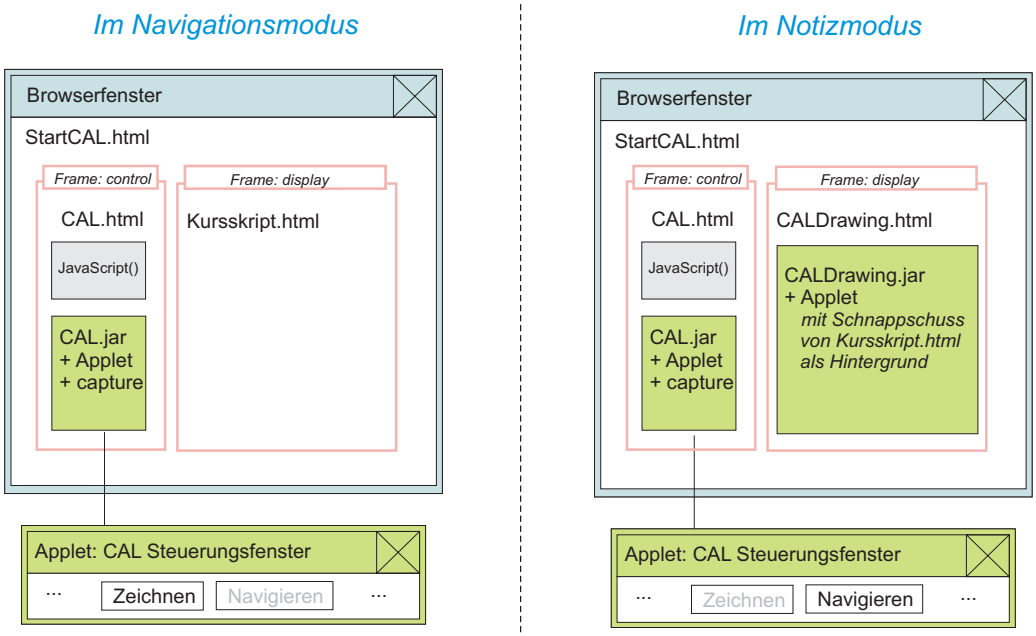


Abb. 38: Integration CAL - Kursskript

Beim genaueren Betrachten der Skizze wird offensichtlich, dass das Kursskript konzeptuell gesehen, innerhalb einer vom CAL-System zur Verfügung gestellten Laufzeitumgebung an-

geboten wird. Der Frame *control* hat eine Breite von 0. Damit sieht der Benutzer im Browserfenster nichts von diesem Frame. Der Frame *display* mit dem Kursskript nimmt den ganzen Browser ein. Das Applet im Frame *control* kann über Aufrufe von JavaScript-Funktionen in der Datei CAL.html Einfluß auf den Frame *display* nehmen. Über solche Funktionsaufrufe ermittelt das Applet unter anderem in regelmäßigen Abständen den Namen der HTML-Seite im Frame *display*. Damit erkennt das Applet jede Navigation im Kursskript und protokolliert diese. Außerdem hat das Applet darüber jederzeit die Möglichkeit, eine andere HTML-Seite in den Frame *display* zu laden.

Wählt der Dozent den Punkt „Zeichnen“ im in Abb. 39 dargestellten Steuerungsfenster zum Starten einer Notiz, so erstellt das Applet im Frame *control* einen bildlichen Schnappschuss des aktuell im Browser dargestellten Kursskripts. Da von einem Java-Applet heraus kein Bildschirmbereich außerhalb eigener Fenster gelesen werden kann, wird vom Applet zum Erstellen des Schnappschusses, ein im *CAL.jar* Archiv abgelegtes, ausführbares C-Programm *capture* gestartet. *capture* erstellt eine Bilddatei, die der Anzeige des Frames *display*, also der aktuellen Darstellung des Kursskripts entspricht. Anschließend ersetzt das Applet in *CAL.jar* unter Zuhilfenahme einer JavaScript-Funktion in *CAL.html* das Kursskript im Frame *display* durch die Datei *CALDrawing.html*. *CALDrawing.html* beinhaltet ein weiteres Applet, welches den gesamten ursprünglichen Anzeigebereich des Kursskripts belegt und das gespeicherte Bild als Hintergrund anzeigt (vgl. Abb. 38). Dieses Applet erlaubt dem Dozenten nun das Erstellen von Notizen auf dem Hintergrundbild. Dabei stehen dem Dozenten verschiedene Werkzeuge zur Verfügung. Der Dozent wählt ein Werkzeug aus zwei Iconleisten aus. In der ersten Iconleiste sind oft benötigte Werkzeuge und Verweise auf weitere Iconleisten mit weniger oft benötigten Werkzeugen oder einer Farbauswahlen enthalten. Wird ein Verweis auf eine der weiteren Iconleisten ausgewählt, erscheint diese als zweite Iconleiste (vgl. Abb. 40). Die erste Iconleiste bleibt also immer erhalten, wohingegen die zweite je nach Bedarf ausgetauscht wird. Damit konnte erreicht werden, dass mit maximal zwei Klicks jedes Werkzeug oder jede Einstellung erreichbar ist und trotzdem nur wenig Anzeigefläche des Kursskripts verloren geht.

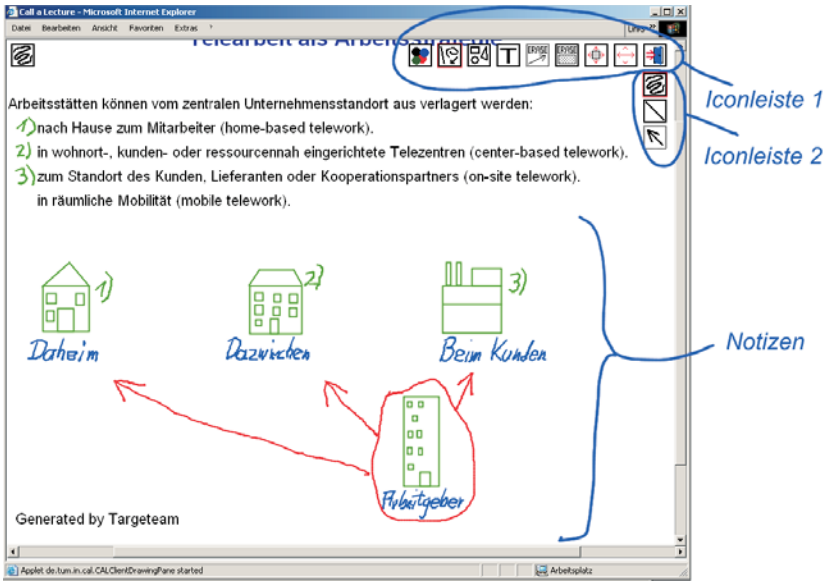


Abb. 40: Applet zum Erstellen von Notizen, mit Werkzeugauswahl

6.1.3.2. Der CAL-Server

Dem Dozenten steht im CAL-System die Möglichkeit offen, Audio und Videoannotationen zu erstellen. Die eigentliche Aufzeichnung wird dabei von einer eigenen Komponente, dem so genannten CAL-Server durchgeführt. Das CAL-Applet kommuniziert mit dem CAL-Server über eine Socketverbindung und löst dabei zum Beispiel das Starten beziehungsweise Stoppen einer Aufnahme aus.

Das Abspalten der audiovisuellen Aufzeichnungsfunktionalität hat zwei praktische Gründe: Zum Einen sind die technischen Möglichkeiten innerhalb eines Java-Applets gegenüber einer allein stehenden Java-Applikation eingeschränkt. Zum Anderen ist der mit der Audio/Videoausrüstung ausgestattete Rechner meist nicht der Rechner, mit dem der Dozent präsentiert - auf dem aber das CAL-Applet läuft. Konkret kam das CAL-System an der Technischen Universität München meist in folgendem Szenario in den Einsatz: Der Dozent präsentiert seinen Kurs über einen Browser auf seinem Laptop vom Rednerpult aus. Dort läuft somit das CAL-Applet. Im Hörsaal fest installierte Kameras liefern ein Videosignal in einen Regieraum. Der Ton des tragbaren, drahtlosen Dozentenmikrofons läuft ebenfalls im Regieraum auf. Im Regieraum ist ein Rechner mit entsprechender Ausstattung fest installiert, der die Audio und Videosignale empfängt. Auf diesem Rechner läuft der CAL-Server.
(Vgl. hierzu zum Beispiel Teege, 2000, Schlichter, 2003 und Schlichter, 2005)

Zu erwähnen bleibt, dass der CAL-Server prinzipiell auch auf dem Dozentenrechner parallel zum CAL-Applet gestartet werden kann. Dies wurde zum Beispiel explizit bei der Aufzeichnung der Vorlesung „Einführung in die Informatik 3“ im WS 2001/02 mit 800 Studenten gemacht (vgl. Schlichter, 2001), da der, bei dieser Studentenzahl notwendige Hörsaal, damals noch nicht über eine entsprechende Videoausstattung verfügte. Für das Videosignal wurde eine einfache WebCam direkt über USB an das Dozentennotebook angeschlossen.

Der CAL-Server erlaubt zur Laufzeit, über die Angabe eines Klassennamens, die Konfiguration, welches Java-Objekt die technische Aufzeichnung von Audio und Video durchführt. Momentan stehen hier zwei Alternativen zur Verfügung: Ein Recorderobjekt, welches Mediastreams im Real Format erzeugt (vgl. RealNetworks, 2005) und eines zur Erzeugung von QuickTime Mediastreams (vgl. Apple, 2005). Beide Formate sind Streamingformate. Die entstehenden Medien lassen sich also über entsprechende Streamingserver im WorldWide-Web anbieten. Ein komplettes Herunterladen der Datei vor dem Betrachten ist bei Streaming nicht notwendig. Hier wird nur der gerade betrachtete Teil über das Netzwerk abgerufen. Liegt die Mediastreamdatei auf dem lokalen Rechner, kann sie auch direkt abgespielt werden. Ein Streamingserver ist dann nicht notwendig. Dies ist insbesondere zur Erfüllung von CAL2 erforderlich.

Der Benutzer kann den CAL-Server so konfigurieren, dass neben Audio und Video zusätzlich jede Navigation im Kursskript aufgezeichnet und unter anderem direkt in die Mediastream-Datei integriert wird. Wurde ein Annotationsmediastream mit dieser Option erstellt, öffnet der, für das entsprechende Format notwendige Abspieler beim Betrachten des Mediastreams automatisch ein zusätzliches Browserfenster und führt dort die aufgezeichneten Navigationsschritte synchron zum Mediastream durch. Konzeptuell gesehen stellt die Videoannotation dann eigenständig sicher, dass der zugehörige annotierte Inhalt des Kursskripts präsentiert wird.

Neben der Navigation des Dozenten im Kursskript, ist es auch möglich, die Erstellung von Notizen des Dozenten mit in den Mediastream abzulegen. Technisch wird dazu jede Notiz

vom CAL-Applet an den CAL-Server weitergegeben und dort in eine - bei Bedarf mehrere- URL's vektororientiert kodiert. Diese URL's werden dem jeweiligen Recorder-Objekt als reguläre URL zum Einarbeiten in den Mediastream übergeben. Wichtig ist zusätzlich die Feststellung, dass vom CAL-System auch das Laden der *CALDrawing.html* Seite zum Starten des Notierens als Navigation im Kursskript aufgezeichnet wird. Der Moduswechsel wird also beim Abspielen auch vollzogen.

Die URL's zum Kodieren einer Notiz haben einen besonderen Aufbau:

`&&<control-Frame>&&javascript:drawEvent('<Notizkodierung>')`

Über diese URL wird der MediaStream Abspieler angewiesen, im HTML-Frame *<control-Frame>* die JavaScript-Funktion *drawEvent()* mit den Parametern *<Notizkodierung>* aufzurufen (vgl. RealNetworks, 2003). Diese Funktion wird von der CAL-Umgebung zur Verfügung gestellt. Sie sorgt dafür, dass im Zeichenapplet in *CALDrawing.html* eine Methode zur Darstellung der Notiz aufgerufen wird.

Aus den bisherigen Erörterungen geht hervor, dass beim CAL-Server alle Informationen zu einem Kurs zusammenlaufen. Dies waren die Navigation im Kursskript, das Video, der Ton und die Notizen. Deshalb wird die Anforderung *CAL2* - dem Bereitstellen aller Kursartefakte zum einfachen Kopieren - über diese Komponente realisiert. Der CAL-Server legt dazu alle Informationen in die in Abb. 41 dargestellte Verzeichnisstruktur ab.

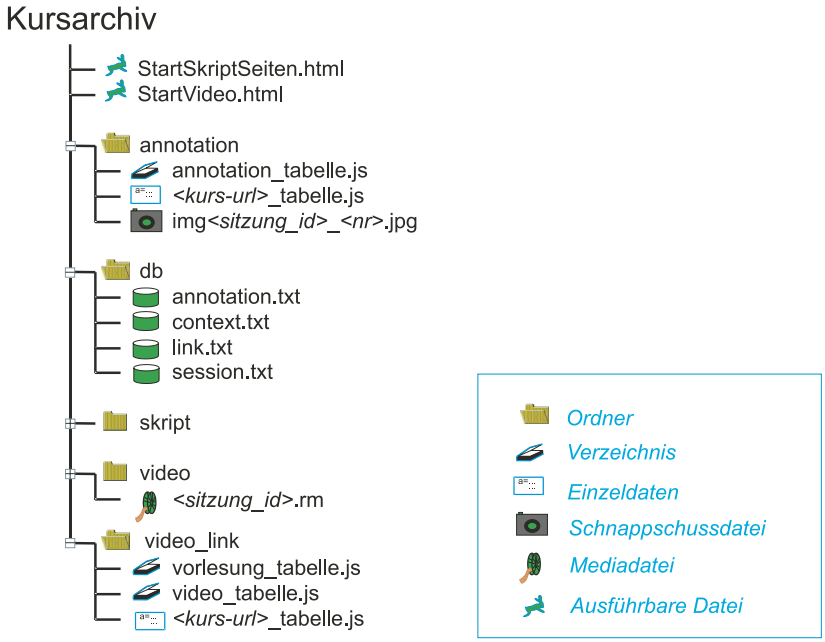


Abb. 41: Verzeichnisstruktur eines mit CAL erstellten Kursarchivs

Alle Referenzen zwischen Dateien innerhalb dieser Struktur beziehen sich auf ihre relative Position in dieser Verzeichnisstruktur. Alle Querverbindungen bleiben also intakt, wenn die ganze Verzeichnisstruktur kopiert wird.

Die Dateien im Einzelnen:

- *StartSkriptSeiten.html*

Diese Datei öffnen die Studenten in einem Browser, um die CAL Umgebung zu starten und das Kursskript darin darzustellen.

- *StartVideo.html*

  Öffnet ein Student diese Datei in seinem Browser, so startet er die CAL Umgebung und bekommt vom System eine Liste der verfügbaren MediaStreams zur Auswahl angeboten.

- *annotation_tabelle.js* und *video_tabelle.js*

  Diese beiden Dateien beinhalten jeweils eine Auflistung aller Namen von Kursskriptseiten zu denen eine Notiz-Annotation oder eine Video-Annotation erstellt wurde. In der jeweiligen Datei ist die Auflistung in Form eines JavaScript-Arrays abgelegt, da sich Java-Script Dateien (Endung *.js) innnerhalb einer HTML-Seite einfach über

  ```
  <script src = "<typ>_tabelle.js" type="text\/javascript"/></script>";
  ```

  laden und einbinden lassen. Über diese Arrays ermittelt das CAL-System dann bei jedem Seitenwechsel im Kursskript, ob dazu auch eine Notiz- oder Videoannotation existiert und aktiviert darüber gegebenenfalls entsprechende Startmöglichkeiten für die Studenten.

