Fakultät für Informatik
der Technischen Universität München

Annotationen zur Verbesserung der Wiederverwendbarkeit von Lehrmaterialien

Weilun Zhuang

Fakultät für Informatik

Lehrstuhl XI
Angewandte Informatik / Kooperative Systeme

Annotationen zur Verbesserung der Wiederverwendbarkeit von Lehrmaterialien

Weilun Zhuang

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. Dr. Peter Paul Spies
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Johann Schlichter
	2.	Univ.-Prof. Bernd Brügge, Ph.D.

Die Dissertation wurde am 13.09.2005 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 13.02.2006 angenommen.

Zusammenfassung

Die gegenwärtige Forschung zur Wiederverwendbarkeit von Lehrmaterialien konzentriert sich darauf, die einzelnen Lernobjekte, aus denen Lehrmaterialien bestehen, möglichst präzise mittels Metadaten zu beschreiben, um basierend auf der Beschreibung eine möglichst gute Wiederverwendbarkeit der Lernobjekte zu erreichen. Dieses Vorgehen engt die Möglichkeiten der Beschreibung von Lernobjekten durch die formale Struktur von Metadaten stark ein. Ebenso werden Rahmenbedingungen, die beachtet werden müssen, wenn Lehrmaterialien kooperativ von mehreren Organisationen und Unternehmen erstellt werden, nicht berücksichtigt: Lehrmaterialien werden in einem zeitlich und räumlich verteilten, wenig organisierten Prozess erstellt. Die Prozesspartner sind lose gekoppelt; eine unmittelbare, persönliche Zusammenarbeit ist oft nicht leicht möglich. Ebenso wenig sind Vorgehensweisen zur Verbesserung der Qualität der Lernobjekte durch Einarbeitung von Feedback vorgesehen. All diese Aspekte haben Auswirkungen auf die Wiederverwendbarkeit der Lehrmaterialien.

In dieser Arbeit wird mit Annotationen ein Ansatz vorgestellt, mit dem die Wiederverwendbarkeit der Lehrmaterialien gezielt verbessert werden kann. Annotationen werden hierbei sowohl zur Optimierung des Lehrmaterialerstellungsprozesses hinsichtlich der Koordination der Prozessteilnehmer als auch als Mechanismus zur umfassenden Beschreibung der Lernobjekte eingesetzt.

Als erstes wird hierzu gezeigt, wie Annotationen eingesetzt werden können, um strukturelle Schwächen von Metadaten auszugleichen. Metadaten zeichnen sich durch einen hohen Grad an Formalisierung aus. Annotationen hingegen erlauben auch subjektive, nuancierte oder bewusst vage Aussagen zu Lernobjekten. Sie erlauben es außerdem, Metadaten von Lernobjekten zu kommentieren und zu werten. Diese zusätzlichen Informationen, die durch Annotationen geliefert werden, geben den Prozessteilnehmern wertvolle Informationen zur Verwendung der jeweiligen Lernobjekte.

Des weiteren wird beschrieben, wie Annotationen genutzt werden können, um die Koordination der Prozessteilnehmer bei der Lehrmaterialerstellung und bei der Nutzung von Lernobjekten zu verbessern. Der Schwerpunkt liegt hierbei auf der Kommunikation zwischen Nutzern von Lernobjekten, die Feedback die Nutzung der Lernobjekte betreffend liefern, und den Designern der Lernobjekte, die das Feedback umsetzen. Ziel dieser Kommunikation ist es, die Qualität der Lernobjekte sukzessive zu erhöhen und durch geeignete Modifikationen der Lernobjekte ihre möglichen Einsatzmöglichkeiten zu erweitern.

Um Annotationen einsetzen zu können, ist Unterstützung durch eine geeignete Infrastruktur nötig, die auf bestehende Lernplattformen aufgesetzt werden kann. In dieser Arbeit wird so eine Infrastruktur entworfen und prototypisch implementiert.

Dank

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftliche Angestellte am Lehrstuhl für Angewandte Informatik / Kooperative Systeme der Fakultät für Informatik an der TU München.

Mein bester Dank gilt Herrn Prof. Dr. Johann Schlichter, meinem Doktorvater, der mich mit vielen wertvollen Anregungen, Diskussionen und Kritik fachlich und persönlich während der letzten vier Jahre unterstützt hat. Bei meinem Zweitgutachter, Herrn Prof. Bernd Brügge, Ph.D., möchte ich mich besonders dafür bedanken, dass viele Aspekte des Software-Engineering in dieser Arbeit Berücksichtigung fanden.

Mein Dank richtet sich weiterhin an Herrn Dr. Jürgen Koch und Herrn Dr. Wolfgang Wörndl, die durch mehrfaches Korrekturlesen und viele Diskussionen zum Gelingen dieser Arbeit beigetragen haben.

Vielen Dank auch an meine Kollegen im Rahmen des Projekts Targeteam, Herrn Prof. Dr. Gunnar Teege (Universität der Bundeswehr München) und Herrn Dipl. Inform. Peter Breitling, für die gute Zusammenarbeit, und an alle meine Kollegen am Lehrstuhl für Angewandte Informatik / Kooperative Systeme für das beste Arbeitsklima. Insbesondere gilt mein Dank jedoch Frau Elfriede Bunke.

Herrn Dr. Jürgen Koch und Herrn Dipl. Ing. (FH) Karl Geigl danke ich dafür, dass sie diese Arbeit in der Endphase trotz enger Zeitplanung hinsichtlich der Sprache Korrektur gelesen haben.

Ein herzliches Dankeschön auch an meine Eltern, Sicheng Zhuang und Huizhu Zhang.

Und nicht zuletzt danke ich dem wichtigsten Menschen in meinem Leben – meiner Tante Miaozhen Zhang – dafür, dass Sie mir bei der Charakterbildung geholfen hat. Sie gab mir jene Perspektive, ohne die diese Arbeit nicht zustande gekommen wäre.

Inhalt

1 Einleitung

1.1 Hintergrund

Die Verbesserung der Wiederverwendbarkeit von Lehrmaterialien ist ein Forschungsgebiet, in dem bereits viel geleistet wurde: Es wurden mit Hilfe der KI (Künstliche Intelligenz) Konzepte entwickelt, die es erlauben, den Inhalt von Lehrmaterialien semantisch zu erfassen und zu katalogisieren (beispielsweise in [Har00]); es wurden Erfahrungen des Software-Engineering aufgegriffen und Komponentenmodelle für Lehrmaterialien entwickelt, die eine weitgehende Kombinierbarkeit der einzelnen Komponenten erlauben [Gri98]; es wurde eine Vielzahl von Möglichkeiten hergeleitet, um Lehrmaterialien bzw. Komponenten, aus denen Lehrmaterialien aufgebaut werden können, möglichst präzise zu beschreiben und so deren Wiederverwendung zu verbessern (siehe [SüFr01]). Zusätzlich wurden große Anstrengungen unternommen, um die Lehrmaterialien leichter zugänglich zu machen: Web-basierte Nutzerschnittstellen wurden vorgestellt, Portale für Lehrmaterialautoren, für Dozenten und für Lerner wurden aufgesetzt (siehe [Brü98]).

Diese und noch viele weitere, hier nicht genannte Ansätze haben die technischen Voraussetzungen geschaffen, um neue Lehrmaterialien aus bereits bestehenden Lehrmaterialien aufzubauen, also Lehrmaterialien wiederzuverwenden, um „das Rad nicht neu erfinden“ zu müssen. Alle diese Ansätze berücksichtigen jedoch nicht das Umfeld, in dem Lehrmaterialien entstehen: Die Entwicklung von Lehrmaterialien ist – völlig unabhängig davon, wie gut wiederverwendbar die Lehrmaterialien sind – eine Aufgabe, die nicht ein einzelner Autor erledigt, sondern dies ist eine Aufgabe, an der viele Menschen beteiligt sind: Hier sind Autoren von Lehrmaterialien, die zu neuen Lehrmaterialien kombiniert werden können, zu nennen, aber ebenso die Dozenten, die Lehrmaterialien für ihre Lehrveranstaltungen zusammenstellen, sowie die Lerner, die die Lehrmaterialien letztlich zur Wissensaneignung nutzen. Das Zusammenspiel all dieser Menschen ist hoch komplex, denn hier liegt keine sequenzielle Abarbeitung bestimmter, vorgegebener Schritte vor. Es wäre naiv anzunehmen, dass Lehrmaterialien einmal erzeugt und dann unverändert immer wieder eingesetzt und neu kombiniert werden. Statt dessen unterliegen sie einem ständigen Wandel, der neuen Erkenntnissen (oder auch der Entdeckung von Fehlern oder Unklarheiten) Rechnung trägt.

Die Situation wird zusätzlich dadurch verkompliziert, dass die Menschen, die die Lehrmaterialien erstellen und nutzen, sich nicht notwendigerweise kennen. Sie können außerdem zu unterschiedlichen Zeiten und an unterschiedlichen Orten mit den Lehrmaterialien arbeiten. Dies hat Auswirkungen auf den gesamten Ablauf der Lehrmaterialerstellung: Wie können in

so einer Situation die Anforderungen des einen Autors oder Dozenten an bestimmte Lehrmaterialien – auch wenn sie ideal wiederverwendbar sind – überhaupt berücksichtigt werden? Wie können in so einer Situation Fehler in Lehrmaterialien an die verantwortlichen Autoren zurückgemeldet werden? Antworten auf diese Fragen sind essenziell, wenn qualitativ hochwertige – also gut wiederverwendbare, aber auch fachlich korrekte und didaktisch gut einsetzbare – Lehrmaterialien erstellt werden sollen.