- *vorlesung_tabelle.js*

  Bei der Datei *vorlesung_tabelle.js* handelt es sich ebenfalls um ein Verzeichnis. Hier werden alle Videoannotationen aufgelistet. Dieses Verzeichnis dient *StartVideo.html* zur Ermittlung aller verfügbaren MediaStreams.

- <Kurs_url>*_tabelle.js, img*<sitzung_id>_<nr>*.jpg*

  In den Dateien <Kurs_url>*_tabelle.js* wird jeweils Detailinformation zu den Annotationen (Video/Notiz) pro Kursskriptseite gespeichert. Die URL der Kursskriptseite <Kurs_url> dient dabei als Identifikator. Außerdem wird für Notizen der jeweilige Schnappschuss des Hintergrunds in Dateien *img*<sitzung_id>_<nr>*.jpg* abgelegt. <*sitzung_id*> steht dabei für einen Identifikator der Videoannotation und <*nr*> für eine laufende Nummer des Wechsels in den Notizmodus innerhalb einer Videoannotation.

6.1.3.3. Konkrete Klassen im CAL-System

Abbildung 42 zeigt das Klassendiagramm zum CAL Applet.

Die Klasse *CALClient* entspricht der Klasse *CcController* in der Komponente *ControlCenter* der Annotationsarchitektur und nimmt damit auch die zentrale Position in der CAL Applet Architektur ein.

Über die Klasse *CALClientJScript* kommuniziert *CALClient* mit dem Kursskriptsystem. Dazu bietet *CALClientJScript* die Methoden `getFrameURL()` und `setFrameURL()` an. Diese beiden Methoden nutzen die von gängigen Browsern zur Verfügung gestellte Schnittstelle zur Kommunikation zwischen Applets und JavaScriptfunktionen in HTML-Seiten (vgl. z.B. Leonardo, 1997). `getFrameURL()` und `setFrameURL()` rufen also ihrerseits JavaScriptfunktionen auf. Diese haben über DOM Zugriff auf die Bestandteile der angezeigten HTML-Seite.

Da die HTML-Seiten des Kursskripts im in CAL angenommenen Szenario nicht verändert werden sollen und es außer dem Browser kein Laufzeitsystem für das Kursskript gibt, kann auch im Kursskriptsystem kein Code ergänzt werden, der die Klasse *CALClient* aktiv von ei-

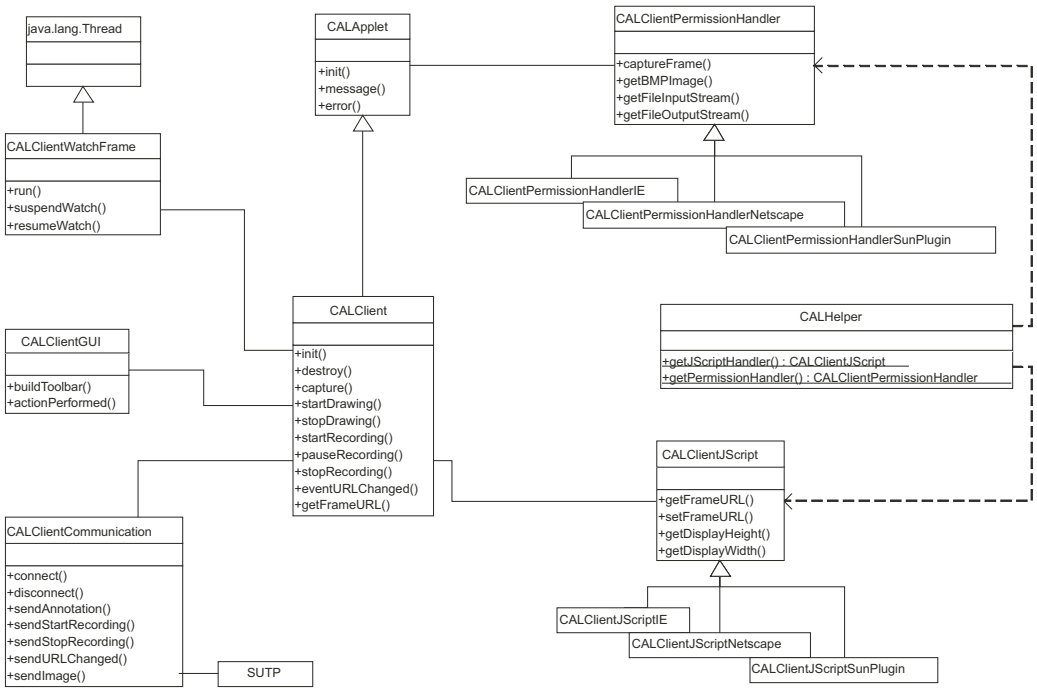


Abb. 42: Klassendiagramm zum CAL Applet

ner Navigation im Kursskript unterrichtet. Somit wird diese notwendige Funktionalität über einen zusätzlichen parallelen Thread *CALClientWatchFrame* innerhalb des CAL-Applets zur Verfügung gestellt. Der Thread fragt im halb Sekunden Takt die URL der aktuell dargestellten Kursskriptseite ab und vergleicht diese mit der letzten Anfrage. Wird ein Unterschied erkannt, signalisiert der Thread der Klasse *CALClient* über `eventURLchanged()` die Navigation des Benutzers innerhalb des Kursskripts. *CALClientJScript* und *CALClientWatchFrame* zusammen bilden somit den *ScriptStub* der vorgeschlagenen Annotationsarchitektur.

Es bleibt zu erwähnen, dass es innerhalb verschiedener Browser in JavaScript (oder ECMAScript, JScript) und dem zugrunde zulegenden Document Object Model (DOM) (W3C, 1998) entscheidende Unterschiede gibt (vgl. Cross-Browser Scripting, u.a. in McFarlane, Chiarelli, De Carli, Li, Updegrave, Wilcox, Wilton & Wootton, 1999). Um das CAL System robust gegenüber unterschiedlichen Browsern und bei Bedarf einfach anpassbar zu halten, wurde ein Erzeugungsmuster Ansatz (vgl. Gamma, Helm, Johnson & Vlissides, 1996) gewählt: An keiner Stelle in CAL wird eine Instanz von *CALClientJScript* direkt instanziiert. Ein Objekt der Klasse *CALClientJScript* wird über die statische Methode `getJScriptHandler()` der Klasse *CALHelper* aufgebaut. Diese Methode ermittelt zuerst, in welcher Browserumgebung das Applet läuft und erstellt daraufhin ein geeignetes, spezialisiertes Objekt einer Kindklasse von *CALClientJScript*.

Ein ähnliches Problem taucht in einem anderen Zusammenhang mit Browsern auf. Java-Applets sind aus Sicherheitsgründen in ihren Möglichkeiten zunächst stark eingeschränkt (vgl. „sandbox“ u.a. in McGraw & Felten, 1999). So dürfen Applets beispielsweise keine Verbindung zu anderen Rechnern, als dem Server vom dem sie geladen werden, aufbauen, lokal keine Dateien schreiben und auch lokal keine anderen Programme starten. Je nach Browser gibt es verschiedene Möglichkeiten diese Einschränkungen aufzuheben. Prinzipiell ist aber für jede Aufweichung der Einschränkungen das Einverständnis des Benutzers nötig. Außerdem

sind bei manchen Browsern Änderungen im Programm des Applets notwendig (vgl. Middendorf, Singer & Heid, 2002: Kap. 15.1.4.). Um das CAL System auch hier anpassbar und robust zu halten wird wieder ein Erzeugungsmuster Ansatz genutzt. Als Erzeuger dient abermals die Klasse *CALHelper* mit einer statischen Methode `getPermissionHandler()`. Wird an einer Stelle im CAL Applet Code eine Funktion benötigt, die der Einschränkungen in Applets wegen normalerweise nicht ausgeführt werden kann, so wird über die Erzeugerfunktion `getPermissionHandler()` eine Instanz der Klasse *CALClientPermissionHandler* aufgebaut und der Funktionsaufruf an dieses Objekt weitergegeben. Zur Erzeugung der geeigneten Klasse prüft die Erzeugerfunktion, in welcher Browserumgebung das Applet läuft und wählt daraufhin eine passende Kindklasse von *CALClientPermissionHandler* zur Erzeugung aus. Dieses Vorgehen hat den Vorteil, dass alle eingeschränkten Funktionsaufrufe in einer Klasse gekapselt werden und bei Änderungen des Browsers nur hier etwas angepasst werden muss.

Die Problematik mit den verschiedenen Browsern hat sich in letzter Zeit bis zu einem gewissen Grad durch das für alle gängigen Browser zur Verfügung stehen eines Java-Plugins direkt von SUN entschärft. Der Browser führt jetzt das Java Applet nicht mehr selbst mit proprietärem Code aus, sondern lädt das Java-Plugin von SUN und überlässt diesem die Ausführung. Dies trägt sehr zur Vereinheitlichung bei.

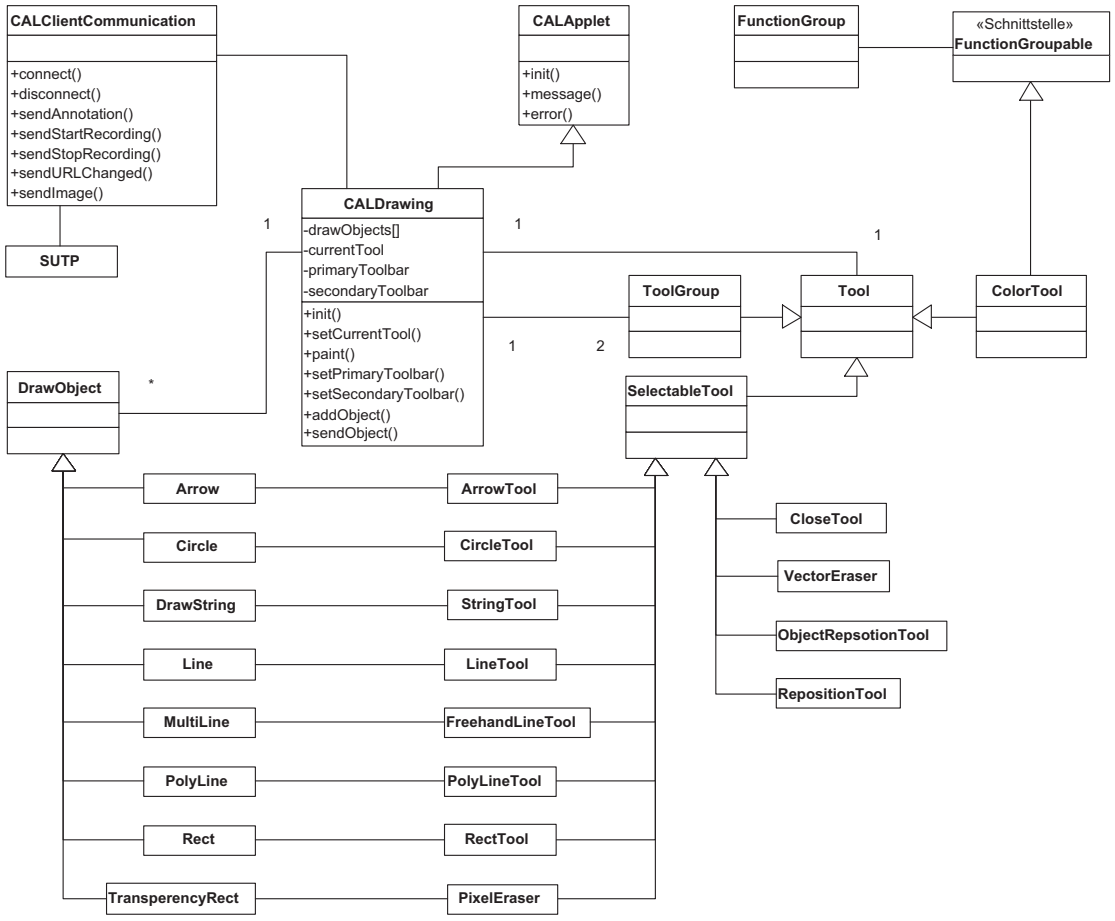


Abb. 43: Klassendiagramm zur Komponente CALDrawing zur Erstellung von Notizen

Die Trennung in eine *Design* und eine *Presentation*-Komponente wie in der Annotationsarchitektur gefordert, ist auch im CAL-System vollzogen, wenngleich die Trennung nicht so offensichtlich ist. Grund hierfür ist der unterschiedliche Ort für die Annotationserstellung von Videoannotationen und Notizen. Notizen werden komplett im CALDrawing-Applet abgehandelt, wohingegen die Videoannotation schlussendlich im CAL-Server erstellt wird.

Beginnt der Benutzer über den Menüpunkt „Zeichnen“ in der Steuerungsoberfläche eine Notiz, so lädt *CALClient* über `setFrameURL()` die HTML-Seite *CALDrawing.html* anstelle des Kursskripts in den Frame *display*. Damit wird das Applet *CALDrawing* gestartet. Abbildung 43 zeigt das Klassendiagramm dieser Komponente zur Erstellung von Notizen.

Die Klasse *CALDrawing* übernimmt die Rolle der Klasse *DsController* der *Design* Komponente in der erarbeiteten Annotationsarchitektur. In der Methode `init()` werden die zur Verfügung stehenden Annotationsformen jeweils repräsentiert durch ein Objekt der Klasse *Tool* in verschiedenen *ToolGroup*-Objekten verankert. Dabei wird eine *ToolGroup* zur so genannten primären ToolGroup. Beachtet man die Vererbungshierarchie von *Tool* und *ToolGroup* wird deutlich, dass neben *Too*l-Objekten auch *ToolGroup*-Objekte in einer *ToolGroup* eingehängt werden können. So werden alle weiteren *ToolGroup*-Objekte unter die primäre eingetragen. *CALDrawing* präsentiert die *ToolGroup*-Objekte dem Benutzer dann als Iconleisten zur Auswahl, wie zuvor in Kap. 6.1.3.1. beschrieben.

Die *Tool*-Klassen des CAL-Systems entsprechen der Klasse *DsDesigner* im Architekturvorschlag, wobei auf die dynamische Erzeugung verzichtet wurde. Steht eine *Tool*-Klasse für ein spezielles Annotationswerkzeug, wie zum Beispiel Linie, so gibt es dazu eine entsprechende *DrawObject*-Klasse. *DrawObject* entspricht somit einem *DsTool* in der Annotationsarchitektur. Auch hier wurde auf eine dynamische Zuordnung verzichtet.

CALDrawing nutzt wie *CALApplet* eine Instanz der Klasse *CALCommunication,* um mit dem *CALServer* zu kommunizieren. *CALDrawing* übermittelt dem *CALServer* eine vektororientierte Beschreibung einer jeden Notiz. Außerdem wird beim Starten des Notizmodus das benutzte Hintergrundbild mit dem Schnappschuss des Kursskripts zur Archivierung übertragen. Somit stehen im CALServer auch alle Informationen zu einer Notiz zur Verfügung.

Neben Notizen als Annotation erlaubt das CAL-System auch das Erstellen von Audio- und Videoannotationen. Die hierfür notwendige Funktionalität liegt wie oben erörtert konsequenterweise beim *CALServer*. Die Klassenstruktur der Komponente *CALServer* ist in Abbildung 44 dargestellt.