Diese Arbeit liefert einen Vorschlag, wie eine Antwort auf diese Fragen aussehen könnte. Ein wesentlicher Aspekt dieser Arbeit ist die integrierte Betrachtung von Wiederverwendung auf verschiedenen Ebenen: Wiederverwendung auf der technischen Ebene mit dem Schwerpunkt der Modellierung von Lehrmaterialien. Hierbei wird auf bereits vorgestellten Konzepten zur Verbesserung der Wiederverwendung von Lehrmaterialien aufgesetzt, denn schließlich sollen die bereits bewährten Ergebnisse anderer Arbeiten genutzt werden. Ein weiterer Punkt ist die Wiederverwendung auf der Prozessebene, also die Betrachtung, wie Lehrmaterialien entstehen und genutzt werden, und welche Besonderheiten hier berücksichtigt werden müssen.

1.2 Motivation und Ziel dieser Arbeit

1.2.1 Motivation

Wiederverwendung von Lehrmaterialien: Sicht auf das Produkt

Lehrmaterialien – speziell zu Themen der IT – liegen in großer, ständig steigender Zahl vor. Zudem unterliegt gerade das Fachwissen der IT einem schnellen und ständigen Wandel, wodurch auch die diese Themen behandelnden Lehrmaterialien ständig angepasst werden müssen. Eine Möglichkeit, dieser Herausforderung zu begegnen, liegt darin, die Wiederverwendbarkeit der Lehrmaterialien zu erhöhen, um so schneller auf Änderungen in den Sichtweisen, Paradigmen, Konzepten und Techniken reagieren zu können.

In der Tat ist die Erhöhung der Wiederverwendung von Lehrmaterialien ein Schwerpunkt der heutigen Forschung für eine effiziente Erstellung und Nutzung von Lehrmaterialien. Aus Zeit- und Kostengründen ist es erforderlich, dass die alten Lehrmaterialien weiterentwickelt, gegeneinander ausgetauscht oder neu konfiguriert und auf andere Weise eingesetzt werden.

Um die Wiederverwendung komplexer und meist unzureichend geordneter Lehrmaterialien zu ermöglichen, werden sie geeignet modularisiert und beschrieben (siehe [SüFr01]). Metadatenbasierte Ansätze unterstützen diese Vorgehensweise ([ARIADNE], [IMS a], [LOM]). Leider decken die heute eingesetzten Metadatenstandards für Lehrmaterialien nur relativ allgemeine Aspekte und längst nicht alle Eigenschaften von Lehrmaterialien sowie deren Nutzungsarten ab. Vor allem sind manche Eigenschaften, wie zum Beispiel der „Schwierigkeitsgrad“ von Lehrmaterialien, so subjektiv und so abhängig von der konkreten Nutzung, dass sie durch Metadaten in der von den Standards geforderten Form nicht treffend beschrieben werden können; eine Beschreibung entsprechend den Standards würde praktisch zu einem Verlust an Qualität der Beschreibung und damit einem Verlust an praktischer Nutzbarkeit führen.

Wiederverwendung von Lehrmaterialien: Sicht auf den Erstellungs- und Nutzungsprozess

Die Erstellung und Nutzung von Lehrmaterialien verläuft heute meist isoliert; Lehrmaterialien anderer Autoren werden bislang nur sehr selten genutzt. Insbesondere Lehrmaterialien anderer Organisationen und Unternehmen werden bislang kaum wiederverwendet. Dies hat unterschiedliche Gründe: Neben rechtlichen Fragen, beispielsweise hinsichtlich des Urheber-

rechts, sind hier auch unterschiedliche, zueinander inkompatible Formate und nicht aufeinander abgestimmte Terminologien zu nennen. So wird das Potential, das eine Wiederverwendung der Lehrmaterialien bietet, nur ungenügend genutzt.

Bei genauer Betrachtung ist der Lehrmaterialerstellungs- und -nutzungsprozess nicht ein zentralisierter, einmaliger Prozess, sondern ein zeitlich und räumlich verteilter, iterativer und mit Rückkopplungen versehener Prozess: Einmal erstellte Lehrmaterialien können – sofern sie geeignet gestaltet und beschrieben sind – bei der Erstellung anderer Lehrmaterialien erneut verwendet werden, und Rückmeldungen von den Nutzern der Lehrmaterialien erlauben es, gezielt Verbesserungen an den Lehrmaterialien vorzunehmen, um so deren Qualität zu erhöhen. Dabei sollte es keine Rolle spielen, von welcher Organisation oder von welchem Unternehmen die Lehrmaterialien bereitgestellt werden. Diese Aspekte zur Prozessoptimierung werden bislang nur wenig betrachtet.

1.2.2 Ziel dieser Arbeit

Gegenstand dieser Arbeit ist die Wiederverwendung digitaler Lehrmaterialien „im Großen“. Dies bedeutet die Wiederverwendung von Lehrmaterialien, die in digitaler Form vorliegen, zwischen Menschen und über Organisations- und Unternehmensgrenzen hinweg. Die Wiederverwendung „im Kleinen“, also Mechanismen, mit deren Hilfe ein einzelner Lehrmaterialautor seine eigenen Lehrmaterialien besser wiederverwenden kann, liefert wertvolle Erkenntnisse und Methoden, ist aber nicht ohne weiteres auf die Wiederverwendung im Großen übertragbar.

Um eine weitgehende Wiederverwendbarkeit von digitalen Lehrmaterialien über Unternehmens- und Organisationsgrenzen hinweg zu ermöglichen, müssen zwei Aspekte betrachtet werden: Es wird ein allgemein anwendbares Konzept benötigt, um Lehrmaterialien von verschiedenen Nutzern (Autoren, Dozenten und Lerner) für verschiedene Arten der Nutzung einsetzbar zu machen. Eine Integration verschiedener Lehrmaterialien – egal ob neu erstellt oder bereits existierend – muss möglich sein. Des weiteren ist es nötig, den Prozess der Erstellung und Nutzung von Lehrmaterialien effizienter zu gestalten. Dies betrifft eine verbesserte Koordination der einzelnen Prozessteilnehmer (Autoren, Dozenten und Lerner) untereinander, aber auch die Möglichkeit, Feedback zu Lehrmaterialien zu geben und in den Prozess zu integrieren.

Diese Arbeit fördert die Wiederverwendung der Lehrmaterialien durch eine gezielte Erweiterung der bestehenden Konzepte und Ansätze, indem die Lehrmaterialien modell- und prozesstechnisch besser integriert werden und somit eine Wiederverwendung in einem durchgängigen Prozess zwischen Autoren, Lehrern und Lernern stattfinden kann. Die in diesem Prozess entstehenden Lehrmaterialien sollen zusätzlich durch den iterativen Charakter des Prozesses eine steigende Qualität aufweisen.

Um dieses Ziel zu erreichen, werden in dieser Arbeit domänen- und anwendungsspezifische Konzepte sowohl im Lehrmaterialmodell als auch im Prozess der Lehrmaterialerstellung und –nutzung integriert. Mittels Annotationen und Metadaten wird ein transparenter und nachvollziehbarer Austausch der Produkt- und Prozessinformationen realisiert und so eine durchgängige Wiederverwendung und Qualitätssicherung der Lehrmaterialien schrittweise erreicht. Der Fokus dieser Arbeit liegt auf Annotationen, mit denen die Wiederverwendbarkeit der Lehrmaterialien gezielt erhöht werden kann. Annotationen werden hierbei sowohl zur Optimierung des Lehrmaterialerstellungsprozesses hinsichtlich der Koordination der Prozessteilnehmer als auch (neben Metadaten) als Mechanismus zur umfassenden Beschreibung der Lernobjekte, aus denen die Lehrmaterialien aufgebaut sind, eingesetzt. Beide Maßnahmen in Kombination erlauben es, einmal erstellte Lernobjekte und Lehrmaterialien leichter

wiederzuverwenden. Um Annotationen einsetzen zu können, ist Unterstützung durch eine geeignete Annotationsinfrastruktur nötig, die auf bestehende Lernplattformen aufgesetzt werden kann.

Der in dieser Arbeit hergeleitete Lösungsvorschlag ist für die Nutzer einfach nachzuvollziehen und zu bedienen und kann für eine beliebige Lernplattform eingesetzt werden. Änderungen an bereits bestehenden Lehrmaterialien oder bereits bestehenden Lernplattformen sind nicht erforderlich. (Es kann jedoch nötig sein, die Lehrmaterialien mit Beschreibungen zu versehen oder bereits vorhandene Beschreibungen zu ergänzen.)

1.3 Verwandte Themengebiete

Das Thema „Verbesserung der Wiederverwendung von Lehrmaterialien“ ist sehr umfangreich; Aspekte des den Lehrmaterialien zu Grunde liegenden Datenmodells sind ebenso zu betrachten wie Aspekte der Prozessoptimierung, aber auch Aspekte der Kommunikation und Koordination von räumlich und zeitlich verteilten Prozessteilnehmern (Lehrmaterialautoren und Lehrmaterialnutzer). Insbesondere Erkenntnisse der Fachgebiete Wissensmanagement, Software-Engineering und Computer Supported Cooperative Work (CSCW) werden in dieser Arbeit genutzt, vgl. Abbildung 1-1. Um einen tragfähigen Vorschlag zur Verbesserung der Wiederverwendung von Lehrmaterialien entwickeln zu können, müssen daher auch die oben genannten Fachgebiete berücksichtigt werden.

Abbildung 1-1: Angrenzende Themen

Im folgenden werden die Fachgebiete kurz vorgestellt und es wird beschrieben, welchen Einfluss sie auf diese Arbeit haben.

1.3.1 Wissensmanagement

Die Kontextualisierung von Lehrmaterialien erfordert Ansätze aus dem Bereich Wissensmanagement. Die Grundlage von Wissensmanagement ist der Begriff „Wissen“. Es gibt viele verschiedene Definitionen von „Wissen“. Eine häufig verwendete Definition besagt: „Wissen bezeichnet eine bedeutungsvolle Vernetzung von Informationen.“ [RMS00]. Generelle Aufgabe des Wissensmanagements ist es, die Erzeugung, Bewahrung und Nutzung von Wissen möglich zu machen und darauf aufbauend die Prozesse zur Wissensgenerierung und Wissensnutzung effizienter zu gestalten.