Der *CALServer* übernimmt damit ebenfalls Aufgaben der Klasse *DsController* der Annotationsarchitektur. Die *DsDesigner*-Klasse für Video-und Audioannotationen ist die Klasse *CALServerRecorder*. Zur eigentlichen Erstellung eines konkreten Mediastreams kommt eine Instanz der - der Klasse *DsTool* entsprechenden - Klasse *ConcreteRecorder* zum Zug. Die konkrete Klasse für die Instanz wird dynamisch über Einstellungen zur Laufzeit festgelegt. Momentan stehen hier drei Alternativen zur Verfügung: *RealRecorder*, *QuickTimeRecorder*, *DummyRecorder*. *DummyRecorder* wird benutzt, wenn nur eine Aufzeichnung der Notizen ohne Video erstellt werden soll. Die beiden anderen Klassen erzeugen MediaStreams in den jeweiligen Formaten. Dazu nutzt *RealRecorder* über Java Native Interface (JNI) die von Real Inc. in C zur Verfügung gestellten Schnittstellen zur Anwendungsentwicklung (Software Development Kit, SDK, vgl. Helix Community, 2005). *QuickTimeRecorder* greift direkt auf die von Apple zur Verfügung gestellten Java-Klassen zurück (vgl. Mareemaa & Stewart, 1999).

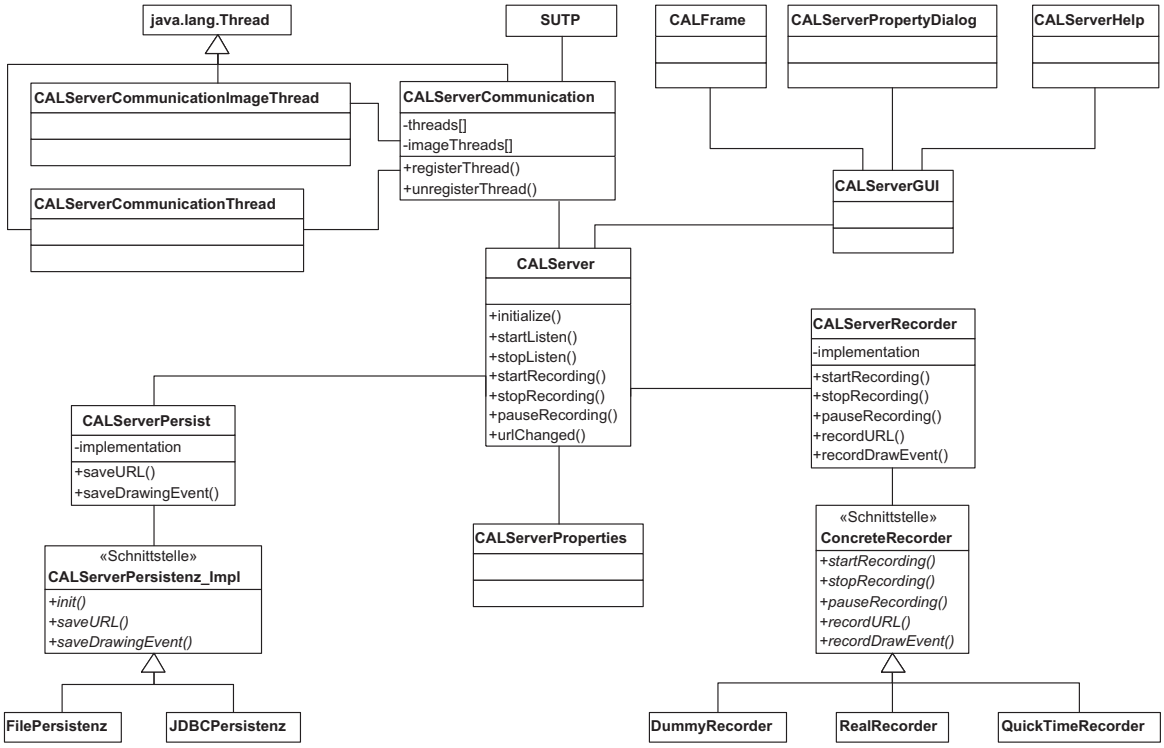


Abb. 44: Klassenmodell zur CALServer Komponente

Neben der Implementierung des Tätigkeitsfelds der Komponente *Designer* im Bezug zu Video und Audio, übernimmt *CALServer* auch Aufgaben der Komponente *Central Server* der Annotationsarchitektur. So entspricht die Klasse *CALServerPersist* dem *PersistenzController* und die Schnittstelle *CALServerPersistenz_Impl* dem *PersistenzManager*. Mit *FilePersistenz* und *JDBCPersistenz* wurden zwei konkrete *PersistenzManager* implementiert, von denen einer je nach Einstellung zur Laufzeit erstellt und genutzt wird. FilePersistenz stellt ein eigenes, dateibasiertes, einfaches Datenbanksystem zur Verfügung. Damit wird die Speicherung von Daten ohne zusätzliche Voraussetzungen ermöglicht. Die Klasse JDBCPersistenz legt Daten in einer über JDBC angebundenen Datenbank ab. Damit kann der Benutzer den besseren Funktionsumfang einer SQL-Datenbank nutzen, hat aber auch die Anforderung, ein solches Datenbanksystem zu installieren und zu administrieren.

Für die Netzwerkverbindung und den Annotationsaustausch nutzt die CALServer Komponente die Klassen *CALServerKommunikation, CALServerCommunicationThread* und *CALServerCommunicationImageThread*. Da der Server gleichzeitig mit mehreren CAL- und CALDrawing-Applets kommunizieren können soll, wurde der Kommunikationsteil als konkurrierender, multithreaded Server erstellt (vgl. Middendorf, Singer & Heid, 2002: Kap. 13.2.). *CALServerKommunikation* läuft als eigener Thread und wartet auf Verbindungswünsche einer Clientkomponente. Versucht eine Clientkomponente eine Verbindung aufzubauen, wird ein neuer Thread *CALServerCommunicationThread* oder *CALServerCommunicationImageThread* gestartet, der diese Verbindung je nach Art abwickelt. Das konkrete zu benutztende Protokoll ist in der Klasse *SUTP* gekapselt. Für Genaueres hierzu sei auf Ovadya (1999) verwiesen.

6.1.4. Bewertung

Das CAL-System wird seit nunmehr gut fünf Jahren am Lehrstuhl Angewandte Informatik und kooperative Systeme der Fakultät für Informatik der Technischen Universität München im Vorlesungsbetrieb eingesetzt. Während dieser Zeit wurde es kontinuierlich erweitert und den Anforderungen des Dozenten angepasst. Die zugrundeliegende Architektur erwies sich aus Programmierersicht als sehr gut erweiterbar.

Die Erweiterung der Benutzungsoberfläche für die Annotationen um einige integrierte, untergeordnete Elemente, sprich also die Einbindung von Steuerflächen zur Erstellung von Annotationen direkt in das Kursskript, erwies sich als große Verbesserung. Der zusätzliche Klick zum Aktivieren der Steuerungsoberfläche von CAL vor dem Erstellen einer Notiz konnte somit entfallen. Die Navigation des Dozenten wurde für die Studenten klarer und verständlicher. Auch ist der Dozent nun nicht mehr so lange durch die Navigation abgelenkt. Dies unterstreicht noch einmal, dass die Ergonomie der Benutzungsoberfläche nicht vernachlässigt werden darf. Deshalb sollte auf Ebene der Steuerungselemente immer ein untergeordnet integrierter Ansatz angestrebt werden.

Die Browserumgebung als System für die Darstellung des Kursskripts ist vorgegeben. Damit kann bei Erhalt der Unabhängigkeit von speziellen Browsern kaum Einfluß auf das Kursskriptsystem genommen werden. Dadurch wurden die Systeme zur Verwaltung der Annotationen und des Kursskripts von vorne herein stark getrennt, wie es auch in der Annotationsarchitektur gefordert wird. Diese Trennung erwies sich als problemlos. Die schmale, in der Annotationsarchitektur geforderte Schnittstelle zwischen beiden Systemen reicht vollkommen aus.

Das CAL-System hat die erste Ausbaustufe der Annotationsarchitektur problemlos erreicht. Die zweite und dritte Ausbaustufe werden in nächster Zeit angegangen.

6.2. Integration von Annotationen in das WOTAN System

Das WOTAN System ist eine Erweiterung von TANGOW (Task-based Adaptive learNer Guidance On the WWW , vgl. Carro, Pulido & Rodríguez, 1999) und soll im Zuge des UCAT Projekts (Ubiquitous Collaborative Adaptive Training) eingesetzt werden (vgl. Rodriguez, 2005).

TANGOW unterstützt momentan keine Annotationen. Zum Einen wird dies als Defizit für eine elektronische Lernumgebung angesehen und soll deshalb in WOTAN ergänzt werden. Zum Anderen ist im UCAT Projekt eine Unterstützung von kollaborativen Aspekten erforderlich. Dies ist auf Ebene von Annotationen vorstellbar. Das Stichwort *Ubiquitous* bringt den Aspekt der Adaptierbarkeit der Anwendungen auf den jeweilige äußeren Kontext hin, wie zum Beispiel die benutzte Hardware, ins Spiel.

Alle drei eben genannten Punkte werden bereits von der in der Arbeit vorgestellten Annotationsarchitektur berücksichtigt. Aus diesem Grund wird eine Integration von Annotationen mit Hilfe dieser Annotationsarchitektur durchgeführt.

6.2.1. Architektur des WOTAN Systems

Im Folgenden wird die Architektur des WOTAN Systems vorgestellt. Dabei konzentrieren sich die Ausführungen auf die Bereiche, die für einen Überblick und die Integration von Annotationen wichtig sind. Es werden somit zentrale, vom WOTAN-System angebotene Teile,

wie der Mechanismus zur Anpassung an den jeweiligen Studenten, nicht detailliert besprochen.

Das WOTAN-System dient der elektronischen Bereitstellung von Kursen für das Selbststudium. Eine Spezialität hierbei ist, dass das System Informationen über den Studenten sammelt und dann einen Kurs auf den jeweiligen Studenten anpasst. Das Erheben der Daten erfolgt dabei durch einen Fragebogen zu Beginn eines Kurses und über die Informationen von bereits bearbeiteten Kursteilen. Der Lernfortschritt wird kontinuierlich über Testaufgaben geprüft. Ein Fortschreiten im Kurs kann vom erfolgreichen Bestehen solcher Tests abhängig gemacht werden.

Neben diesen Aspekten der studentischen Sicht auf das System bietet WOTAN auch Unterstützung für die Erstellung von Kursen und die Administration der Studenten. Die zu integrierende Annotationsunterstützung betrifft momentan nur die Studenten, weshalb auf diese Teile von WOTAN nicht detaillierter eingegangen wird.

WOTAN vermittelt Kurse über das World Wide Web (WWW). Als Hintergrundtechnologie dienen dazu Servlets und Java Server Pages (JSP) (vgl. Sun Microsystems, 2001) die über Apache Tomcat 5.5 (vgl. Apache Software Foundation, 2005) interpretiert und angeboten werden. Als direkte Konsequenz daraus ergibt sich, dass für das Bearbeiten eines Kurses eine Netzwerkverbindung zum jeweiligen Server (hier zu Tomcat) benötigt wird.

Das Modell für Kurse in WOTAN wird in Abbildung 45 verdeutlicht. Ein Kurs wird über einen *Course* repräsentiert. Ein solcher *Course* verweist auf einen *Task* (Haupttask). Jeder *Task* besteht aus *Rules*, über die andere *Tasks* untergeordnet werden können und, beziehungsweise oder, *Fragments* über die konkrete Inhalte verlinkt werden. Der konkrete Inhalt liegt dabei in einer der dem *Fragment* angehängten *Versions*.

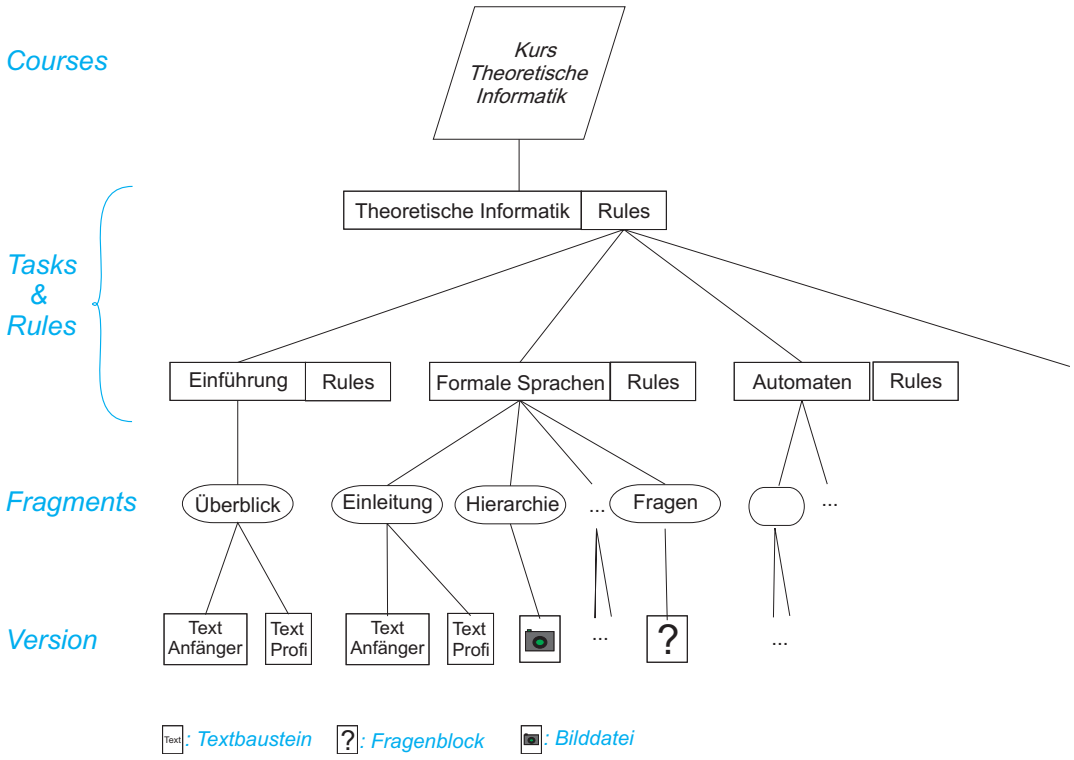


Abb. 45: Modell eines Kurses in WOTAN

Versions sind aus Sicht des Kursmodells von WOTAN atomar und bilden die kleinsten Bausteine. Soll einem Benutzer ein gewisser *Task* gezeigt werden, wird über den aktuellen Kontext und geltenden *Rules* entschieden, welche *Version* eines *Fragments* dargestellt wird. Es kann also verschiedene *Versions* zu einem *Fragment* geben, wobei einem Studenten immer nur eine angezeigt wird. Regelmäßig wird dieser Mechanismus benutzt, um Anfängern eine ausführlichere Darstellung des Sachverhalts eines *Fragments* zu geben als Fortgeschrittenen.

Das dargestellte Kursmodell wird bei der Durchführung der Integration zur Festlegung der Referenzierung von Annotation und Kursskript benutzt.

Das WOTAN-System ist in drei Hauptkomponenten gegliedert:

- Admin

  Diese Komponente erlaubt das Verwalten von Kursen und Studenten

- Exerciser

  Zu jedem Kursabschnitt kann es einen Fragenblock geben, der dem Studenten Auskunft über seinen Wissensstand gibt. Diese Fragenblöcke werden über die Exerciser Komponente angeboten und ausgewertet.