Wissensmanagement beschäftigt sich vor allem mit der Verwaltung von Wissen, wofür insbesondere Data-Warehouses, Knowledge-Discovery, Data-Mining, Knowledge-Maps, usw. eingesetzt werden. Content Management betrachtet nun den Teil des Wissensmanagements, der sich mit dem Handling einer großen Menge von Informationen befasst. Das Content Management ist damit das grundlegende Werkzeug zum Generieren, Einspielen, Verteilen und Nutzen von Informationen, worauf alle weiteren Methoden des Wissensmanagements aufsetzen.

Der Fokus von Wissensmanagement liegt ferner auf dem Auffinden und Abgleichen von abgeleiteter Information aus und mit wirklich eingetroffener, realer Information. Wissensmanagement will somit eine Organisation der Daten schaffen, die so nahe an der Umwelt wie möglich ist. Des weiteren sollen neu eingehende Informationen unter dem Hintergrund der bereits erfassten und modellierten Daten interpretiert werden, und es sollen aus dem beste-

henden Datenbestand heraus neue Informationen generiert werden. Im Umfeld von E-Learning werden diesbezüglich der Inhalt von Lehrmaterialien im Kontext ihrer Domäne und Nutzung definiert und spezifiziert. So soll sichergestellt werden, dass die Lehrmaterialien in entsprechender Qualität erstellt werden und dass eine dem Inhalt der Lehrmaterialien passende Nutzung stattfindet.

Zum Thema „Wissensmanagement“ ist umfangreiche Literatur verfügbar. Der interessierte Leser sei beispielsweise auf [NoTa95], [Mai04] oder [Inm96] verwiesen.

1.3.2 Wiederverwendung im Software-Engineering

Auch das Gebiet des Software-Engineering beeinflusst diese Arbeit. Die Wiederverwendung von Software hat in den letzten Jahren besondere Bedeutung in der Software-Entwicklung erlangt. Dies liegt einerseits in der durch die Wiederverwendung von Software-Komponenten erreichbaren deutlichen Verbesserung der Produktivität in der Software-Entwicklung begründet und andererseits in der höheren Qualität der entstehenden Software-Produkte. Dies – höhere Produktqualität und höhere Produktivität bei der Produkterstellung – sind Ziele, die auch für die Erstellung von Lehrmaterialien erstrebenswert sind.

Die Wiederverwendung von Software-Komponenten ist zentrales Element im Entwicklungszyklus komplexer Softwaresysteme. Angrenzende, in diesem Zusammenhang ebenfalls interessante Bereiche sind Domain Modeling, Reverse Engineering oder heutige Komponentenmodelle.

Aktuelle Software-Entwicklungsumgebungen, Klassenbibliotheken, Komponententechnologien (beispielsweise Enterprise Java Beans, Microsoft .Net, etc.) sowie diverse Enterprise Reference Architectures bieten Wege, Software wiederzuverwenden. Dabei ist es aber nicht nur der Quellcode, der wiederverwendet werden soll, sondern darüber hinaus auch Konzepte und Software-Architekturen. Solche Architektur- und Design-Lösungen sind in Forschung und Entwicklung gleichermaßen anerkannt und zum Teil bereits gut dokumentiert (beispielsweise Architekturmuster, Entwurfsmuster etc.). Zum Thema „Wiederverwendung im Software-Engineering“ sei auf [BrDu04], [Gri98] und [Bal98] verwiesen.

Im Umfeld des E-Learning können einige Ideen zur Wiederverwendung von Software-Komponenten übernommen werden, beispielsweise für die Inhaltsverwaltung von Lehrmaterialien. Allerdings müssen zur kompletten Modellierung von Lehrmaterialien sowie deren Erstellung und Nutzung zusätzlich menschliche und didaktische Aspekte (Nutzungsarten, Interaktionsverhalten) bei den Analyse- und Designphasen berücksichtigt werden; diese Aspekte werden vom Software-Engineering aufgrund einer anderen Zielrichtung nicht abgedeckt.

1.3.3 Computer Supported Cooperative Work (CSCW)

Werden Lehrmaterialien von mehreren Autoren gemeinsam erzeugt, die eventuell noch zeitlich und räumlich verteilt arbeiten, müssen Probleme der Koordination und der Kommunikation zwischen den Autoren gelöst werden. Eine ähnliche Situation tritt ein, wenn die Nutzer von Lehrmaterialien Kontakt zu den Autoren aufnehmen wollen. Die Lösung dieser Probleme ist eine der zentralen Aufgaben der CSCW-Forschung: Eine effiziente Unterstützung der Kommunikation, Koordination und auch Kooperation zwischen Mitgliedern räumlich und / oder zeitlich verteilter Gruppen von Menschen. Falls die Zusammenarbeit der Autoren und eventuell auch der Nutzer eher lockeren Charakter hat, müssen ebenso soziale Mechanismen berücksichtigt werden, die weniger in Gruppen, aber umso mehr in (virtuellen) Communities wirken. Dies betrifft sowohl die Kommunikation zwischen Autoren und Endnutzern der Lehrmaterialien mit dem Zweck, Unklarheiten zu lösen oder Fehler zu korrigieren, als auch die Kommunikation zwischen den Autoren zum Zwecke der Koordination des Lehrmaterial-

erstellungsprozesses. Kommunikation ist hierbei ein Werkzeug sowohl zur Erreichung einer höheren Qualität der Lehrmaterialien als auch zur Koordination der Arbeitsschritte, die zur Erstellung der Lehrmaterialien nötig sind.

Das Feld der CSCW- und Community-Forschung ist sehr weit und kann hier nur überblicksartig vorgestellt werden. Dem interessierten Leser sei hier [BoSc00] oder [TSMB95] ans Herz gelegt.

1.4 Aufbau dieser Arbeit

1.4.1 Überblick

Ziel dieser Arbeit ist es, wie bereits in 1.2.2 beschrieben, eine Möglichkeit aufzuzeigen, wie die Wiederverwendung von Lehrmaterialien erleichtert werden kann. Dies wird mittels dreier Maßnahmen erreicht, die für sich allein genommen keine ausreichende Lösung bieten, in Kombination jedoch neue Möglichkeiten aufzeigen:

Es muss ein Datenmodell bereitgestellt werden, das es erlaubt, neue und bestehende Lehrmaterialien zu kapseln, um einheitliche, für alle Lehrmaterialformen und Lehrmaterialformate anwendbare Beschreibungen sowie Schnittstellen zum Zugriff zu bieten.

Es muss der Prozess der Lehrmaterialerstellung und –nutzung beschrieben, kritisch betrachtet und gezielt verbessert werden. Dabei muss es auf Basis des Datenmodells möglich sein, neue Lehrmaterialien auf einfache Weise zu erstellen sowie bereits bestehende Lehrmaterialien wiederzuverwenden und in den Prozess nahtlos einzugliedern. Durch die so optimierten Abläufe wird die Qualität der entstehenden Lehrmaterialien verbessert und die Geschwindigkeit der Lehrmaterialerstellung erhöht.

Dies geht einher mit der Bereitstellung eines Kommunikations- und Koordinationsmechanismus, der auch in lose gekoppelten Gemeinschaften von Autoren und Nutzern von Lehrmaterialien eingesetzt werden kann. Annotationen bieten eine effiziente Implementierung dieses Kommunikations- und Koordinationsmechanismus. Eine Voraussetzung für den Einsatz von Annotationen ist, dass das Datenmodell eine Annotation aller Teile von Lehrmaterialien erlauben muss: Sowohl ihr Inhalt als auch ihre Beschreibung müssen annotierbar sein.

Diese Forderungen spiegeln sich auch im Aufbau dieser Arbeit wider, vgl. Abbildung 1-2. Kapitel 2 legt die Grundlagen, indem wichtige Begriffe und Ideen zu den Themen Lehrmaterialien und Wiederverwendung vorgestellt werden, die sich durch die ganze Arbeit ziehen. Die Konzepte werden, wo sinnvoll, durch die Beschreibung bereits implementierter Systeme illustriert. Kapitel 3, Kapitel 4 und Kapitel 5 beschreiben die drei oben genannten Maßnahmen (Herleitung eines Datenmodells, Prozess der Lehrmaterialerstellung und –nutzung, Bereitstellung von Annotationsmechanismen) im Detail. In den Kapiteln 6 und 7 schließlich wird aufbauend auf den Ergebnissen der Kapitel 3, 4 und 5 eine Architektur vorgestellt, die die drei oben genannten Maßnahmen integriert und so zu einer Verbesserung der Wiederverwendung von Lehrmaterialien beiträgt. Die Architektur wurde prototypisch implementiert; hierauf wird insbesondere in Kapitel 7 eingegangen.

1.4.2 Beschreibung der Kapitel

In Kapitel 2 wird anhand von Definitionen und Szenarien geklärt, was in dieser Arbeit unter Lehrmaterialien verstanden wird und welche Eigenschaften Lehrmaterialien haben. Anschließend werden ausgewählte konzeptionelle Ansätze und Mechanismen zur Erreichung und Unterstützung von Wiederverwendbarkeit vorgestellt: Aus konzeptioneller Sicht bieten

sich Objektorientierung, Entwurfmuster sowie komponenten- und Framework-basierte Konzepte der Wiederverwendung an. Aus technischer Sicht wird heute gern der Ansatz der Metadaten zur Definition, Konkretisierung und Spezifikation von Lehrmaterialien genutzt. Auch Einschränkungen der Metadaten (Constraints) und Transformationsregeln finden Anwendung. Anschließend wird beschrieben, wie die Wiederverwendung von Lehrmaterialien umgesetzt werden kann und welche Überlegungen dabei nötig sind. Diese Vorgehensweise wird in verschiedenen Szenarien illustriert. In diesem Zusammenhang werden auch verschiedene Standards und Systeme vorgestellt. Abgeschlossen wird Kapitel 2 mit einer kritischen Betrachtung der vorgestellten Ansätze und Standards und einer vertieften Motivation für diese Arbeit.

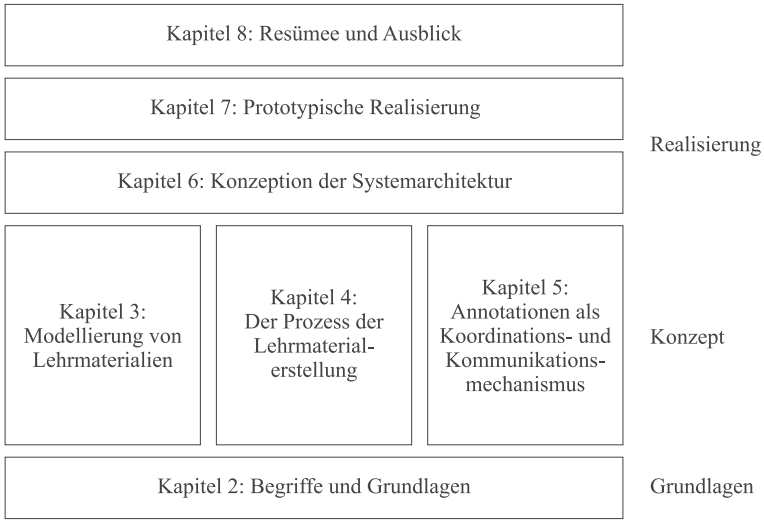


Abbildung 1-2: Aufbau dieser Arbeit

In Kapitel 3 wird ein konzeptionelles Datenmodell beschrieben, das es erlaubt, beliebige, neu erstellte sowie bereits existierende Lehrmaterialien mit einer gemeinsamen Schnittstelle zu versehen. Diese Schnittstelle stellt einerseits einen einheitlichen Satz an Zugriffsmöglichkeiten bereit und bietet andererseits eine für alle Lehrmaterialien genormte Beschreibung. Das Datenmodell macht bewusst keine Annahmen hinsichtlich des internen Aufbaus und des Datenformats der Lehrmaterialien; der Zugriff erfolgt ausschließlich über die oben genannte gemeinsame Schnittstelle, die sich wie ein Mantel um die Lehrmaterialien legt. Lehrmaterialien völlig verschiedenen Formats können auf diese Weise einheitlich genutzt werden. So werden die technischen Voraussetzung für eine Wiederverwendung der Lehrmaterialien geschaffen. Zur Realisierung werden Konzepte wie Typisierung, Parametrisierung und Constraint-Bildung auf Lehrmaterialien betrachtet. Dadurch erlangt man eine modelltechnische Integration und es entsteht eine einheitliche, gemeinsame Datenbasis wiederverwendbarer Lehrmaterialien, die während der gesamten Erstellung und Nutzung der Lehrmaterialien verwendet werden kann. Grundlage der Beschreibungen der Lehrmaterialien sind Metadaten. Es wird gezeigt, wie Parameter und Constraints mittels Metadaten realisiert werden können und welche konzeptbedingten Nachteile dabei in Kauf genommen werden müssen.

In Kapitel 4 wird ein generischer Erstellungs- und Nutzungsprozess für Lehrmaterialien hergeleitet. Ziel dieses Kapitels ist es, einen allgemeinen Prozess zu beschreiben, in dem sich ein großer Teil der derzeit praktizierten Zusammenarbeitsmodelle bei der Erstellung und Nutzung von Lehrmaterialien wiederfinden. So lassen sich Aussagen und Verbesserungsvorschläge zu diesen Prozess leicht auf andere, konkret angewandte Prozesse übertragen. Bei der Modellierung des generischen Prozesses werden Rollen, Aktivitäten sowie der Daten-

fluss genau beschrieben. Grundlage hierbei ist das in Kapitel 3 beschriebene Datenmodell. Aufgrund der Heterogenität der Input- / Outputdaten und der großen Zahl der über mehrere Organisationen und Unternehmen verteilten Rollen ist eine enge Koordination der Prozessschritte unumgänglich, um einen effizienten Prozessdurchlauf leisten zu können.

In Kapitel 5 werden Annotationen als der Mechanismus vorgestellt, der die enge Koordination zwischen den Prozessbeteiligten herstellen kann. Außerdem wird gezeigt, wie Annotationen konzeptionelle Schwächen des auf Metadaten aufbauenden Datenmodells überwinden können. Kapitel 5 ist insofern von zentraler Bedeutung, da hier aufgezeigt wird, wie mittels Annotationen sowohl Schwächen des Prozesses als auch Unzulänglichkeiten des Datenmodells elegant gelöst werden können. Hierzu wird folgendermaßen vorgegangen: Um ein Gefühl für den sinnvollen Einsatz von Annotationen zu bekommen, werden sie in diesem Kapitel erst konzeptionell vorgestellt und entsprechend verschiedener Ebenen (Formalisierungsgrad, Nutzung, etc.) klassifiziert. Anschließend werden sie in das in Kapitel 3 vorgestellte Datenmodell integriert und es wird gezeigt, wie der in Kapitel 4 vorgestellte Prozess erweitert werden kann, um mittels Annotationen eine bessere Koordination der Prozesspartner und damit sowohl eine beschleunigte Prozessdurchführung als auch eine bessere Qualität der Lehrmaterialien zu erreichen.

Annotationen bieten somit eine Querschnittsfunktionalität, die sowohl im Datenmodell als auch im Prozess eine wichtige Rolle spielt. In Kapitel 6 wird das Konzept einer Infrastruktur vorgestellt, die eine Unterstützung des Datenmodells sowie des Prozesses durch Annotationen erlaubt. In einem ersten Schritt werden allgemeine Anforderungen an diese Annotationsinfrastruktur genannt und es wird der Querschnittscharakter der Funktionalität, die durch Annotationen bereitgestellt wird, durch eine Einordnung der Annotationsinfrastruktur in gängige Architekturen von Lernplattformen herausgestellt. Anschließend wird die Annotationsinfrastruktur konzeptionell beschrieben, wobei sowohl auf Aspekte der Datenhaltung als auch auf Aspekte der Awareness und der Kommunikation zwischen den Prozessteilnehmern eingegangen wird.

In Kapitel 7 schließlich wird die prototypische Implementierung dieser Architektur beschrieben. Hierbei wird insbesondere der interne Aufbau der in Kapitel 6 genannten Komponenten der Annotationsinfrastruktur aufgezeigt und es wird beschrieben, wie diese Komponenten umgesetzt wurden. Wird in Kapitel 6 Wert auf das Konzept, das hinter der Annotationsinfrastruktur steht, gelegt, so konzentriert sich Kapitel 7 auf die Umsetzung des Konzeptes.

In Kapitel 8 wird schließlich ein Resümee dieser Arbeit gezogen: Es wird beschrieben, was die in dieser Arbeit hergeleitete Annotationsinfrastruktur leistet, welche organisatorischen Rahmenbedingungen bestehen müssen, damit die Annotationsinfrastruktur gewinnbringend eingesetzt werden kann, und wo ihre Grenzen liegen. Des weiteren werden offene Fragen angesprochen, die diese Arbeit aufgeworfen hat, die aber nicht in dieser Arbeit beantwortet werden können, und es werden nächste Schritte angeregt, die in nachfolgenden Arbeiten angegangen werden müssen.

2 Grundlagen und Umfeld von Lehrmaterialien

2.1 Überblick

Ziel dieses Kapitels ist es, die Grundlagen für diese Arbeit zu legen. Es werden häufig verwendete Begriffe im Zusammenhang mit Lehrmaterialien eingeführt und gegeneinander abgegrenzt. Des weiteren werden allgemeine Konzepte zur Unterstützung der Wiederverwendbarkeit, beispielsweise aus dem Software-Engineering, vorgestellt. Diese zwei Linien – Lehrmaterialien und Konzepte zur Unterstützung der Wiederverwendbarkeit – werden anschließend zusammengeführt, indem Wiederverwendbarkeit insbesondere bei Lehrmaterialien untersucht wird.

Zuerst wird in Abschnitt 2.2 der Begriff der Lehrmaterialien vorgestellt und aufbauend darauf weitere, in der Literatur oft genutzte Begriffe eingeführt und gegeneinander abgegrenzt. Dieser Abschnitt bildet die erste Linie, auf der diese Arbeit aufbaut: Die Linie der Lehrmaterialien.

Anschließend werden in Abschnitt 2.3 allgemeine Konzepte und Mechanismen zur Unterstützung von Wiederverwendbarkeit präsentiert. Hierbei wird neben Metadaten und Transformationsregeln auch auf strukturelle Ansätze wie Objektorientierung und Frameworks eingegangen. Dieser Abschnitt bildet die zweite Linie, auf der diese Arbeit aufbaut: Die Linie der Wiederverwendung.

Beide Linien – Lehrmaterialien und Wiederverwendung – werden in Abschnitt 2.4 zusammengeführt. In diesem Abschnitt wird untersucht, wie sich die in 2.3 beschriebenen Konzepte anwenden lassen, um die Wiederverwendung von Lehrmaterialien zu ermöglichen, und es werden Systeme beschrieben, in denen die einen oder anderen Ansätze zum Einsatz kommen. In 2.4.1 werden Anforderungen genannt, die wiederverwendbare Lehrmaterialien erfüllen müssen, und es wird in 2.4.2 vorgestellt, wie domänenspezifische und domänenübergreifende Lehrmaterialien geplant werden müssen.

In 2.5 schließlich wird anhand ausgewählter, heute üblicher Mechanismen – Metadaten und Transformationsregeln – gezeigt, wie die Lehrmaterialien dann anschließend realisiert werden können. Dies wird anhand verschiedener Szenarien in 2.6 vertieft.