- Presenter

  Über die *Presenter*-Komponente werden die zur Verfügung stehenden Kursinhalte präsentiert. Dabei nutzt *Presenter* eine *AdaptionEngine*. Diese *AdaptionEngine* entscheidet mit Hilfe der *Rules* für die *Tasks*, welche Kursinhalte zu jedem Zeitpunkt angeboten werden und welche konkreten Repräsentationen, also *Versions* dafür benutzt werden.

Alle drei Hauptkomponenten werden jeweils durch ein eigenes gleichnamiges Servlet repräsentiert.

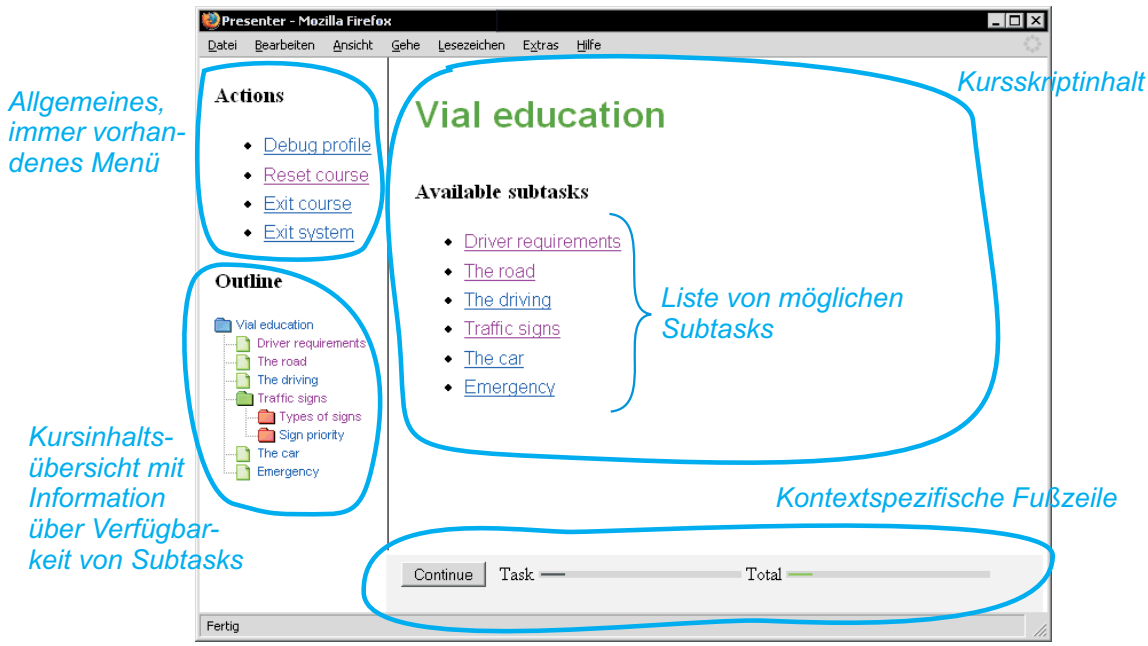


Abb. 46: Aufbau der Darstellung einer Kursskriptstelle in WOTAN

Da die Annotationsfunktionalität Studenten zuerst zur Ergänzung ihres Kursskripts zur Verfügung gestellt werden soll, steht im Weiteren die *Presenter*-Komponente im Vordergrund.

Die von der Presenter-Komponente ausgelieferte Darstellung einer Stelle im Kursskript hat prinzipiell immer den in Abbildung 46 gezeigten Aufbau. Der Kursskriptinhalt entspricht dabei einem *Task*. Je nach den *Rules* werden Verweise zu untergeordneten *Tasks* angeboten (wie in der Abbildung zu sehen) und, beziehungsweise oder, *Fragments* angezeigt.

Zu jedem Kursskriptinhalt wird eine kontextspezifische Fußzeile angezeigt. Sie stellt dar, wo sich der Studenten im aktuellen *Task* und wo er sich im gesamten Kurs befindet. Der Knopf „Continue“ führt je nach Kontext und Student über den Aufruf von `prozessNext()` der *Presenter*-Komponente weiter zu einem anderen *Tasks*.

Die Bereiche links geben allgemeinere Auskunft. Der obere Block bietet dabei die global zur Verfügung stehenden Aktionen an. Der untere Block stellt einen Überblick über den gesamten Kurs als Inhaltsverzeichnis dar. Dabei erkennt der Student über die unterschiedliche Färbung der Punkte, welche zur Auswahl stehen (grün), für welche noch eine Vorbedingung zu erfüllen ist (rot) und welche schon bearbeitet wurden (weiß).

Das sequentielle Durchlaufen eines Kurses steuert die Funktion `prozessNext()` der Presenter-Komponente. Sie arbeitet wie im Sequenzdiagramm in Abbildung 47 angegeben. Wichtig ist hier zu klären, welche Aufrufe welche Bereiche in der oben vorgestellten Benutzungsoberfläche erstellen. Damit wird klar, wo bei einer späteren Integration von Annotationen angesetzt werden kann.

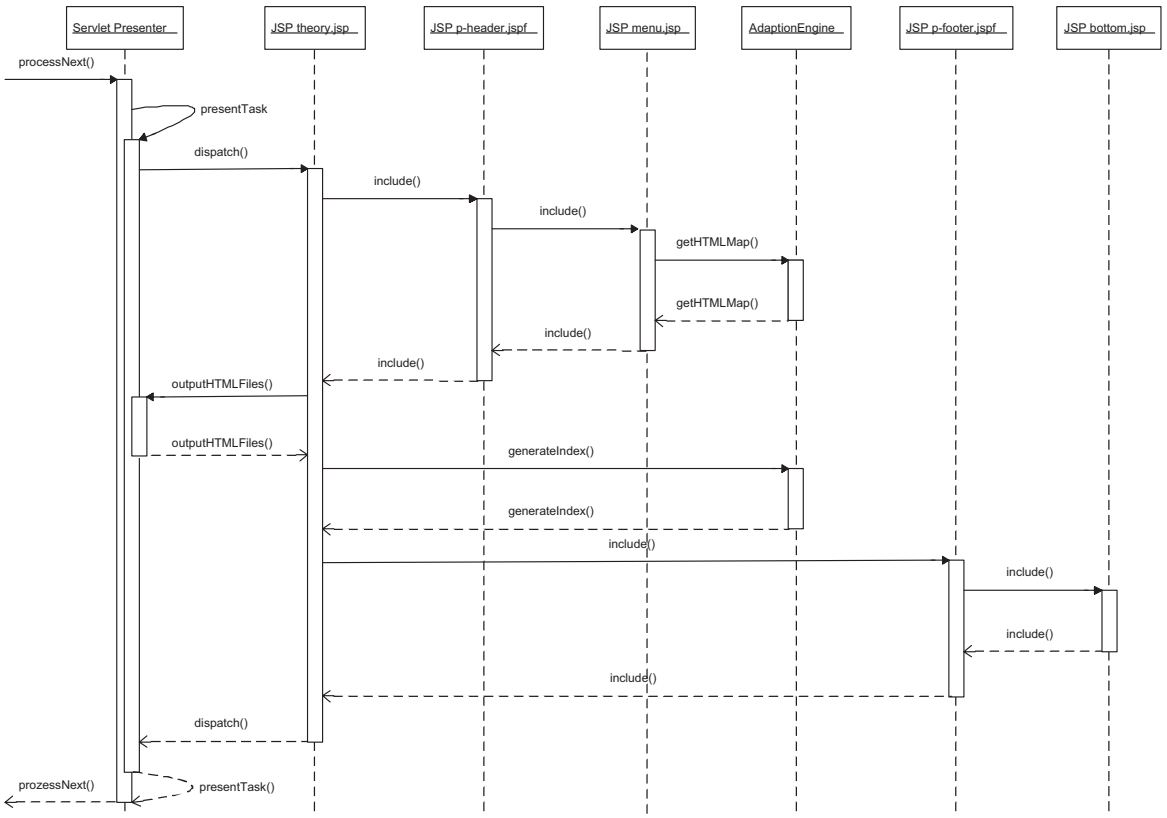


Abb. 47: Sequenzdiagramm für Presenter.prozessNext()

Der Inhalt des linken allgemeinen Bereichs wird von *menue.jsp* bestimmt. Dabei ist der obere Block zur Verfügung stehender allgemeiner Aktionen direkt in *menue.jsp* gegeben. Der untere Teil mit der Darstellung des Kurses als Inhaltsverzeichnis wird dynamisch über die Methode `getHTMLMap()` der jeweiligen *AdaptionEngine* eingebunden.

Der Bereich für das Kursskript rechts wird über zwei Methoden gefüllt: Zuerst werden über `outputHTMLFiles()` des *Presenter*-Servlets zu allen Fragments dieses Tasks die zum Kontext passenden *Versions* ausgegeben. Anschließend werden über `generateIndex()` der *AdaptionEngine* anhand der *Rules* Verweise auf die in diesem Kontext möglichen untergeordneten *Tasks* ergänzt.

Die kontextspezifische Fußzeile rechts unten wird über *bottom.jsp* gestaltet.

Insgesamt erstellen die genannten Funktionen den textlichen Inhalt des jeweiligen Bereichs. Das Erscheinungsbild wird über Cascading Style Sheets (CSS) (vgl. W3C, 2005c) geregelt.

Soweit die Ausführungen zum Ausgangssystem.

6.2.2. Anforderungen an die Integration

Folgende Anforderungen an die Integration von Annotationen wurden an eine erste Umsetzung gestellt:

- WOTAN1: Studenten können zu einzelnen Tasks Annotationen verfassen und betrachten.
- WOTAN2: In der ersten Ausbaustufe sind Erscheinungsformen von Annotationen, die mit Hilfe eines Textwerkzeugs erstellbar sind, ausreichend.
- WOTAN3: Studenten können Annotationen anderer anzeigen und übernehmen.
- WOTAN4: Ein einfaches Browsen in den eigenen Annotationen ist auch ohne Kursskript möglich.
- WOTAN5: Die komplette Steuerungsoberfläche zu den Annotationen erscheint im WOTAN-System integriert.

Die Anforderung WOTAN5 bedingt, dass für die Annotationsunterstützung Änderungen in Java ServerPages und Servlets des WOTAN-Systems notwendig sind. Dabei soll die Trennung von Annotationsunterstützung und ursprünglichem WOTAN-System aber möglichst scharf bleiben.

Dass das WOTAN-System einen Kurs nur über einen zentralen Server anbietet, ergibt kombiniert mit Anforderung WOTAN5, dass auch das Annotationssystem von einer ständigen Verbindung zum WOTAN-Server ausgehen kann. Insofern sind keine lokalen Komponenten beim Benutzer (beachte auch WOTAN2) notwendig. Die Nutzung von Textwerkzeugen lässt sich komplett über eine Webschnittstelle realisieren.

6.2.3. Durchführung der Integration

Die Integration von Annotationen geschieht durch direktes Einbinden der Referenzimplementierung zur vorgestellten Annotationsarchitektur (vgl. Schütz, 2005b). Die in Kapitel 5.4. angeführte einfache Anpassung der Annotationsarchitektur an bestehende Systeme zur Unterstützung der Lehre wird damit unter Beweis gestellt. Hier ist besonders erwähnenswert, dass sich das Modell für die Kommunikation zwischen Anwender und Programm in WOTAN grundsätzlich von dem nicht webgestützter Applikationen unterscheidet. Die

Kommunikation läuft über einen Anfrage-Antwort (Request-Response) Mechanismus ab. Die Anwendung kann also primär nicht selbst aktiv werden, sondern nur auf Anfragen der Benutzer reagieren. Trotzdem erweist sich, wie im folgenden dargestellt, die Annotationsarchitektur auch als dafür flexibel genug.

In einem ersten Schritt werden die notwendigen Erweiterungen im WOTAN-System vorgestellt. *Tasks* stellen in der weiteren Umsetzung die logische Verbindungsebene zwischen Annotationssystem und WOTAN dar. Ein Student kann also eine Annotation zu einem Task erstellen. Wird der Task später wieder angezeigt, erscheint auch ein Hinweis auf die Annotation. Dieser Festlegung und den obigen Ausführungen zur Gliederung der Benutzungsoberfläche bei der Darstellung von Kursskriptinhalten folgend, werden die Steuerungselemente und die Darstellungsflächen für Annotationen in der Fußzeile unter dem Kursskriptinhalt angesiedelt. Dazu wurde die Datei *bottom.jsp* am Ende um folgende Zeile erweitert:

```
<%=PresenterWithAnnotation.getAnnotationStuff(sTaskID)%>
```

Diese Zeile bewirkt, dass die statische Methode `String getAnnotationStuff(String sTaskID)` der Klasse *PresenterWithAnnotation* mit dem Identifikator des angezeigten Tasks aufgerufen wird. Diese Methode liefert HTML-Code, der direkt in die Antwort an den Benutzer übernommen wird.

Die Klasse *PresenterWithAnnotation* erbt von der Klasse *Presenter* des WOTAN-Systems und ersetzt diese im sonst gleich gebliebenen WOTAN-System. Alle annotationsrelevanten Ergänzungen zu WOTAN werden in ihr gekapselt und die ursprüngliche Logik der Klasse Presenter über den Vererbungsmechanismus weiterverwendet.

Die Methoden der Klasse *PresenterWithAnnotation* lassen sich in zwei Kategorien aufspalten: Methoden des WOTAN-Systems, die das Annotationssystem einrichten und unterstützen und Methoden, die *PresenterWithAnnotation* in der Rolle des Kursskripts innerhalb des Annotationssystems bereitstellen muß.

Zur ersten Kategorie zählen:

- public void init(ServletConfig config)

  Sie ruft die init()-Methode der ursprünglichen Presenter-Klasse auf. Anschließend führt sie initAnnotEZ() aus.

- private void initAnnotEZ()

  initAnnotEZ() richtet die Umgebung der Annotationsunterstützung ein. Darunter fällt das Starten des zentralen Annotationsservers über startCentralServer(), das Setzen einiger Einstellungen für Annotationen in der Registry und das Einrichten des *WotanScriptStub*-Objekts als Verbindung zwischen Kursskript- und Annotationssystem

- private void startCentralServer(ServletConfig config)

  Diese Methode startet den zentralen Annotationsserver als eigenständige Applikation außerhalb der Servletumgebung. Auf den ersten Blick mag das Nutzen eines zentralen Annotationsservers verwundern, da ja im Fall des WOTAN-Systems das Annotationssystem wegen des Einsatzes von Servlettechnologie sowieso schon zentral läuft. Entweder die Local Storage- oder die Central Server-Komponente erscheint unnötig. Die Designentscheidung trotzdem beide Komponenten zu behalten, begründet sich über mehrere Aspekte. Der pragmatischste ist wohl, dass damit kaum Änderungen im Annotationssystem notwendig sind. Überzeugender ist aber die Art der Datenhaltung im WOTAN-System. Für jeden Be-

nutzer wird unter einem Verzeichnis /users ein Unterverzeichnis angelegt und alle anfallenden Daten dort in entsprechende XML-Dateien abgelegt. Somit hat im Benutzermodell des WOTAN-Systems jeder Benutzer einen eigenen privaten Bereich. Diese Logik wird der Konsistenz wegen auch in der Annotationsunterstützung beibehalten. Die *Locale Storage*-Komponente legt die Annotationsdaten also nicht lokal auf dem Rechensystem des Benutzers ab, sondern lokal im privaten Verzeichnis des Benutzers auf dem Server. Damit bleibt ein Migrieren eines kompletten Benutzers auf ein anderes WOTAN-System auch inklusive der Annotationsinformation einfach durch kopieren des Verzeichnisses möglich. Die *Central Server*-Komponente wird eigenständig weiterbenutzt, da die Daten typischerweise in ein Datenbanksystem laufen und dort die Anfragen über mehrere Benutzer hinweg erheblich effizienter bearbeitet werden, als über die Suche in Dateien in verschiedenen Unterverzeichnissen.