Den Abschluss bildet mit Abschnitt 2.7 eine Bewertung der vorgestellten Systeme und Konzepte. Aus dem Ergebnis der Bewertung wird die Motivation für diese Arbeit abgeleitet.

2.2 Begriffsbestimmung

2.2.1 Definition

Der Begriff „Lehrmaterialien“ wird in der Literatur und in der Forschung auf vielerlei Weise verstanden: Lehrmaterialien sind Lehrmittel, welche Lehrbücher, Arbeitsbücher, Kopiervorlagen oder Medienverbünde umfassen können. Sie beziehen auch Wissen aus heterogenen Informationsquellen, das in Lexika, Sachbüchern, etc. organisiert ist, mit ein (vgl. beispielsweise [Wiki a]).

In dieser Arbeit wird der Begriff „Lehrmaterialien“ ähnlich der Definition des IEEE Learning Technology Standards Committee (IEEE LTSC) verstanden. IEEE LTSC definiert Lehrmaterialien folgendermaßen [LTSC]:

> *„Any entity, digital or non-digital, which can be used, reused or referenced during technology supported learning.”*

Im Unterschied zu dieser Definition werden in dieser Arbeit jedoch ausschließlich Lehrmaterialien in digitaler Form betrachtet, die mit Hilfe von Rechnern modifizierbar sind. Nicht-digitale Lehrmaterialien werden in dieser Arbeit nicht berücksichtigt.

2.2.2 Eigenschaften und Funktionalitäten von Lehrmaterialien

Lehrmaterialien sind komplexe Objekte. Sie haben folgende Eigenschaften (Abbildung 2-1):

- Inhalt

  Lehrmaterialien gehören immer zu einem bestimmten Themengebiet. Hierbei muss man allerdings auf die Benennung der Themengebiete achten: Lehrmaterialien können gleich bezeichnet sein, aber zu unterschiedlichen Themengebieten gehören (homonyme Benennung der Themengebiete). Tauchen zum Beispiel Lehrmaterialien über „Prozessmodellierung“ sowohl im Themengebiet „Maschinenbau“ als auch im Themengebiet „Informationsverarbeitung“ auf, so existieren in der Folge Lehrmaterialien mit jeweils unterschiedlichem Inhalt zum selben Thema „Prozessmodellierung“.

  Lehrmaterialien können ebenso unterschiedlich bezeichnet sein, aber das gleiche Themengebiet betreffen (synonyme Beschreibung der Themengebiete): So könnte es beispielsweise Lehrmaterialien über „Prozessmodellierung“ und über „Ablaufmodellierung“ geben, die weitgehend identische Inhalte haben.

- Syntaktische und semantische Struktur

  Der Begriff „Struktur“ umfasst bei Lehrmaterialien sowohl syntaktische als auch semantische Aspekte. Die Syntax ist durch die verschiedenen Datentypen der Lehrmaterialien festgelegt: Graphen, Listen, Tabellen und ebenso verschiedene Medienobjekte wie Bilder, Animationen, etc. haben einen bestimmten syntaktischen Aufbau[1]. Aus diesen Datentypen und / oder Medienobjekten werden rekursiv ein Satz, ein Absatz oder ein Kapitel aufgebaut. Durch die Verwendung von Medienobjekten verschiedener Formate eröffnen sich neue Chancen für die Nutzung von Lehrmaterialien, aber zugleich erhöht sich auch die Komplexität bei deren Erstellung.

[1] Der syntaktische Aufbau kann hierbei beispielsweise durch das verwendete Dateiformat (ASCII-Text, JPEG-Bild, MPEG-Video, etc.) definiert sein.

Des weiteren können Lehrmaterialien auch semantische Bedeutungen zugeordnet werden. Beispielsweise können einem Satz die Bedeutungen „Stichwort“ oder „Querverweis“ zugeordnet werden, „Definition“ oder „Illustration“ einem Absatz und „Einführung“, „Zusammenfassung“ oder „Glossar“ einem Kapitel. Durch die Zuordnung einer Semantik entsteht implizit eine inhaltliche und semantische Strukturierung der Lehrmaterialien. So können Lehrmaterialien inhaltlich miteinander kombiniert werden, um komplexe Sachverhalte darzustellen und einen Kurs aufzubauen.

- Präsentationsformat

  Lehrmaterialien können in verschiedenen Präsentationsformaten angeboten werden. Präsentationsformate umfassen beispielsweise das traditionelle Buch, aber auch elektronische Formate wie beispielsweise Hypertexte, PDF-Dokumente oder Sätze von Microsoft Powerpoint-Folien. In dieser Arbeit werden nur elektronische Formate betrachtet.

- Navigation und Darstellung

  Unter Darstellung versteht man die Visualisierung und das Layout der Lehrmaterialien. Die Darstellung legt die Gestaltung des Seitenaufbaus fest, also Schriftfarbe und –art, GUI-Elemente, etc.

  Analog kann für Lehrmaterialien auch ein Navigationsstil definiert werden, also die Art und Weise, wie die Navigation durch die Lehrmaterialien erfolgt. Beispiele hierfür sind Menüs, Navigationsleisten, Hypertexte, die als interaktive Bäume gestaltet sind, Sequenzen von einzelnen Lehreinheiten, etc.

  Hierbei ist zu beachten, dass die Darstellung bzw. die Navigation vom jeweiligen Präsentationsformat abhängig ist.

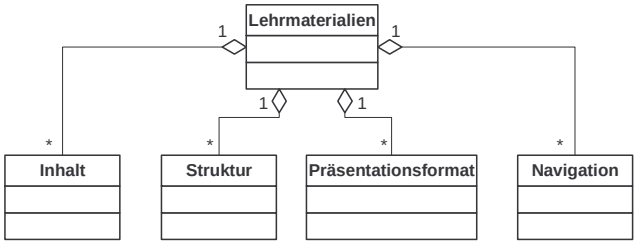


Abbildung 2-1: Eigenschaften von Lehrmaterialien

2.2.3 Granularität von Lehrmaterialien: Lernobjekte

Lehrmaterialien sind sehr umfangreiche und komplexe Gebilde. Um sie besser fassen und die Komplexität in der Betrachtung reduzieren zu können, werden sie in kleinere Objekte aufgeteilt. Die so entstehenden Objekte werden in der Literatur oft als *Lernobjekte* (*learning objects*) bezeichnet (vgl. [LTSC], [Hod00], [Hof02]). Sie sind inhaltlich eigenständig und in ihrer Struktur so beschaffen, dass sie miteinander kombiniert werden können.

Die *Granularität* der Lernobjekte bezeichnet die Größe der einzelnen Lernobjekte. Sie beschreibt, in welche und in wie viele Einheiten Lehrmaterialien aufgeteilt werden sollen und auf welche Weise die Lernobjekte am besten geschnitten werden, um syntaktisch korrekt und semantisch sinnvoll zu sein. So kann beispielsweise ein Kapitel, ein ganzer Kurs oder ein Textstück mit einem bestimmten Inhalt ein Lernobjekt sein, ein einzelnes Wort üblicherweise jedoch nicht. Je gröber der Granularität von Lernobjekten gewählt wird, desto einge-

schränkter und unflexibler ist man in der Nutzung der Lernobjekte. Auch die Möglichkeit der Wiederverwendung eines Lernobjekts sinkt, je gröber dessen Granularität ist. Dagegen führt eine zu fein gewählte Granularität der Lernobjekte aufgrund sehr vieler Querbeziehungen und Abhängigkeiten (inhaltlich, bezüglich der Struktur, etc.) zwischen ihnen zu komplexen Geflechten von Lernobjekten, die nur sehr schwer zu pflegen sind.

Die geeignete Auswahl der Granularität von Lernobjekten beeinflusst ferner direkt die Nutzung der Lernobjekte. In Anlehnung an [Gri98] soll deshalb ein Lernobjekt klein genug sein, dass es eigenständig zu erstellen, zu warten und zu pflegen ist, und es soll groß genug sein, um eine anwendbare Funktionalität zu bieten und eine individuelle Pflege zu rechtfertigen. Des weiteren soll es mit standardisierten Schnittstellen ausgestattet sein, um einfach mit anderen Lernobjekten ausgetauscht oder kombiniert werden zu können.

In folgenden werden verschiedene Arten von Lernobjekten, die in der Literatur genannt werden (vgl. [LTSC], [Hod00], [Hof02]), vorgestellt und gegeneinander abgegrenzt. Abbildung 2-2 verdeutlicht die Zusammenhänge zwischen den Begriffen.

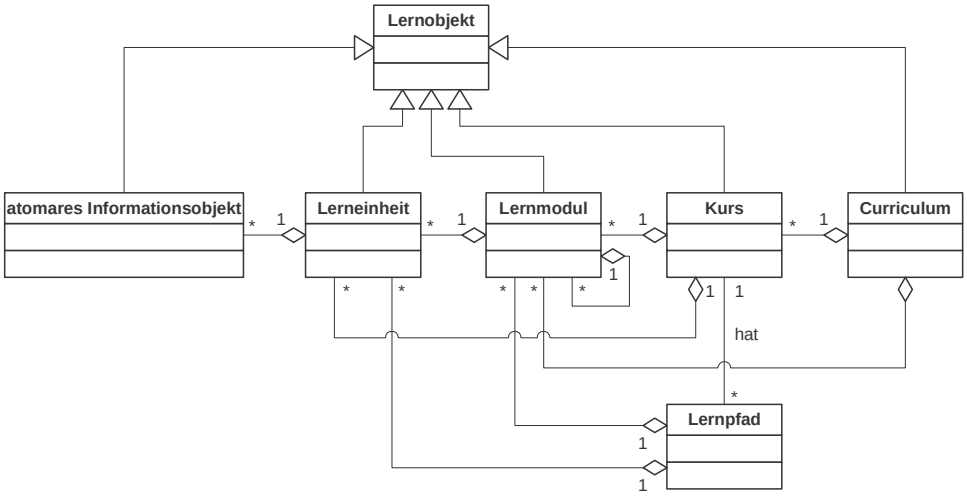


Abbildung 2-2: Begriffe zum Thema „Lernobjekt“ in der Literatur und ihr Zusammenhang

Atomare Informationsobjekte

Atomare Informationsobjekte sind rein syntaktische Einheiten, das heißt ein sehr kleines, eventuell multimediales Datenelement ohne eigenständige Nutzung in der Lehre. Sie sind nicht mehr sinnvoll unterteilbar. Beispiele für atomare Informationsobjekte sind ein Text, ein Bild oder ein beliebiges anderes Medienobjekt. Verwandte Begriffe im Lehr- und Lernumfeld sind *Lernfragment*, *Datenelement* und *Atom*. Wesentlicher Verwendungszweck atomarer Informationsobjekte ist es, zu weiteren Lernobjekten kombiniert zu werden [Hof02]; eine eigenständige Nutzung atomarer Informationsobjekte, beispielsweise als Ziel eines Verweises, ist im allgemeinen nicht vorgesehen.