- protected void processRequest(HttpServletRequest request, HttpServletResponse response)

  Diese Methode nimmt in Servletumgebungen Anfragen des Benutzers entgegen und erstellt die zurückzuliefernde Antwort. Im konkreten Fall informiert sie die Annotationsumgebung zuerst über refreshScriptReference(), dass sich die anzuzeigende Kursskriptseite verändert hat. Das WOTAN-System kodiert in die Anfragen an Servlets die vom Benutzer gewünschte Aktion im Anfrageparameter `op`. Diese Technik wird von der Annotationsunterstützung übernommen. Demzufolge wird im nächsten Schritt geprüft, ob die Anfrage eine Aktion im Annotationssystem auslösen soll. Wenn ja, wird die Anfrage an die Methode processAnnotationAction() weitergegeben. Diese Methode liefert als Ergebnis, ob die Anfrage komplett durch das Annotationssystem abgehandelt wurde. Falls die Anfrage nicht komplett bearbeitet ist oder von vornherein nicht für das Annotationssystem bestimmt war, wird die Methode processRequest der ursprünglichen Presenter-Klasse aufgerufen. Mit dieser Technik sind nur minimale Ergänzungen im Kursskriptsystem notwendig und die Wiederverwendung des ursprünglichen WOTAN-Programmcodes maximal.

  Die Umsetzung von processRequest() sieht konkret wie folgt aus:

```
protected void processRequest(HttpServletRequest request,
                              HttpServletResponse response)
                              throws ServletException, IOException {
    boolean done = false;
    String op = (String) request.getAttribute("op");

    refreshScriptReference();

    try {
        if (op != null && op.equals("annotation_action")) {
            done = processAnnotationAction(request, response);
            if (!done) { request.setAttribute("op", "task"); }
        }

        if (!done) {
            super.processRequest(request, response);
        }
    } catch (Exception ex) {
        showError(response.getWriter(), "Error processing request", ex);
    }
}
```

- private boolean processAnnotationAction(HttpServletRequest request, HttpServletResponse response)

  Diese Methode ist ein gutes Beispiel für die strikte Trennung der Annotationslogik von der Kursskriptlogik. Die Methode extrahiert nur den actionCode, der die gewünschte Aktion im Annotationssystem kodiert aus der Anfrage und ruft die Methode doAction(actionCode) des *WotanScriptStub*-Objekts auf. Eine eigene Verarbeitung erfolgt an dieser Stelle nicht. Nur das Annotationssystem ist verantwortlich für die Abarbeitung von Benutzeranfragen bezüglich Annotationen. Wird die Benutzungsschnittstelle eigenständig vom Annotationssystem geliefert, wie bei parallel getrennten Steuerungselementen, so ist dies problemlos. Bei untergeordnet, integrierten Steuerungselementen ist dies weniger offensichtlich. Abbildung 48 verdeutlicht das allgemeine Schema.

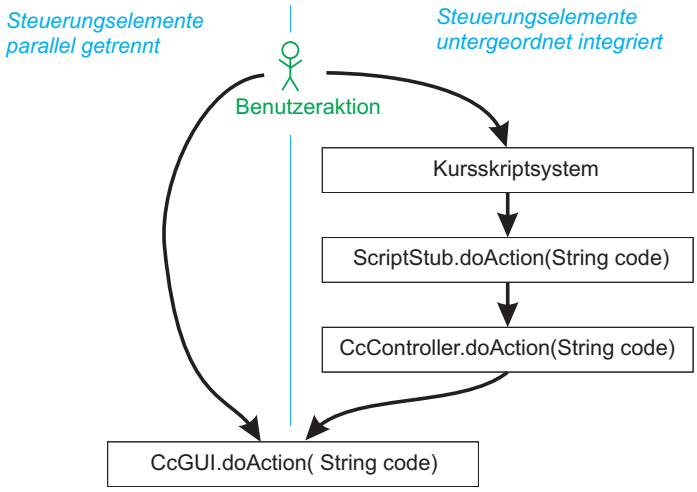


Abb. 48: Bearbeitungswege von Benutzeraktionen im Annotationssystem

Benutzeraktionen werden also bei untergeordnet integrierten Steuerungselementen vom Kursskriptsystem über das *SkriptStub*- und *CcController*-Objekt an das Benutzungsoberflächenobjekt *CcUI* übergeben, als kämen sie direkt von einer Benutzungsoberfläche des Annotationssystems. Wird dieser Mechanismus eingehalten, ist ein Umstieg zwischen parallel getrennten und untergeordnet integrierten Steuerungselementen nur mit geringem Aufwand verbunden.

- public static String getAnnotationStuff(String sTaskID)

  Diese Methode bietet dem Annotationssystem die Möglichkeit, wie oben bei *bottom.jsp* erwähnt, Steuerungselemente im Fußbereich der Kursskriptdarstellung unterzubringen. Außerdem werden bei dieser Umsetzung auch vorhandene Annotationen angezeigt. Damit sind die Darstellungsflächen horizontal verschränkt.

  Hier die Implementierung der Methode:

```
public static String getAnnotationStuff(String sTaskID) {
    // Annotation Menu
    String html = (String) _scriptStub.createMenu();

    // Annotations to this Taskid
    html += (String) _scriptStub.getAnnotationSummary();

    return html;
```

```
}
```

Wie diesen recht übersichtlichen Programmzeilen leicht zu entnehmen ist, wird auch hier vom Kursskriptsystem keine eigene Verarbeitung durchgeführt, sondern über das *ScriptStub*-Objekt die jeweiligen Methoden im Annotationssystems angestoßen.

Dies waren die Methoden, die im WOTAN-System ergänzt wurden, um das Annotationssystem aufzusetzen und in das Kursskript zu integrieren. Nun werden kurz die Methoden vorgestellt, die das WOTAN-System dem Annotationssystem für eine Kopplung zum Kursskript zur Verfügung stellen muss:

- void refreshScriptReference();

  Das Annotationssystem nutzt diese Methode, um vom Kursskriptsystem die aktuelle Position im Kursskript zu erhalten. Bei der aktuellen Implementierung im WOTAN-System wird eine Annotation einem *Task* zugeordnet. Als Kursskriptreferenz wird somit im Feld *LogicalMark* der Kursskriptreferenz der Identifikator des angezeigten Tasks eingetragen. Alle anderen Felder der Kursskriptreferenz bleiben momentan leer.

- Object getObject(String objectID);

  Über diese Methode können auf das WOTAN-System spezialisierte Klassen des Annotationssystems Daten vom WOTAN-System abfragen. Im konkreten Fall dient diese Methode dem *Text_WOTAN_Servlet*-Toolobjekt in der *Design*-Komponente zur Abfrage des eingegebenen Texts.

- String getUserID();

  Bietet das Kursskriptsystem schon eigenständig eine Benutzerverwaltung, kann diese vom Annotationssystem über ein geeignetes *UsrManager*-Objekt mit benutzt werden. Wird das *UsrManager*-Objekt intern im Annotationssystem nach der Benutzerkennung gefragt, gibt es die Anfrage über das *ScriptStub*-Objekt an das Kursskriptsystem weiter. Diese Anfrage mündet in der hier aufgeführten Methode `getIUserID()`.

Neben diesen doch recht überschaubaren Ergänzungen im WOTAN-System wurden im Annotationssystem einige Ergänzungen vorgenommen. Wichtig ist hier herauszustellen, dass die Infrastruktur innerhalb der Annotationsarchitektur keinerlei Änderungen erfährt. Es werden nur spezialisierte Schnittstellenklassen zur Anpassung an die geänderte Umgebung des Annotationssystems über Vererbung ergänzt. Das ganze wurde über circa zweihundert zusätzliche Quellcodezeilen erreicht, was die einfache Anpassbarkeit der Annotationsarchitektur deutlich unterstreicht. Die Ergänzungen können im Anhang S.176ff nachgelesen werden.

6.2.4. Bewertung

Das WOTAN-System erfährt mit der Integration von Annotationen eine deutliche Verbesserung. Das dabei für Studenten nicht einfach nur die Möglichkeit hinzukam, isoliert eine Notiz zu machen, sondern diese auch mit anderen Studenten auszutauschen und in den Annotationen suchen zu können, ist erheblich mehr, als eine eigene Umsetzung mit diesem geringen Aufwand hätte bieten können.

Dass hinter der vorgeschlagenen Annotationsarchitektur ein leistungs- und sehr anpassungsfähiges Konzept steckt, zeigt sich besonders darin, dass auch ein prinzipiell anderes Modell für die Kommunikation zwischen Benutzer und System, wie es das Request-Response Schema bei Servlets darstellt, kein Problem ist.

In einem weiteren Schritt werden andere Annotationsformen, wie zum Beispiel Hervorhebung eingebaut. Dafür werden Referenzen feingliedriger als auf Task Ebene gebraucht. Auch die Integration der Darstellungsflächen wird dann noch enger, da diese Annotationen zum Beispiel als farbige Hervorhebung direkt im Kursskriptinhalt integriert werden und nicht wie jetzt am Ende des Kursskriptinhalt zusammengefasst auftauchen.

Des Weiteren wird WOTAN im Zuge das UCAT-Projekts um die Fähigkeit ergänzt, auf verschiedene Ausgabemedien reagieren zu können und die Anzeige entsprechend anzupassen. Diese Fähigkeit sollte sich problemlos über die vorhandenen Mechanismen in das Annotationssystem integrieren lassen. Dies kann aber erst nach der momentan durchgeführten Erweiterung des WOTAN-Systems getestet werden.

Insgesamt ergänzen sich die beiden Systeme technisch gesehen hervorragend. Bleibt abzuwarten, ob sich die Sinnhaftigkeit im praktischen Einsatz bestätigt.

7. Bewertung

7.1. Zusammenfassung

Durch eine Zusammenfassung der verschiedenen Aspekte von Annotationen in der Lehre, die so übergreifend und doch konzentriert und unter Einbindung der Rollen Dozent und Student noch nicht gegeben wurde, konnte im Zuge dieser Arbeit eine fundierte Begründung für die Notwendigkeit von Annotationen in der Lehre gegeben werden. Aus dieser Notwendigkeit und dem aktuell bei elektronischer Lehre meist in Kauf genommenen Medienbruch folgt direkt, dass auch eine elektronische Umsetzung von Annotationen in elektronisch unterstützter Lehre wünschenswert ist.

Des Weiteren wurde ein allgemeingültiger Anforderungskatalog für eine elektronische Umsetzung von Annotationen in elektronischen Systemen zur Unterstützung der Lehre auf Basis von Lehr- und Lernprozessen erstellt. Mit diesem Anforderungskatalog wird eine Bewertung verschiedener Systeme möglich und eine Entscheidungshilfe bei der Wahl eines elektronischen Systems zur Unterstützung der Lehre gegeben. Diese Entscheidungshilfe kommt insbesondere auch Nicht-Technikern zugute.

Verschiedene, meist prototypische Implementierungen von Annotationen in elektronischen Systemen zur Unterstützung der Lehre wurden betrachtet und mit dem Anforderungskatalog verglichen. Dabei wurden folgende Probleme erkannt:

- Annotationsunterstützung meist nur für den Dozenten
- Annotationen existieren meist nur isoliert beim Ersteller
- Kaum Zusammenarbeit auf Annotationsebene möglich
- Möglichkeiten der elektronischen Umgebung meist nicht voll genutzt

Die meisten Umsetzer von Annotationen in der Lehre erstellen früher oder später ein eigenständiges komplettes Lehrsystem. Die didaktische Relevanz und Nutzbarkeit des einzelnen Konzepts *Annotation* wird dabei oft hinter der technischen Machbarkeit zurückgelassen (vgl. Schütz, 2002). Zwar gibt es gute Einzelideen und -lösungen. Insgesamt fehlen aber *allgemeine* Architektur- und Umsetzungsvorschläge für Annotationen in elektronisch unterstützter Lehre. Annotationen werden bei der Entwicklung eines Systems zur elektronischen Unterstützung von Lehre oft nur als Anhängsel betrachtet. Gerade deshalb ist es für Entwickler solcher Systeme wünschenswert, sich nicht lange um eine Architektur für eine Umsetzung von Annotationen kümmern zu müssen, sondern auf ein fundiertes Gerüst aufbauen zu können. Dieses wird mit dieser Arbeit gegeben.

Bevor im Rahmen dieser Arbeit an die Entwicklung eines solchen eigenständigen, allgemeinen Architekturvorschlags gegangen wurde, erfolgte zusätzlich die Betrachtung von Standardsoftware mit Annotationsmöglichkeit. Dabei konnten weitere wünschenswerte Anforderungen an Annotationssysteme identifiziert werden. Zu diesen zählt beispielsweise das Arbeiten nur mit den Annotationen, ohne den annotierten Inhalt.

Mit diesem Rüstzeug ausgestattet, wurde eine eigene, universelle Annotationsarchitektur entwickelt. Diese erlaubt sehr schnell, umfassende Annotationsunterstützung in bestehenden Systemen zu ergänzen. Außerdem ist der Vorschlag stufig angelegt. Es besteht damit die

Möglichkeit, Annotationen über drei Phasen hinweg schrittweise zu integrieren. Zudem ist bei einer auf dieser Architektur basierenden Anwendung eine Erweiterung um spezielle Annotationsformen jederzeit ohne großen Aufwand über Vererbung und eingebaute, dynamische Lademechanismen möglich.

Für die Annotationsarchitektur wurde im Zuge der Arbeit eine Referenzimplementierung in Java erstellt (vgl. Schütz, 2005b). Über diese Referenzimplementierung konnte durch Durchführung einer Integration von Annotationen in das WOTAN-System die Tragfähigkeit der vorgestellten Lösung demonstriert werden.

Bisher waren Annotationen im Umfeld der elektronischen Lehre kaum Inhalt eigenständiger Forschung. Falls doch, wurden meist eigene Systeme über sehr spezielle Implementierungen um die Möglichkeit von Annotationen ergänzt. Hier stellt die vorliegende Arbeit durch die Loslösung von einem konkreten System, also eine klare Innovation dar. Der erzeugte Mehrwert liegt eindeutig in der schnelleren und fundierteren Umsetzung von Annotationen in jegliche System zur elektronischen Unterstützung der Lehre. Die Möglichkeit der phasenweisen Umsetzung erlaubt eine schrittweise Einführung von Annotationen, ohne Gefahr zu laufen, in einer Sackgasse zu landen. Die problemlose individuelle Erweiterung der möglichen Annotationsformen erlaubt es, auch sehr speziellen Anforderungen gerecht zu werden.

Eine besondere Herausforderung in dieser Arbeit lag von vorne herein darin, mehrere Disziplinen über diese Arbeit zu einem gemeinsamen Verständnis von Annotationen zu bewegen. Insbesondere bekommen Pädagogen und Didaktiker ein durchdachtes Grundgerüst an die Hand, mit dessen Hilfe auch sie, als in der Entwicklung von Softwaresystemen weniger routinierte Personen, Annotationen umsetzen beziehungsweise Umsetzungen von Annotationen bewerten können.

7.2. Eigene Erfahrungen

Die Erfahrungen der technischen Durchführung der Integration von Annotationen in bestehende Systeme wurden in Kapitel 6.2. dargestellt.

Nach Abschluss dieser technischen Betrachtung wird nun über die Betrachtung der eigenen praktischen Erfahrungen zu Annotationen im Umfeld elektronisch unterstützter Lehre der Kreis zum Ausgangspunkt in der Einleitung - nämlich der pädagogischen Relevanz - geschlossen.

Dass ein Bedarf an der Möglichkeit von elektronischen Annotationen für Dozenten besteht, lässt sich aus der steigenden Anzahl an Dozenten die elektronische Annotationen nutzen (vgl. z.B. Schulte, 2003) und aus den regelmäßigen Anfragen nach unseren persönlichen Erfahrungen ablesen. Beim Versuch, die Beweggründe herauszuarbeiten, wird regelmäßig das sowieso schon elektronisch vorhandene Kursskript als Ausgangspunkt genannt. Die Dozenten sehen es als nachteilig an, das Kursskript elektronisch zu präsentieren, aber Annotationen auf einem anderen Medium machen zu müssen. Das Vermeiden von Annotationen wird als unbefriedigend empfunden. Der in der vorliegenden Arbeit mehrfach angeführte Medienbruch ist also oftmals ein Beweggrund.

Wir haben in unseren eigenen Vorlesungen zudem festgestellt, dass bei vorzeitiger Herausgabe eines vollständigen Kursskripts, die Studenten eher unkonzentriert sind oder der Vorlesung ganz fernbleiben. Hier hat sich die Möglichkeit bewährt, ein stichpunktartiges Kursskript anzubieten, welches in der Vorlesung dynamisch über Annotationen ergänzt wird.

Nach unserer langjährigen Erfahrung mit Medien zur Eingabe von Dozentenannotationen können wir sagen, dass mit den heute zur Verfügung stehenden, preislich attraktiven Tablet-PC's hardwaremäßig nichts mehr gegen elektronische Annotationen spricht. Mit dem Tablet-PC ist ein einziges Gerät notwendig, dessen Bedienung im Umfeld von Annotationen dem des Overheadprojektors gleichkommt. Steht ein fest installiertes Smartboard mit Rückprojektion zur Verfügung, ist der Arbeitsaufwand und Arbeitskomfort ähnlich dem bei der Arbeit mit einer Tafel. Alle anderen Varianten sind der Komplexität und des Aufwandes wegen aus unserer Sicht wenig empfehlenswert.

Nach unseren Erfahrungen an der Fakultät für Informatik, TU München ist der Ruf nach Annotationsunterstützung auf Seiten der Studenten noch recht leise. Dies liegt zum Einen sicher an den Kosten. Zum Anderen haben die Studenten im Gegensatz zum Dozenten noch ein anderes Problem: Sie hören verschiedene Vorlesungen, in denen momentan verschiedene Systeme zur Kursskriptdarstellung verwendet werden. Damit sind potentiell auch ebenso viele verschiedene Systeme zur Annotationserstellung notwendig. Sie alle bedeuten zumindest einen gewissen Einarbeitungsaufwand. Hier ist es denkbar, dass die Sache bei Fernuniversitäten anders liegt. Bieten diese Ihre Kurse über ein einheitliches Format an und ist die Annotationsunterstützung einheitlich gelöst, ist auch hier ein steigender Bedarf unter den Studenten zu erwarten. Werden dann noch Mehrwerte, wie Austauschen von Annotationen und Arbeiten alleinig auf den Annotationen (Suchen, Filtern, ...) unterstützt, ist auch hier mit breiter Akzeptanz zu rechnen.

7.3. Ausblick

Aus den nun langjährigen Erfahrungen mit elektronischen Annotation im Lehrumfeld ergeben sich Fragestellungen, die einen weiteren Forschungsbedarf aufzeigen.

So ist eine interessante Frage, inwiefern sich die vom W3C im Bereich Semantic Web angeregten Annotationen für die Lehre eignen. Prinzipiell ist eine Kopplung der Bereiche denkbar. Das Semantik Web versucht, Wissen zu repräsentieren und die Pädagogik versucht, Wissen zu vermitteln. Beide Sparten haben erkannt, dass eine gewisse Individualisierung innerhalb der Domäne angebracht ist. Beide haben Annotationen als Konzept dafür entdeckt. Einzig die Erwartungen an die Annotationen und die Benutzungsschnittstellen sind noch recht unterschiedlich. Weitere interdisziplinäre Forschung in diesem Bereich könnte interessante Früchte tragen.

Zu den eher technisch ausgerichteten Fragestellungen zählt die Umsetzung der Lehrumgebung als campusweit verfügbarer Webservice. Speziell die Unterstützung des Dozenten bei real gehaltenen Vorlesungen ist hierbei noch ein kaum beachtetes Feld. Könnte der Dozent am Rechner in seinem Büro das Kursskript vorbereiten und hätte dieses dann in allen Hörsälen problemlos mit gleicher Funktionalität und Annotationsunterstützung zur Verfügung, so wäre die Akzeptanz des Einarbeitungsaufwands größer. Momentan stellt eine Verlegung in einen anderen Hörsaal den Dozenten oft vor technische Probleme, da er meist sein eigenes System mitbringen und auf die neue Situation anpassen muss.

Eine weitere, noch recht visionär klingende Fragestellung ergibt sich aus der Nutzung von 3D-Anzeigegeräten. Verfolgt ein Student einen Kurs über das Kursskript, erstellt dabei eigene Annotationen und erhält zudem Informationen über Annotationen anderer Studenten, ist dies eine anspruchsvolle Aufgabenstellung für die Gestalter der Benutzungsoberfläche. Wird ein Multilayer-Monitor für die Darstellung benutzt, können verschiedene Informationen auf verschiedene Layer gelegt werden. Nach einer Eingewöhnungsphase ist mit einem raschen

Erkennen und Auffinden von Informationen zu rechnen. Speziell im Umfeld der Lehre sollten damit mögliche Effekte untersucht werden.

Aus der vorliegenden Arbeit ergibt sich auch weiterer Forschungsbedarf mehr am Rande der Technik. Diese Fragestellungen sind überhaupt erst, durch das Vorhandensein einer umfassenden Annotationsunterstützung praktikabel zu untersuchen.

Bisher gibt es elektronische Annotationen in der Lehre gerade einmal an Universitäten. Wieso nicht auch an Schulen? Welche Vorteile, aber auch welche Risiken sind zu erwarten?

Eine weitere Fragestellung an die Pädagogik beziehungsweise Didaktik ist: Welche neuen Möglichkeiten erhält der Dozent über das automatische Auswerten des Annotierverhaltens? Lassen sich hier Metriken definieren? Können Verhaltensmuster identifiziert und daraus neue Fördermöglichkeiten abgeleitet werden?

Insgesamt bleibt das Thema Annotationen in der Lehre spannend. Nach den durch diese Arbeit beigesteuerten Grundlagen vielleicht sogar ein wenig spannender...

Quellenverzeichnis

Abeck, S. (2004).
ed.learn: Verteilte Informationssysteme (Kurs)
URL: *http://i71rs03.cm-tm.uka.de/areas/extern/customization* (dort: Lehre->Verteilte Informationssysteme)

Abowd, G.D., Brotherton, J.A. & Bhalodia, J. (1998).
Classroom 2000: A System for Capturing and Accessing Multimedia Classroom Experiences. In: *CHI '98 Demonstration Paper*, May, 1998

ADLNET (2005).
Homepage zu SCORM,
URL: *http://www.adlnet.org/scorm/* (Stand: 4.5.2005)

Adobe (2005).
Adobe Acrobat 7.0 Professional Datenblatt. URL: *http://www.adobe.de/products/acrobat/pdfs/02_Acrobat7StandDB.pdf* (Stand: 5.7.2005)

Apache Software Foundation (2005).
The Apache Jakarta Tomcat 5.5 Servlet/JSP Container Documentation.
URL: *http://jakarta.apache.org/tomcat/tomcat-5.5-doc/index.html*
(Stand: 23.5.2005)

Apple (2005).
Quicktime Homepage, Apple Computer, Inc,
URL: *http://www.apple.com/quicktime* (Stand: 1.6.2005)

Arnold, H.F. (1942).
The comparative effectiveness of certain study techniques in the field of history.
In: *Journal of Educational Psychology*, 32, S. 449-457

Bacher, C., Müller, R., Ottmann, T. & Will, M. (1997).
Authoring on the fly. In: H.U. Hoppe & W. Luther (Hrsg.) *Informatik und Lernen in der Informationsgesellschaft*, S. 14-18, Springer

Bitzer , D.L., Braunfeld, P. & Lichtenberger, W. (1961).
PLATO: An automatic teaching device.
In: *IRE Transactions on Education*, E-4, S. 157-161

BMBF (2004).
Projektträger Neue Medien in der Bildung + Fachinformation des BMBF (Hrsg.)
Notebook University - Ergebnisse und Erfahrungen einer Förderinitiative, Stand: August 2004

Bolour, A. (2003).
Notes on the Eclipse Plug-in Architecture.
URL: *http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html* (Stand: 4.5.2005)

Brokmann-Nooren, Ch., Rieken, H. & Schröder, A. (1995).
Studieneinheit »Lehr- und Lernziele«. In: Ch. Brokmann-Nooren, I. Grieb &

H.-D. Raapke (Hrsg.): *NQ-Materialien. Handbuch Erwachsenenbildung*, S. 9-66, Weinheim und Basel: Beltz

Buzan, T. (1993).
The Mind Map Book, London: BBC Books

Brenneisen, W. (1988).
Survival in der Schule, München: Heyne-Verlag

Bretzing, B.H. & Kulhavy, R.W. (1981).
Note-Taking and Passage Style.
In: *Journal of Educational Psychology*, 73-2, S. 242-250

Brockschmidt, K. (1995).
Inside OLE - 2nd Edition. Redmond, Washington: Microsoft Press

Carro, R.M., .Pulido, E. & Rodríguez, P. (1999).
Task-based Adaptive learNer Guidance On the WWW: the TANGOW System.
In: *Proceedings of the Eighth International World Wide Web Conference*, Toronto, Canada, May 11-14, 1999.
Computer Science Report 99-07, Eindhoven University Technology, S. 49-57.

Caverly, D.C. & Orlando, V.P. (1991). Textbook study strategies. In R.F. Flippo & D.C. Caverly (Hrsg.): *Teaching reading and study strategies at the college level*, S. 86-165, Newark, DE: International Reading Association

Clayberg, E. & Beck, K. (2004).
Eclipse: Building Commercial-Quality Plug-Ins. Portland: Addison-Wesley

Dewey, J. (1938).
Experience and education. New York: Collier MacMillan Publishers

Dilts, R.B., Dilts, R.W. & Epstein T. (1991).
Tools for Dreamers: Strategies for Creativity and the Structure of Innovation. Cupertino, Ca.: Meta Publications

Di Vesta, F.J. & Gray, S.G. (1972).
Listening and notetaking. In: *Journal of Educational Psychology*, 64, S. 278-287

Driscoll, M. & Thomson, R. (1997).
The Web as Learning Environment, In: *WET-ICE '97: Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, S. 333-340, Washington: IEEE Computer Society

Dunkel, P. & Davy, S. (1989).
The heuristic of lecture note taking: Perceptions of American and international students regarding the value and practice of note taking.
In: *English for Specific Purposes*, 8, S.33-50

Dutra, J., Gibbons, J., Pannoni, R., Sipusic, M., Smith, R. & Sutherland, W. (1999).
Virtual Collaborative Learning: A comparison between Face-To-Face Tutored Video Instruction (TVI) and Distributed Tutored Video Instruction (DTVI), *SUN Microsystems Research Technical Report TR-99-72*

eCase (2004).
Homepage zu eCase. URL: *http://www-i7.informatik.rwth-aachen.de/d/projects/uli/ecase.html* (Stand: 15.11.2004)

Europäisches Parlament und Rat (1995).
Beschluss Nr. 2493/95/EG des Europäischen Parlaments und des Rates vom 23. Oktober 1995 über die Veranstaltung eines Europäischen Jahres des lebensbegleitenden Lernens (1996)
In: *Amtsblatt Nr. L 256 vom 26/10/1995*, S. 45-48). Brüssel: Europäische Gemeinschaft

Feuerhelm, D., Bonn, M. & Abeck, S. (2002).
Eine Werkzeugarchitektur zur Unterstützung von Autoren und Dozenten in der Internet-basierten Aus- und Weiterbildung.
In: *Proceedings zu LIT2002*, Leipzig

Furnas, G. (1981).
The Fisheye View: new look at structured files.
Bell Laboratories technical memorandum, #81-11221-9.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1996).
Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software
Aus dem amerikanischen von Dirk Riehle, Bonn: Addison-Wesley-Longman

Grüner, G. (1961).
Arbeiten mit der Kreide. Braunschweig: Westermann

Hartley, J. (1976).
Lecture handouts and student note-taking.
In: *Programmed Learning Educational Technology* , 13, S. 58–64

Hartley, J. (1983).
Notetaking research: Resetting the scoreboard.
In: *Bulletin of the British Psychological Society*, 36, S. 13-14

Helix Community (2005).
Helix DNA Producer SDK Documentation.
URL: *https://producersdk.helixcommunity.org/docs/* (Stand: 16.6.2005)

IEEE (2002).
Draft Standard for Learning Object Metadata
URL: *http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf*
(Stand: 3.12.2004)

Iles, A., Glaser, D., Kam, M. & Canny, J. (2002).
Learning via Distributed Dialogue: Livenotes and Handheld Wireless Technology. In *Proceedings of Conference on Computer Support for Collaborative Learning*, Boulder, Colorado, January 2002.

Kahan, J., Koivunen, M.-R., Prud'hommeaux, E. & Swick, R.R. (2001).
Annotea: An Open RDF Infrastructure for Shared Web Annotations.
In: *The Tenth International World Wide Web Conference*, Hong Kong, S. 623-632

Kam, M., Tarshish, O., Glaser, D., Iles, A. & Canny, J. (2002).
Communicating through Handheld Wireless Tablets: Livenotes and Shared Group Awareness. In: *Supplemental Proceedings of ACM Conference on Computer Supported Cooperative Work*

Kam, M., Wang, J., Iles, A., Tse, E., Chiu, J., Glaser, D., Tarshish, O. & Canny, J. (2005).
Livenotes: A System for Cooperative and Augmented Note-Taking in Lectures.
To appear in: *Proceedings of ACM Conference on Human Factors in Computing Systems (Portland, Oregon) 2005*

Kiewra, K.A. (1985).
Students' note taking behaviors and the efficacy of providing the instructor's notes for review. In: *Contemporary Educational Psychology*, 10, S. 378-386

Kiewra, K.A. & DuBois, N.F. (1991).
Note-Taking Functions and Techniques
In: *Journal of Educational Psychology*, 83, S. 240-245

Kiewra, K.A. (2002).
How classroom teachers can help students learn and teach them how to learn.
In: *Theory Into Practice*, 41, 4, S. 265-267

Köhne, S., Ruisz, R. & Krcmar, H. (2002).
Werkzeuge für das E-Learning. In: T. Sommerlatte, M. Bellmann & H. Krcmar (Hrsg.). *Praxishandbuch Wissensmanagement. Strategien - Methoden - Fallbeispiele*. Düsseldorf: Symposion Publishing

Kommission der Europäischen Gemeinschaft (1993).
Wachstum, Wettbewerbsfähigkeit, Beschäftigung: Die Herausforderungen und Wege vorwärts ins 21. Jahrhundert. Brüssel: Europäische Kommission.