Lerneinheiten

Eine Lerneinheit ist ein kleines Lernobjekt, das eine kleine Anzahl von atomaren Informationsobjekten wie Texte, Bilder, etc. kombiniert. Eine Lerneinheit strukturiert die in ihm enthaltenen Informationsobjekte auf elementarer Ebene; sie kann keine anderen Lerneinheiten beinhalten. Ein verwandter Begriff im Lehr-, Lernumfeld ist (*einfaches*) *Lernmodul*. Lerneinheiten werden analog zu den atomaren Informationsobjekten zu größeren Lernobjekten kombiniert. Im Gegensatz zu atomaren Informationsobjekten können Lerneinheiten aufgrund

ihrer konkreten Ausprägung (inhaltliche Abgeschlossenheit, etc.) auch als Ziel eines Verweises dienen, beispielsweise als Fußnote.

Lernmodule

Lernmodule können aus Lerneinheiten, atomaren Informationsobjekten und wiederum aus Lernmodulen kombiniert werden (strukturelle Rekursion), um einen konkreten Inhalt zu vermitteln. Ein Lernmodul hat eine logische Struktur. Ein Beispiel für ein Lernmodul wäre eine Beschreibung des Grove-Algorithmus, bestehend aus einer Einleitung, einer detaillierten Beschreibung und einem Beispiel. Ein verwandter Begriff im Lehr- und Lernumfeld ist *Reusable Learning Object* (*RLO*).

Kurse

Ein Kurs ist eine abgeschlossene Lehrveranstaltung zur Vermittlung von komplexen Inhalten mit einer starken logischen Strukturierung und einem didaktischen Ziel. Kurse sind nicht nur einem bestimmten Themenbereich, sondern auch einer bestimmten Nutzung, beispielsweise vorlesungsbegleitend, als Skript für das Selbststudium, für eine Tutorübung, etc., zugeordnet.

Curriculum

Ein Curriculum stellt einen Rahmen, meist eine Zusammenstellung von Kursen, bereit, der zur Erreichung eines wohldefinierten akademischen Zieles dient. Ein Beispiel hierfür ist „Curriculum des MBA in International Management“.

Lernpfad

Ein Lernpfad ist eine im allgemeinen (nutzungs-)spezifische Struktur, bestehend aus Lernmodulen und Lerneinheiten eines Kurses. Ein Lernpfad ermöglicht eine individuelle Nutzung der Lehrmaterialien bezüglich der Reihenfolge der einzelnen Themen. Lernpfade können vorgegeben oder adaptiert (personalisiert) für (einen einzelnen) Lerner sein.

2.3 Allgemeine Konzepte zur Unterstützung von Wiederverwendbarkeit

In diesem Abschnitt werden verschiedene Konzepte vorgestellt, um Wiederverwendbarkeit beliebiger Objekte[2] zu erreichen, wobei hier nicht nur Lernobjekte im Blickfeld stehen, sondern auch Konzepte der Wiederverwendung aus anderen Disziplinen, beispielsweise dem Software-Engineering, betrachtet werden.

Einige Konzepte bauen darauf auf, komplexe und stark spezialisierte Objekte in kleinere Objekte aufzuteilen, die für sich genommen nicht so speziell und somit leichter wiederzuverwenden sind. Die Konzepte der Objektorientierung und der Entwurfsmuster sowie Framework-basierte Ansätze sind Beispiele hierfür. Andere, eher technisch orientierte Konzepte sehen vor, Objekte sowie die Zusammenhänge zwischen Objekten möglichst formal und möglichst präzise zu beschreiben, um sie auf diese Weise für verschiedene Arten der Nutzung zugänglich zu machen. Beispiele hierfür sind der Metadatenansatz und die Anreicherung von Objekten mit einer Beschreibung ihrer Semantik, wie dies beispielsweise mit dem Resource Description Framework (RDF) [RDF] realisiert werden kann. Im folgenden werden diese Konzepte beschrieben.

[2] In diesem Abschnitt wird die Wiederverwendung beliebiger Entitäten – nicht nur im Zusammenhang mit Lehrmaterialien – betrachtet. Die wiederzuverwendenden Entitäten werden hier als Objekte bezeichnet.

2.3.1 Strukturelle Ansätze zur Erreichung von Wiederverwendbarkeit

2.3.1.1 Objektorientierung

Objektorientierung bietet eine Vielzahl konzeptueller und struktureller Möglichkeiten, Wiederverwendung im Kleinen (überschaubare, wiederverwendbare Elemente wie Klassen, etc.) zu erreichen [Rie96].

Grundlagen: Klassen, Objekte und Datenkapselung

Klassen dienen als Datenabstraktion zur Instanziierung von konkreten Objekten. Die Klasse ist quasi als Bauplan der konkreten Objekte zu verstehen. Eine Klasse als Einheit enthält identifizierte gemeinsame Eigenschaften (Struktur: Attribute, Verhalten: Methoden) aller Objekte dieser Klasse. Der Umfang dieser gemeinsamen Eigenschaften bestimmt den Wiederverwendungsgrad der Klassen (also was von ihnen und wie weit sie wiederverwendet werden). In diesem Fall wird Wiederverwendung dadurch unterstützt, dass die Eigenschaften einer Klasse (und damit die Eigenschaften aller Objekte der Klasse) gekapselt sind und lediglich über eine Schnittstelle verfügbar gemacht werden (Datenkapselung). Die Klassen bzw. die aus ihr instanziierten Objekte sind so in hohem Maße über klar definierte Schnittstellen wiederverwendbar.

Im Kontext „Wiederverwendung von Lehrmaterialien“ kann das Konzept der Objektorientierung folgendermaßen genutzt werden: Lernobjekte (Klassen) definieren allgemeine Eigenschaften des Anwendungsbereichs, beispielsweise das Themengebiet, nicht jedoch konkrete Inhalte. Objekte (Instanzen) dieser Klassen werden erzeugt, indem die Lernobjekte um bestimmte, speziellere Eigenschaften (beispielsweise den spezifizierten Schwierigkeitsgrad, das benötigte Vorwissen, etc.) und Verhaltensweisen des Anwendungsumfelds erweitert und mit konkreten Inhalten versehen werden. Wiederverwendung und Wartung werden nun dadurch erleichtert, dass konkrete Objekte mit ähnlichen Eigenschaften in Klassen, die jeweils nur einmal existieren, zusammengefasst sind.

Vererbung und Polymorphie

Eine Klasse ist Ausgangsbasis für die Vererbung durch weitere Klassen, sogenannte Unterklassen. Die Vererbung von einer Oberklasse zu einer Unterklasse wird durch eine Spezialisierung der Oberklasse realisiert: Struktur (Attribute) und Verhalten (Methoden) der Oberklasse werden in den Unterklassen wiederverwendet und – falls es der Anwendungsbereich erfordert - modifiziert. Da Klassen die Konzepte eines Anwendungsbereichs abbilden, kann durch Vererbung eine speziellere Nutzung innerhalb des Anwendungsbereichs realisiert werden.

Polymorphie ist dadurch gekennzeichnet, dass einzelne Operationen einer Objektschnittstelle durch mehrere Methoden innerhalb einer Klassenhierarchie implementiert werden können und je nach konkretem Objekt zur Laufzeit ermittelt wird, welche der Methoden ausgeführt werden soll. So können Operationen unter Beibehaltung der originalen Bezeichnung abgeändert werden.

Die Konzepte Vererbung und Polymorphie beschreiben im Kontext „Wiederverwendung von Lernobjekten“ die Idee, dass bereits konkretisierte Lernobjekte sowohl statisch (anhand der vorgesehenen Nutzungen), als auch dynamisch (beispielsweise aufgrund schwer absehbarer Nutzerinteraktionen) zur Laufzeit spezialisiert werden. Durch das Konzept der Polymorphie liefert die Objektorientierung die Voraussetzungen, eine verbesserte Erweiterbarkeit von Lernobjekten zu gewährleisten.

Klassenbibliotheken

Klassenbibliotheken sind anwendungsübergreifende Baukästen, die beispielsweise ein Unterprogramm, eine fundamentale Basisklasse, etc., bereitstellen [Bal98], welche in verschiedenen Anwendungen als Ganzes genutzt werden können. Eine typische Vorgehensweise ist die Verwendung vorgefertigter Klassenbibliotheken, die dann anstatt oder ergänzend zu den vorhandenen Klassen in der Objektdesign- und in der Implementierungsphase eingebaut werden.

Das Konzept der Klassenbibliotheken beschreibt im Kontext „Wiederverwendung von Lernobjekten“ die Idee, dass bereits vorgefertigte Lernobjekte, die nur rudimentäre Eigenschaften besitzen, ohne Änderung bzw. lediglich durch Konfiguration angepasst in den anderen Lernobjekten wieder verwendet werden können.