Kommission der Europäischen Gemeinschaft (2000).
Memorandum über Lebenslanges Lernen [SEK(2000) 1832],
Brüssel: Europäische Kommission.

Krupina, O. (2005).
Client-Server Architecture for a Neural Simulation Tool, PhD thesis,
Fachbereich Mathematik und Informatik, Freie Universität Berlin

Kuwan, H. & Waschbüsch, E. (1998).
Delphi-Befragung 1996/1998 - Abschlußbericht zum „Bildungs-Delphi", Potentiale und Dimensionen der Wissensgesellschaft - Auswirkungen auf Bildungsprozesse und Bildungsstrukturen, München: Bundesministerium für Bildung und Forschung.

Laurillard, D. (1979).
The Process of Student Learning. In: *Higher Education* 8, S. 395-409

Lave, J. & Wenger, E. (1991).
Situated learning - Legitimate peripheral participation.
Cambridge: Cambridge University Press

Leonard-Barton, D. (1995).
Wellsprings of Knowledge: Building and Sustaining the Sources of Innovation, Boston, Massachusetts: Harvard Business School Press

Leonardo, L.M. (1997).
Using Netscape Liveconnect, Que Pub

Litosseliti, L., Marttunen, M., Laurinen, L. & Salminen, T. (2002).
Computer-Based and Face-To-Face Collaborative Argumentation: Analysing secondary school student's interaction in the UK and in Finland. In: B. Méndez-Vilas, and J. M. González (Hrsg.) *Information Society and Education: Monitoring a Revolution - Proceedings of ICTE 2002*, Badajoz: Formatex

Liwicki, M. (2004).
Erkennung und Simulation von logischen Schaltungen für E-Chalk, Diplomarbeit, Fachbereich Mathematik und Informatik, Freie Universität Berlin

Mackinlay, J.D., Robertson, G.G. & Card, C.K. (1991).
The perspective wall: Detail und context smoothly integrated.
In: *Proceedings of CHI'91*, S. 173-179

Mandl, H. (1981).
Vorwort In: H. Mandl (Hrsg.) *Zur Psychologie der Textverarbeitung: Ansätze, Befunde, Probleme*, München: Urban und Schwarzenberg

Maple (2005).
Maplesoft Homepage, URL: *http://www.maplesoft.com/* (Stand 3.4.2005)

Maras, R. (1979).
Das Tafelbild im Unterricht: Grundlegung, Gestaltung, Anwendung, München: Lurz

Mareemaa, T. & Stewart, W. (1999).
QuickTime for Java: A Developer Reference,
Morgan Kaufmann & Academic Press

Marton, F. & Säljö (1976a).
On Qualitative Differences in Learning — 1: Outcome and Process.
In: *British Journal of Educational Psychology*, 46, S. 4-11

Marton, F. & Säljö (1976b).
On Qualitative Differences in Learning — 2: Outcome as a function of the learner's conception of the task. In: *British Journal of Educational Psychology*, 46, S. 115-27

Maurer, H. & Dietinger, T. (1997).
MANKIND: A General Description.
URL: *http://www.iicm.edu/mankind.htm* (Stand: 7. Juni 2004)

McAndrew, D.A. (1983).
Underlining and notetaking: Some. Suggestions from research,
In: *Journal of Reading*, 27(2), S. 103-108

McDoughall, I.R., Gray, H.W. & McNicol, G.P. (1972).
The effect of timing of distribution of 'handouts' on improvement of student performance. In: *British Journal of Medical Education*, 6, S. 155-157

McFarlane, N., Chiarelli, A., De Carli, J., Li, S., Updegrave, S., Wilcox, M., Wilton, P. & Wootton, C. (1999).
Professional JavaScript, Birmingham: Wrox Press Ltd.

McGraw, G. & Felten, E.W. (1999).
Securing Java: Getting Down to Business with Mobile Code, 2nd Edition, Wiley

Microsoft (2005).
Microsoft Homepage zu Tablet PC's. URL: *http://www.microsoft.com/windowsxp/tabletpc/default.mspx* (Stand: 1.7.2005)

Middendorf, S., Singer, R. & Heid, J. (2002).
JavaTM, Programmierhandbuch und Referenz für die JavaTM-2-Plattform, Standard Edition, 3. Auflage, Heidelberg: dpunkt.verlag GmbH

Moallem, M., Kermani, H. & Chen, S.J. (2002).
Handheld, Wireless Computers: How they can improve learning and instruction, In: B. Méndez-Vilas, and J. M. González (Hrsg.) *Information Society and Education: Monitoring a Revolution - Proceedings of ICTE 2002*, Badajoz: Formatex, S. 684-688

Müller, R. & Ottmann, T. (2002).
Electronic Note-taking, Systems, Problems and their use at Universities, In: H.H. Adelsberger, B. Collis, J.M. Pawlowski (Hrsg.), *Handbook on Information technologies for education and training*, Springer-Verlag, S. 121-134

Netscape (2004).
Netscape Plug-in Software Developement Kit.
URL:*http://wp.netscape.com/comprod/development_partners/plugin_api/*
(Stand: 3.4.2004)

Nist, S.L. & Simpson, M.L. (1988).
The effectiveness and efficiency of training college students to annotate and underline texts. In: J.E. Readence & R.S. Baldwin (Hrsg.), *Dialogues in literacy research: 37th yearbook of the National Reading Conference*, S. 321-328, Chicago, IL: National Reading Conference

Nonaka, I. & Takeuchi, H. (1995).
The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation, New York: Oxford University Press Inc.

Norman, D.A. (1980).
Cognitive engineering and education. In D.T. Tuma & F. Reif (Hrsg.), *Problem solving and education*. Hillsdale, New Jersey: Erlbaum

Object Technology International (2001)
Whitepaper: Eclipse platform technical overview (Juli 2001, ergänzt Februar 2003) URL *http://www.eclipse.org/whitepapers/eclipse-overview.pdf*

O'Donnell, A. & Dansereua, D.F. (1993).
Learning from lectures: Effects of cooperative review.
In: *Journal of Experimental Education*, 61, S. 116-125

Oesterreich, B. (1998).
Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified modeling language, 4., aktualisierte Auflage, München, Wien: Oldenburg

OMG (2005).
CORBA- Spezifikation. URL: *http://www.omg.org/technology/documents/corba_spec_catalog.htm* (Stand: 28.5.2005)

Open Group (1997).
CAE Specification DCE 1.1: Remote Procedure Call, Document Number: C706

Ovadya, D. (1999).
Implementierung einer Aufnahme- und Ablaufumgebung für Vorlesungen über Real G2 mit URL-Unterstützung, Diplomarbeit, eingereicht am Institut der Informatik - Technische Universität München

Palmatier, R.A., & Bennett, J.M. (1974).
Note-taking habits of college students. *Journal of Reading*, 18, S. 215-218

Pederson, E., McCall, K., Moran, T.P. & Halasz, F.G. (1993).
Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings.
In: *Proceedings INTERCHI 93: Human Factors in Computing Systems*. Amsterdam, S. 391-398.

Peters, D.L. (1972)
Effects of note taking and rate of presentation on short-term objective test performance. In: *Journal of Educational Psychology*, 63-3, S. 276-280

Prenzel, M., Baumert, J., Blum, W., Lehmann, R., Leutner, D., Neubrand, M., Pekrun, R., Rolff, H.-G., Rost, J. & Schiefele, U. (Hrsg.). (2004).
PISA 2003. Der Bildungsstand der Jugendlichen in Deutschland - Ergebnisse des zweiten internationalen Vergleichs. Münster: Waxmann

Pressley, M., McDaniel, M.A., Turnure, J.E., Wood, E. & Ahmad, M. (1987).
Generation and precision of elaboration: Effects on intentional and incidental learning. In: *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13, S. 291-300

RBG TU Darmstadt (2004).
Flyer zu NUTools Schulung. URL: *http://www.nu.tu-darmstadt.de/DLZ/FlyerNU-Schulung.doc* (Stand: 3.11.2004)

RealNetworks (2003).
Dokumentation zum Aufbau von Verweisen auf HTML-Seiten, URL *http://service.real.com/help/videoccg/synchmm.html#frames* (Stand: 10.7.2003)

RealNetworks (2005).
RealNetworks, Inc. Homepage,
URL: *http://www.realnetworks.com* (Stand: 1.6.2005)

Rickards, J. & August, G.J. (1975).
Generative underlining strategies in prose recall.
In: *Journal of Educational Psychology*, 67, S. 860-865

RFC2046 (1996).
Multipurpose Internet Mail Extensions (MIME), Part Two: Media Types.
URL: *http://www.isi.edu/in-notes/rfc2046.txt* (Stand: 5.7.2005)

Rodriguez, P.M. (2005).
Homepage UCAT Projekt.
URL: *http://orestes.ii.uam.es/ucat/index.php* (Stand: 16.6.2005)

Rößling, G., Bär, H. & Mühlhäuser, M. (2004).
Interaction Support in Large Scale Lectures. In: *Proceedings of ITiCSE'04*, ACM

Rhode, P. & Thomas, W. (2003).
Ein e-Lecture-System für die Theoretische Informatik. In: A. Bode, J. Desel, S. Ratmayer, M. Wessner (Hrsg.) *DeLFI 2003: Proceeding der 1. e-Learning Fachtagung Informatik*, GI-Edition - Lecture Notes in Informatics (LNI), P-37, S. 17-26, Bonn: Bonner Köllen Verlag

Savery, J. R. & Duffy, T. M. (1995).
Problem based learning: an instructional model and its constructivist framework.
In: *Educational Technology*, 35 (5), S. 31-38.

Schlichter, J. (2000).
Lecture 2000 Homepage. URL: *http://www11.in.tum.de/proj/lecture/*

Schlichter, J. (2001).
Einführung in die Informatik 3, WS 2001/02, Institut für Informatik, Technische Universität München, Homepage
URL: *http://www11.in.tum.de/lehre/vorlesungen/ws2001-02/info3/index.html*

Schlichter, J. (2003).
Einführung in die Informatik 3, WS 2003/04, Institut für Informatik, Technische Universität München, Homepage
URL: *http://www11.in.tum.de/lehre/vorlesungen/ws2003-04/info3/index.html*

Schlichter, J. (2005).
Verteilte Anwendungen, SS 2005, Institut für Informatik, Technische Universität München, Homepage
URL: *http://www11.in.tum.de/lehre/vorlesungen/ss2005/va/index.html*

Schnell, T.R. & Rocchio, D.J. (1978).
A comparison of underlining strategies for improving reading comprehension and retention. *Reading Horizons*, 18, S. 106-109 .

Schütz, F. (2001).
CAL (Call A Lecture) Projekt Homepage.
URL *http://www11.in.tum.de/proj/cal/index.html*

Schütz, F. (2002).
Annotations: Though standard in conventional learning a stepchild of elearning.
In: B. Méndez-Vilas, and J. M. González (Hrsg.) *Information Society and Educa-*

tion: Monitoring a Revolution - Proceedings of ICTE 2002, S. 199-203, Badajoz: Formatex

Schütz, F. (2003).
Producing eLearning materials on the fly - only a great dream?
In: Mendez Vilas, A., Gonzales, J.A.M., Solo de Zaldivar, I. (Hrsg.) *Advances in technology-based education: Towards a knowledge-based society - Proceedings of mICTE2003*, S. 959-963, Badajoz

Schütz, F. (2004).
Unveröffentlichte, interne Umfrage auf der Vorlesungsverwaltungsseite zum Grundstudium Informatik, Dezember 2004, vgl. Anhang, S. 173

Schütz, F. (2005a).
Unveröffentlichte, interne Umfrage auf der Vorlesungsverwaltungsseite zum Grundstudium Informatik, Juni 2005, vgl. Anhang, S. 174

Schütz, F. (2005b).
Homepage zu AnnotEZ, der Referenzimplementierung zur erarbeiteten Annotationsarchitektur. URL: *http://www11.in.tum.de/proj/annotEZ/index.html*

Schulte, J. (2003).
Evaluation des Einsatzes der Software „E-Kreide“ in der universitären Lehre, Magisterarbeit, TU Berlin, 7. November 2003

Seel, N. M. (2000).
Psychologie des Lernens, München: Reinhard

Seufert, S., Back, A. & Häusler, M. (2001).
E-Learning, Weiterbildung im Internet. Das „Plato-Cookbook“ für interentbasiertes Lernen. 1.Auflage, Kilchberg: SmartBooks Publishing

Slavin, R. (1990).
Cooperative Learning: Theory, Research, and Practise. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

SmartTech (2005).
Herstellerhomepage zu SmartBoard.
URL:*http://www.smarttech.com/* (Stand: 1.2.2005)

Sun Microsystems (2001).
Java(TM) Servlet 2.3 and JavaServer Pages(TM) 1.2 Specifications.
URL: *http://jcp.org/aboutJava/communityprocess/first/jsr053/index.html* (Stand: 23.5.2005)

TechSmith (2004).
Produkthomepage zu Camtasia Studio.
URL: *http://www.techsmith.com/products/studio/* (Stand: 3.11.2004)

Teege, G. (2002).
Targeteam Projekt Homepage. URL*http://www.targeteam.org/*

Teege, G. (2000).
Hypermedia - Konzepte und Sprachen im World Wide Web, SS 2000, Institut für

Informatik, Technische Universität München, Vorlesungshomepage
URL: *http://www11.in.tum.de/lehre/vorlesungen/ss2000/hypermedia/index.html*

Teufel, S., Sauter, C., Mühlherr, T. & Bauknecht, K. (1995).
Computerunterstützung für die Gruppenarbeit.
Addison-Wesley Publishing Company

Truong K.N., Abowd, G.D. & Brotherton, J.A. (1999).
Personalizing the capture of public experience.
In: *Proceedings of ACM UIST'99* (November 7-10, Asheville, NC), S.121-130

Trompler, C., Mühlhäuser, M. & Wegner, W. (2002).
OPEN CLIENT LECTURE INTERACTION: An Approach to Wireless Learners-in-the-Loop. In: *Proceedings of 4th International Conference on New Educational Environments.* S. 43-46

Trompler, C., Rößling, G., Bär, H.C. & Choi, C.M.M. (2003).
Kooperative Digitale Mitschriften auf mobilen Computern. In: A. Bode, J. Desel, S. Ratmayer & M. Wessner (Hrsg.) *DeLFI 2003: Proceeding der 1. e-Learning Fachtagung Informatik*, GI-Edition - Lecture Notes in Informatics (LNI), P-37, S. 17-26, Bonn: Bonner Köllen Verlag

Tung, T.L. (1997).
MediaBoard: A Shared Whiteboard Application for the MBone, M.S. thesis, University of California, Berkeley

RBG TU Darmstadt (2005).
Homepage für das Notebook University Tool *TVRemote* der Rechnerbetriebsgruppe TU Darmstadt,
URL: *http://www.nu.tu-darmstadt.de/TVremote* (Stand: 1.7.2005)

Verein zur Förderung Berufspädagogischer Forschung e. V. (Hrsg.) (1968).
Der Overhead-Schreibprojektor : eine neuartige Arbeitstafel.
Stuttgart: Neckar-Verlag

Vester, F. (1978).
Denken, Lernen, Vergessen- Was geht in unserem Kopf vor, wie lernt das Gehirn und wann lässt es uns im Stich?. München: dtv

W3C (1998).
Document Object Model (DOM) Level 1 Specification, Version 1.0, W3C Recommendation 1 Oktober, 1998
URL: *http://www.w3.org/TR/REC-DOM-Level-1/* (Stand: 24.6.2005)

W3C (2005a).
Homepage zum Annotea Protokoll
URL: *http://www.w3.org/2001/Annotea/* (Stand: 24.6.2005)

W3C (2005b).
Homepage zum Annotea-fähigen Browser Amaya
URL: *http://www.w3.org/Amaya/* (Stand: 24.6.2005)

W3C (2005c).
Cascading Style Sheets Homepage.
URL: *http://www.w3.org/Style/CSS/* (Stand:24.6.2005)

Wacom (2005).
Herstellerhomepage zu Cintiq und Volito.
URL: *http://www.wacom.com* (Stand: 1.2.2005)

Weidenmann, B. (1994).
Lernen mit Bildmedien: psychologische und didaktische Grundlagen, 2. Auflage, Weinheim, Basel: Beltz Verlag

Weinand, A. (1992).
Objektorientierte Architektur für grafische Benutzungsoberflächen - Realisierung der portablen Fesntersystemschnittstelle von ET++,
Berlin, Heidelberg: Springer-Verlag

Weinstein, C. E. (1982).
Training Students to use elaboration learning strategies,
In: *Contemporary Educational Psychology*, 7, S.301-311

Weinstein, C.E. & Mayer, R.E. (1985).
The teaching of learning strategies. In: M.C. Wittrock (Hrsg.), *Handbook of research on teaching*, S. 315-327, New York: Macmillan

Wharrad, H.J. & Leeder, D. (2003).
Experiences with re-usable learning objects: process and evaluation. Paper presented at the *Association for Learning Technology conference (ALT-C): Communities of Practice*, Sheffield, 8-10 September 2003

Wild, K.P. (1996).
Beziehungen zwischen Belohnungsstrukturen der Hochschule, motivationalen Orientierungen der Studierenden und individuellen Lernstrategien beim Wissenserwerb.
In: Lompscher Joachim, Mandl Heinz (Hrsg.), *Lehr- und Lernprobleme im Studium. Bedingungen und Veränderungsmöglichkeiten.* Bern:Verlag Hans Huber

Xu, C. (2000)
Interaction and Cooperation Mechanisms for Distributed Communities and Groups in Educational Settings, Dissertation eingereicht bei: Technische Universität München, Fakultät für Informatik, 25.4.2000

Yang, S.J.H., Chen, I.Y.L. & Shao, N.W.Y. (2004).
Ontology Enabled Annotation and Knowledge Management for Collaborative Learning in Virtual Learning Community. In: *Educational Technology & Society*, 7 (4), S. 70-81.

Ziewer, P. & Seidl, H. (2002).
Transparent TeleTeaching
In: *Proceedings of ASCILITE 2002*, Auckland, NZ, Dec. 2002

Zirk, O. (1969).
Die Schule des Tafelzeichnens. München:Ehrenwirth

Index

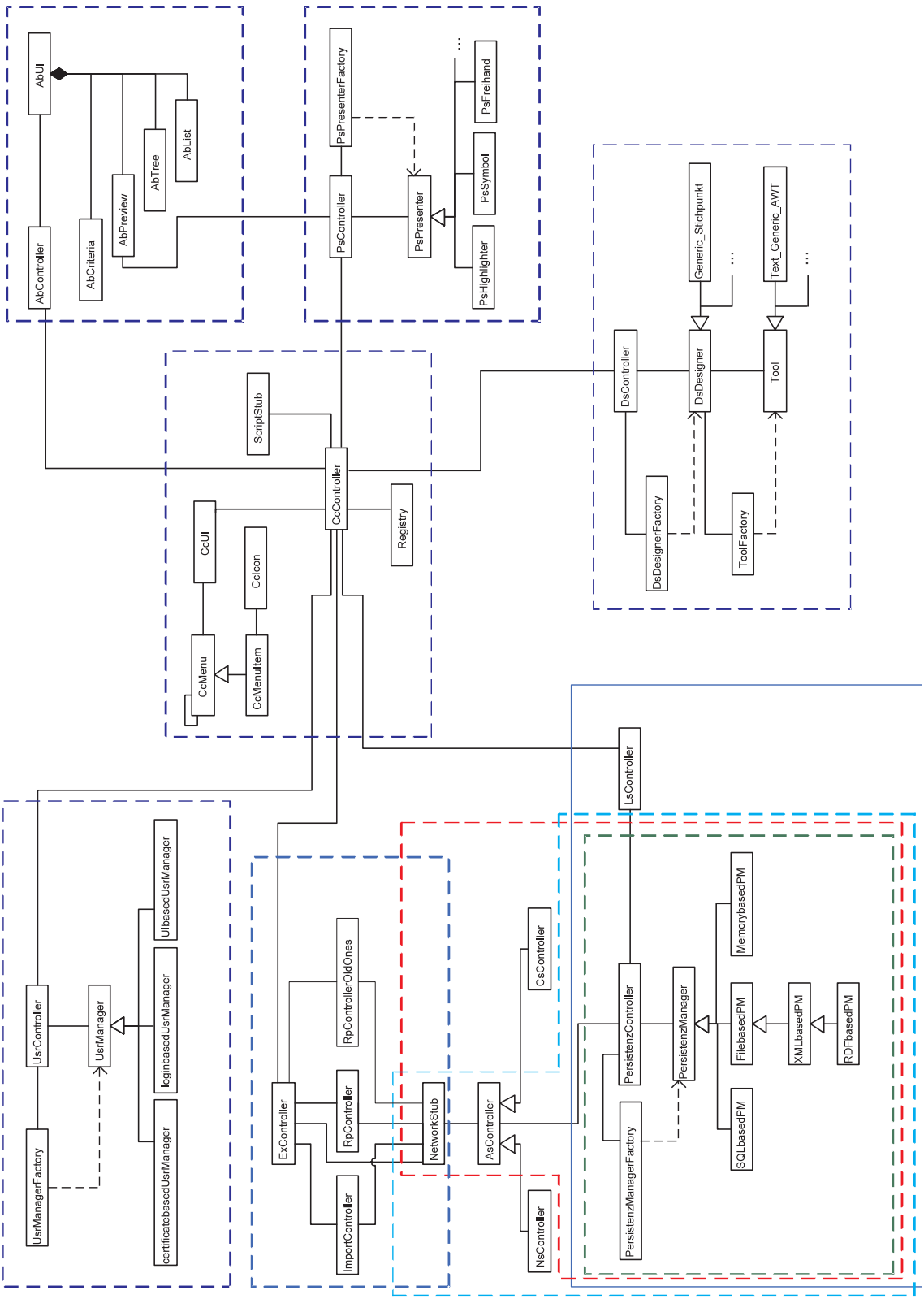
V

W

Z

Anhang

Klassendiagramm Gesamtüberblick



Umfrage: Verfügbarkeit von Notebooks

Interne Umfrage im Grundstudium der Informatik, Technische Universität München: Verfügbarkeit von Notebooks für den Übungsbetrieb, Dezember 2004

Umfrage: Notebooks

(Die Zahl hinter den Alternativen gibt die Anzahl der Studenten an, die diese Alternative gewählt hat.)

Können Sie auf ein Notebook zurückgreifen, um es einmal wöchentlich zum Übungsbetrieb mitzubringen?

Ja. Immer.	345
Meistens.	131
Sehr selten.	68
Nein.	310

Umfrage: Elektronische Annotationen in der Lehre - Studentensicht

Interne Umfrage im Grundstudium der Informatik, Technische Universität München: Elektronische Annotationen in der Lehre, Juni 2005

(Die Zahl hinter den Alternativen gibt die Anzahl der Studenten an, die diese Alternative gewählt hat.)

1. Was halten Sie davon die Notizen, die ein Dozent während einer Vorlesung macht, elektronisch zur Verfügung gestellt zu bekommen?

Es ist besser, wenn ich es selber abschreibe.	37
Dies ist unnötig.	17
Dies ist interessant.	339

2. Wie wichtig ist Ihrer Meinung nach dabei, dass die Dozentennotizen direkt eine Verbindung in ein elektronisch vorhandenes Kursskript aufbauen?

Kein weiterer Vorteil.	18
Nettes Feature.	236
Erst dann macht es überhaupt Sinn.	137

3. Was halten Sie davon, als Student direkt elektronisch in ein elektronisches Kursskript Notizen einfügen zu können?

Es reicht, wenn die Dozentennotizen im Kursskript erscheinen.	110
Wenn schon ein elektronisches Skript, dann auch gleich alles elektronisch.	185
Nichts. Auch wenn das Kursskript elektronisch ist, arbeite ich bei meinen Notizen lieber mit Papier.	77
Nichts. Ich mache sowieso keine eigenen Notizen.	20

4. Wenn Sie elektronische Notizen in einem elektronischen Kursskript nur mit Netzwerkverbindung erstellen können, wäre dies

eine vertretbare Einschränkung, da sie zuhause sowieso nichts notieren.	27
eine vertretbare Einschränkung, da sie zuhause auch online sind.	269
eine unzumutbare Einschränkung.	93

5. Welche Arten von Notizen müssen zu einem elektronisch verfügbaren Kursskript MINDESTENS möglich sein, bevor sie ein solches System nutzen würden?

Texteingabe (Tastatur) am Rand des Kursskripts.	106
Texteingabe (Tastatur) beliebig im Kursskript.	116
Freihandlinien (Maus oder Stift) beliebig im Kursskript.	29
Freihandlinien (Maus oder Stift) und Texteingabe (Tastatur) beliebig im Kursskript.	136

=> bitte umblättern

6. Was halten Sie davon, Notizen anderer (z.B. auch Dozent) über das Netzwerk importieren zu können?

Uninteressant.	30
Von Dozent interessant, von anderen Studenten uninteressant.	113
Von Dozent uninteressant, von anderen Studenten interessant.	4
Von Dozenten und anderen Studenten interessant.	242

7. Wie wichtig ist Ihnen eine Zugriffsbeschränkung, wenn andere Personen Ihre Notizen einsehen/importieren können?

Jeder darf sehen, was ich notiere.	141
Alle Studenten ja, dem Dozenten lieber nicht	20
Der Dozent ja, die anderen Studenten lieber nicht.	7
Ich muß genau festlegen können, wer meine Notizen sehen darf.	221

8. Was wäre der größte Vorteil, wenn die Notizen elektronisch zur Verfügung stehen?

Ich kann in meinen eigenen Notizen elektronisch suchen.	150
Ich kann Notizen anderer „per Klick“ importieren.	54
Ich kann andere Notizen durchsuchen/betrachten.	146
Andere sehen meine Notizen und geben mir Feedback.	34

Ergänzungen in AnnotEZ für die Kopplung an das WOTAN-System:

Klasse WotanScriptStub

```
package de.tum.in.annotEZWotanServlet.stub;

import de.tum.in.annotEZ.scriptStub.ScriptInterface;
import de.tum.in.annotEZ.scriptStub.ScriptStub;

public class WotanScriptStub extends ScriptStub {

   public WotanScriptStub(ScriptInterface script) {
      super(script);
   }

   public String getScriptRepresentationType() {
      return "WOTAN";
   }
```

Klasse CcUI_Servlet

```
public class CcUI_Servlet_WOTAN extends CcUI_Servlet {
   public Object createMenu() {

public class Text_WOTAN_Servlet extends DsTool {

   public Text_WOTAN_Servlet(DsDesigner designer) {
      super(designer);

   }

   public byte[] getAnnotationBlob() {
      byte[] rueck=null;
      String text = (String) getScriptStub().getObject("text");

      if (text != null) {
         rueck = text.getBytes();
      }
      return rueck;
   }

   public Object doAnnotate() {
      String html ="";
      html += "<h2>"+ getDesigner().getAnnotationType()+"</h2>";

      html += "<form action=\"s\" method=\"post\">";
      html += "<input type=\"hidden\" name=\"op\"
               value=\"annotation_action\">";
      html += "<input type=\"hidden\" name=\"actioncode\"
               value=\"finish_annotation\">";
      html += "<input type=\"hidden\" name=\"id\" value=\"<taskid>\">";
```

```
      html += "<textarea name=\"text\" cols=\"50\"
               rows=\"5\"></textarea>";

      html += "<input type=\"submit\" name=\"Submit\" value=\"Submit\">";
      html += "<input type=\"button\" name=\"Cancel\" value=\"Cancel\"
               onClick=\"self.location.href='s?op=task&id=<taskid>'\">";
      html += "</form>";

      return html;
   }

   public boolean isModal() {
      return false;
   }
```

Klasse WOTAN_CrossReference:

```
public class WOTAN_CrossReference extends DsDesigner {

   public WOTAN_CrossReference(DsController dsController) {
      super(dsController);

   }

   public String getToolType() {
      return "Text";
   }
```

Klasse WOTAN_Headword:

```
public class WOTAN_Headword extends DsDesigner  {

   public WOTAN_Headword(DsController dsController) {
      super(dsController);

   }

   public String getToolType() {
      return "Text";
   }
}
```

Klasse PsPresenter:

```
public class PsPresenter_Headword_Text_Servlet extends PsPresenter {
   public Object getMinimizedImage( Annotation annotation) {

      String text;
      if (annotation.getContentSyntax().equals("Text")) {
```

```
            text = new String(annotation.getData());
        } else {
            text = "Annotationstyp entspricht nicht dem Anzeigetool.";
        }

        return text;
    }
```

Klasse PsSummary_Servlet_standalone:

```
public class PsSummary_Servlet_standalone extends PsSummary {
    private Vector<String> _icons;
    private String _summary;

    public PsSummary_Servlet_standalone() {
        _icons = new Vector<String>(0);
    }

    public void addAnnotation(PsPresenter presenter,
                              Annotation annotation) {
        String icon = (String) presenter.getMinimizedImage(annotation);
        _icons.add(icon);
    }

    public void clear() {
        _icons.clear();
        _icons = new Vector<String>(0);

        _summary = "<hr/<hr/>";
        _summary += "<p>No own annotations yet</p>";
    }

    public void refresh() {
        int n = _icons.size();

        _summary = "<hr/<hr/>";
        if (n == 0) {
            _summary += "<p>No own annotations yet</p>";
        } else {
            _summary += "<h3>Own annotations</h3>";
            _summary += "<hr/>";

            for (int i = 0; i < n; i++) {
                _summary += "<p>" + _icons.elementAt(i) + "</p>";
                _summary += "<hr/>";
            }
        }
    }

    public String getSummary() {
        return _summary;
    }
}
```

Klasse UserFromScript:

```
public class UserFromScript extends UsrManager {
    public UserFromScript( UsrController controller) {
        super(controller);
    }
    public String getUserID() {
        return getUserIDFromScriptStub();
    }
}
```

Klasse UserFilePersistenzManager:

```
public class UserFilePersistenzManager extends FilePersistenzManager {

    public UserFilePersistenzManager(PersistenzController controller) {
        super(controller);
    }

    private void setUserDirectory() {
        String sep = System.getProperty("file.separator");
        setHomeDirectory( sep +"temp"+sep+"wotan"+sep
                          +"db"+sep+getUserID()+sep);
    }

    public boolean existTable(String tablename) {
        setUserDirectory();
        return super.existTable(tablename);
    }

    public synchronized void createTable(String tablename,
                                         String[][] tablestruct) {
        setUserDirectory();
        super.createTable(tablename, tablestruct);
    }

    public synchronized String insert(String table, String[][] values) {
        setUserDirectory();
        return super.insert(table, values);
    }

    public synchronized Vector<String[]> select(String table, String field,
                                                String searchValue) {
        setUserDirectory();
        return super.select(table, field, searchValue);
    }
```