2.3.1.2 Entwurfsmuster

In der objektorientierten Entwicklung sind Entwurfsmuster Lösungsvorschläge, die Entwickler mit dem Ziel entworfen haben, eine Reihe wiederkehrender Probleme bereits beim Design zu lösen [GHJV96]. Entwurfsmuster sind eine Möglichkeit, Entwurfserfahrung so festzuhalten, dass sie von Entwicklern wiederverwendet werden kann.

Ein Entwurfsmuster besteht aus wenigen Klassen, die durch den Einsatz von Delegation und Vererbung eine robuste und modifizierbare Lösung ermöglichen. Diese Klassen können für das zu erstellende System angepasst und verfeinert werden [BrDu04]. Ein schöner Überblick über viele Entwurfsmuster mit festgelegtem Beschreibungsschema (Patternname, Problembeschreibung, Problemlösung) ist in [GHJV96] zu finden.

So wie Entwurfsmuster Entwurfserfahrung für häufig vorkommende Probleme beim Software-Entwurf beschreiben, können Entwurfsmuster auch als Best Practices für die Gestaltung von Lehrmaterialien im Hinblick auf ein didaktisches Ziel herangezogen werden. So kann beispielsweise eine besonders passende Struktur von komplizierter aufgebauten Lernobjekten, die alle ein ähnliches (didaktisches) Ziel haben, in einem „Lehrmaterial-Entwurfsmuster“ beschrieben sein.

2.3.1.3 Komponenten und Frameworks

Ein Framework ist eine abstrakte, mehr oder weniger vollständige Architektur eines Problembereichs, welches als konzeptionelles Mittel für Analyse, Design und Implementierung von Software dient. Framework-basierte Wiederverwendung eignet sich für die Wiederverwendung im Großen (komplexe Funktionen, Use Cases, Teilsysteme), welche auf der Bildung von Anwendungsbaukästen basiert. Dies unterscheidet Frameworks von Klassen und Klassenbibliotheken, die meist eine Wiederverwendung im Kleinen (Methoden, einzelne Klassen) unterstützen [Gri98] und wenig bereichsspezifisch sind [BrDu04].

Frameworks fassen das Wissen über Funktionsweisen und Strukturen einer Anwendungsdomäne in Form von abstrakten Klassen zusammen und kondensieren dieses in einer weiter- und wiederverwendbaren Form [Gri98]. Komponenten[3], die auf einem Framework basieren, sind dann eine Ausprägung des Frameworks. Die Verfeinerung von Framework-basierten Komponenten kann sowohl in White Box-, als auch in Black Box-Entwicklung erreicht werden. Bei der White Box-Entwicklung wird die Grundstruktur, die durch das jeweilige Framework spezifiziert wurde, konkretisiert, während bei der Black Box-Entwicklung eine Rekonfiguration des Frameworks über definierte Schnittstellen durchgeführt wird. In beiden Fällen

[3] Frameworks können für verschiedenste Domänen entwickelt werden. In dieser Arbeit sind insbesondere Frameworks von Interesse, die mit Lernobjekten als Komponenten genutzt werden. Da in diesem Abschnitt jedoch allgemein skizziert wird, wie Wiederverwendung mittels Frameworks funktioniert, soll hier statt von Lernobjekten weiterhin der allgemeinere Begriff der Komponente verwendet werden.

kann die entstandene Komponente aufgrund ihres abgeschlossenen Charakters selbst wieder in kompletten Anwendungen als Black Box wiederverwendet werden.

Beim Framework-basierten Ansatz muss für eine Komponente zuerst eine Grundstruktur als „Rahmen“ (Container) definiert werden. Dies geschieht mit der Auswahl des passenden Frameworks. Das Framework liefert die Grobstruktur der Komponenten und bietet klare Schnittstellen zum Anfügen weiterer Komponenten und für die Anpassung des Frameworks an konkrete Anforderungen. Im Kontext „Wiederverwendung von Lernobjekten“ beschreibt dieser Ansatz die Idee, für große Lernobjekte, wie zum Beispiel einen Kurs, zuerst den Rahmen zu definieren, indem die Struktur des Kurses festgelegt wird, und dann erst konkrete Inhalte für unterschiedliche Nutzungen einzufügen.

2.3.2 Technische Ansätze zur Erreichung von Wiederverwendbarkeit

2.3.2.1 Metadaten

Metadaten sind eine häufig eingesetzte Methode, beliebige Objekte zu beschreiben. Mit Hilfe dieser Beschreibungen wird die Anwendung und auch eine Wiederverwendung dieser Objekte deutlich erleichtert. In dieser Arbeit sollen insbesondere Metadaten zu Lernobjekten betrachtet werden.

Metadaten werden verwendet, um Eigenschaften von Lernobjekten oder Beziehungen zwischen Lernobjekten zu beschreiben. Metadaten werden bei der Erstellung der Lernobjekte definiert und bei der Nutzung der Lernobjekte ausgewertet. Im folgenden werden Metadaten – und insbesondere Metadaten zu Lernobjekten – näher betrachtet.

Daten und Metadaten

Der Begriff *Metadaten* bezeichnet (strukturierte) Informationen über Informationen. Metadaten sind ein formaler Mechanismus zur Beschreibung und zur Strukturierung von umfangreichen Daten und Beziehungen zwischen diesen Daten. Es gibt keine exakte Festlegung von Metadaten; es hängt immer von den jeweils zu beschreibenden Objekten und ihrer Anwendung ab [ToD].

Der Vorteil von Metadaten besteht zum einem darin, dass zusätzliche Informationen, die nicht in den Daten selbst enthalten sind, bearbeitet werden können, und zum anderen, dass die Relevanz der beschriebenen Daten direkt ermittelt werden kann, ohne auf die Daten selbst zugreifen zu müssen. Der Overhead der beschreibenden Daten (der Metadaten) im Vergleich zu den zu beschreibenden Daten bleibt meist gering.

In Tabelle 2-1 sind Metadaten zu einem Lernobjekt auszugsweise beschrieben, mit denen dieses Objekt und die damit gebundenen Eigenschaften deklarativ spezifiziert werden können:

Name	Angabe nötig	Kommentar	Beispiel
Identifier	Reserviert	Datum und Eingangsnummer	„29_04_01_001“
Title	Verpflichtend	Name des Objekts	„Korrelation“
Authors	Verpflichtend	Autoren	N.N
Date	Verpflichtend	Datum des Objekts	29.04.01
Language	Verpflichtend	Sprache	„deutsch“
Publisher	Verpflichtend	Partner, Universität	„N.N., Rostock“

Tabelle 2-1: Allgemeine Metadaten für ein Lernobjekt (Ausschnitt) [Schu03]

In verschiedenen Standards, beispielsweise LOM [LOM], IMS [IMS a] oder Dublin Core [Dub b] sind bereits umfangreiche Metadaten für Lernobjekte festgelegt (vgl. 2.5.2.).

Klassifizierung von Metadaten

Da es für Metadaten keine eindeutige Begriffsbildung gibt, können sie hinsichtlich ihrer Anwendung nicht exakt klassifiziert werden. Im folgenden wird in Anlehnung an [IM01] ein Beispiel für eine Klassifizierung vorgestellt:

Es gibt *allgemeine Metadaten*. Typischerweise handelt es sich dabei um identifizierende Merkmale und Verwaltungsinformation der Daten, wie etwa der Name des Autors, ein generierter Identifikator (ID), Autoren, Copyright, etc.

Eine andere Sorte von Metadaten sind die *inhaltlichen Metadaten*. Als inhaltliche Metadaten werden Daten bezeichnet, die den Inhalt der Daten beschreiben und zugleich weiterführende Informationen bieten, beispielsweise eine Referenz auf eine Wissensbasis zu einem bestimmten, benachbarten Themengebiet. So können beispielsweise Verweise auf andere Lernobjekte mittels inhaltlicher Metadaten realisiert werden.

Ferner gibt es *analytische Metadaten*, die es ermöglichen, die Nutzung von Daten gezielt zu ermitteln. Die analytischen Metadaten sind Informationen, die den Zugang zu den Inhalten der Ressourcen ermöglichen und verbessern, beispielsweise Schlagwörter oder Stichworte[4].

Daneben gibt es noch *strukturelle Metadaten*, die sich auf die Struktur der Daten beziehen, wie beispielsweise Document Type Definition (DTD)s [DTD], Extensible Markup Language (XML)-Schema [XML Schema], etc.

Neben den bisher genannten Metadaten gibt es noch technische Metadaten, rechtliche Metadaten. etc., welche in dieser Arbeit jedoch nicht weiter betrachtet werden sollen.

Darstellung und Speicherung von Metadaten

Metadaten können unabhängig von den Daten, die sie beschreiben, gespeichert werden. Sie müssen nicht immer manuell erfasst werden; unter bestimmten Randbedingungen (bekannte Semantik der Daten, bestimmte, für alle Daten verbindliche Vorgaben hinsichtlich ihrer Struktur, etc.) ist es auch möglich, sie (teilweise) automatisch zu erfassen.

Semantik von Metadaten: Das Resource Description Framework (RDF)

Das Resource Description Framework (RDF) ist eine Sprache, mit der die Semantik von beliebigen Ressourcen formal beschrieben werden kann. RDF kann so auch für die Beschreibung von Lernobjekten verwendet werden: Eine semantische Interoperabilität von Lernobjekten wird möglich.

RDF stellt einen allgemeinen Rahmen dar, so dass RDF-Informationen und damit die Semantik von Ressourcen – hier insbesondere von Lernobjekten – zwischen Programmen und Operationen ausgetauscht werden können, ohne dass die Bedeutung der Ressourcen verloren geht. Grundsätzlich besteht die Beschreibung einer Ressource in RDF aus einem Tripel „Objekt-Attribut-Wert“: Ein Objekt *O* hat ein Attribut *A* mit dem Wert *V*. Man kann dieses Tripel (*O, A, V*) auch als eine mit *V* gewichtete Kante zwischen zwei Knoten *O* und *A* auffassen. Auf diese Weise wird durch die über RDF-Spezifikationen intendierte Semantik der erste Schritt zur technologischen Grundlage des Semantic Web gemacht [Schi04].

[4] In dieser Arbeit fallen inhaltliche und analytische Metadaten aus Gründen der Einfachheit zusammen.

Im Zusammenhang mit RDF wurden bereits viele Arbeiten veröffentlicht. Der interessierte Leser sei beispielsweise auf [RDF] oder [SW a] verwiesen.

2.3.2.2 Einschränkungen (Constraints) und Transformationsregeln

Neben Metadaten sind Einschränkungen (Constraints) und Transformationsregeln (beispielsweise Style Sheets, mehr hierzu in [Wiki] und 3.4.2.3) eine weitere Möglichkeit zur Unterstützung der Wiederverwendung. Einschränkungen und Regeln definieren auf unterschiedlichem Abstraktionsniveau (Ebene der konkreten Objekte oder Ebene der Strukturen zwischen Objekten) den Austausch und die Integration von Lehrmaterialien, indem sie Konventionen eines Anwendungsbereichs festlegen oder offenlegen: So muss zur Nutzung eines bestimmten Objekts sichergestellt sein, dass das Objekt die durch die Regeln bzw. Einschränkungen bestimmten Vor- und Nachbedingungen erfüllt und so die oben genannten Konventionen des Anwendungsbereichs einhält.

Ein Beispiel hierfür ist die Objekt Constraint Language (OCL), mit der Einschränkungen von Lernobjekten und ihren Eigenschaften in formaler Weise ausgedrückt werden können, vgl. [OMG]. Diese Einschränkungen können entweder als wahr oder falsch ausgewertet werden. So kann die Wiederverwendung durch Invarianten, Vorbedingungen und Nachbedingungen sichergestellt werden, indem nicht nur Eigenschaften der Lernobjekte, sondern auch die von ihnen erwarteten Eigenschaften und Funktionalitäten anderer Lernobjekte modelliert werden, vgl. [BrDu04].

Style Sheets sind eine Anwendung von Transformationsregeln. Ein bekanntes Beispiel hierfür wäre die Extensible Stylesheet Language (XSL) [XSLT], mit der es möglich wird, hierarchisch angeordnete Dokumente in Dokumente mit einer anderen Struktur zu überführen. Eine andere Anwendung von Transformationsregeln sind Cascading Style Sheets (CSS) [CSS], mit denen es möglich wird, einem Dokument mehrere, unterschiedliche Layouts zuzuweisen und dieses eine Dokument für verschiedene Anwendungen zu nutzen.

2.4 Wiederverwendbarkeit von Lernobjekten

Aus Zeit- und Kostengründen ist es nicht erstrebenswert und oft auch nicht möglich, Lehrmaterialien für jede Lehrveranstaltung und für jeden Kurs bzw. jede Nutzung im Kurs neu zu entwickeln. Man wird statt dessen versuchen, bereits bestehende Lehrmaterialien – und hier insbesondere die Lernobjekte – so weit wie möglich wiederzuverwenden.

Den Einsatz wiederverwendbarer Lehrmaterialien in verschiedenen Kursen oder Nutzungen nennt man *Wiederverwendung*. Wiederverwendung ist also das erneute Anwenden von bei der Erstellung von Lehrmaterialien entstandenen Artefakten (den Lernobjekten) und des angesammelten Wissens bei der Entwicklung neuer Lehrmaterialien, um den Aufwand zur Erstellung und Pflege dieser neuen Lehrmaterialien zu reduzieren (in Anlehnung an [Gri98] und [BiPe89]). Schwerpunkt dieser Arbeit ist die Wiederverwendung von Lehrmaterialien zwischen verschiedenen Personen und über Organisations- und Unternehmensgrenzen hinweg.

Bevor genauer betrachtet wird, welche Schritte nötig sind, um möglichst gut wiederverwendbare Lernobjekte zu gestalten, sollen allgemeine Überlegungen angestellt werden um zu klären, welche Eigenschaften wiederverwendbare Lernobjekte generell haben sollten.

2.4.1 Allgemeine Anforderungen an wiederverwendbare Lernobjekte

In der Literatur hat man bereits eine recht präzise Vorstellung davon, wie möglichst gut wiederverwendbare Lernobjekte gestaltet sein müssen (vgl. [TeBr02], [SüFr01], [ToD]). Im folgenden sind die zentralen Anforderungen zusammengefasst:

Flexibilität

Lernobjekte müssen innerhalb eines Themen- oder Anwendungsbereichs möglichst flexibel einsetzbar sein; im Idealfall sind sie sogar themen- oder anwendungsübergreifend nutzbar. Sie müssen daher „ausreichend allgemein“ gestaltet sein und dürfen nicht ausschließlich auf eine bestimmte Anwendung (beispielsweise auf eine einzige Lehrveranstaltung, eine bestimmte Art der Darstellung, etc.) zugeschnitten sein. Hier muss versucht werden, einen Ausgleich zu schaffen: Je allgemeiner ein Lernobjekt ist, desto weniger genügt es möglicherweise den Anforderungen für bestimmte Aufgaben. Aber je spezieller ein Lernobjekt auf eine bestimmte Anwendung ausgerichtet ist, desto schwieriger ist es, das Lernobjekt für andere Anwendungen (beispielsweise in anderen Lehrveranstaltungen mit anderen Lehrformen) wiederzuverwenden. Dieser Ausgleich kann erreicht werden, indem die Lernobjekte möglichst lose mit ihren Anwendungen gekoppelt und möglichst schwach mit anderen Lernobjekten verbunden werden.

Modularisierung

Die Idee zur Modularisierung ist daraus motiviert, dass Lehrmaterialien einer Lehrveranstaltung als Ganzes sehr komplex und so nur schwer zu beherrschen sind. Um Lehrmaterialien leichter pflegen zu können, werden sie in kleinere, einfachere Lernobjekte zerlegt, die als Module für neue Lehrmaterialien dienen können. Voraussetzung ist hierbei, dass die Lernobjekte inhaltlich in sich abgeschlossen sind – enge Beziehungen zwischen verschiedenen Lernobjekten müssen so weit wie möglich vermieden werden.

Ein weiterer Aspekt betrifft die Größe (Granularität) der Lernobjekte. Je größer ein Lernobjekt ist, desto umfangreicher ist auch das Thema – bezüglich des Detaillierungsgrades oder der Vielfalt der angrenzenden Themengebiete. Große, umfangreiche Lernobjekte sind im allgemeinen deutlich schwieriger wiederzuverwenden als kleine, auf ein überschaubares Thema fixierte Lernobjekte. So hat die Größe und der Umfang eines Lernobjekts unmittelbar Einfluss auf dessen Wiederverwendbarkeit. Dieser Zusammenhang muss auch berücksichtigt werden, wenn aus kleinen Lernobjekten größere, komplexere Lernobjekte aufgebaut werden. Zur weiterführenden Erklärung der Modularisierung wird an dieser Stelle auf [BMRS98] verwiesen.

Trennung von Struktur und Navigation der Lernobjekte

Werden Lernobjekte iterativ zu neuen, komplexeren Lernobjekten kombiniert, entsteht eine hierarchische Struktur. Diese Struktur sollte nicht einzige Grundlage der Navigation zwischen den Lernobjekten werden, da man sonst unnötig auf eine Navigation in einem Baum eingeschränkt wird. Statt dessen sollte die Navigation unabhängig von der Strukturierung der Lernobjekte entworfen werden. So ist es dann beispielsweise möglich, eine Navigationsstruktur zu erstellen, die den Studenten anzeigt, welche Lernobjekte des Kurses vom aktuell angezeigten Lernobjekt unter welchen Bedingungen (Vorwissen, bereits absolvierte Lernobjekte) erreicht werden können. Die Navigation kann eine einfache lineare Struktur sein wie bei einem Vortrag, aber auch eine wesentlich komplexere, verlinkte Struktur.

Schaffung von Möglichkeiten zur Adaptierung

Lernobjekte sollen nicht starr, sondern flexibel an die beabsichtigte Nutzung anpassbar sein. Das heißt, sie müssen eine Möglichkeit der Anpassung an unterschiedliche Nutzungen bieten. Dies kann mittels Eigenschaften, die von den Autoren oder den Nutzern geeignet gesetzt werden, erreicht werden. Des weiteren ist es sinnvoll, Lernobjekte nicht fest miteinander zu verdrahten, sondern die Beziehungen zwischen Lernobjekten flexibel zu gestalten. So könnten beispielsweise Beziehungen zwischen Lernobjekten auf Eigenschaften der Lernobjekte

oder auf bestimmten, von den Autoren vorgegebenen Schablonen basieren. Mehr dazu siehe [Cok] oder [Har00].

2.4.2 Planung der Wiederverwendung von Lernobjekten

Bei der Wiederverwendung müssen zwei Aspekte betrachtet werden: Wiederverwendung von bereits bestehenden Lernobjekten und Berücksichtigung der Wiederverwendung bei der Erstellung neuer Lernobjekte.

Bei der Wiederverwendung bereits bestehender Lernobjekte werden aus vorliegenden Lehrmaterialien einzelne Teile (die Lernobjekte) extrahiert. Anschließend werden diese – falls möglich und nötig – geeignet überarbeitet und dann in die neuen Lehrmaterialien integriert [CaHo01].

Soll die Wiederverwendung von Lernobjekten bereits bei deren Erstellung berücksichtigt werden[5], so müssen die Lernobjekte so konzipiert werden, dass sie in Kursen mit unterschiedlichen Themengebieten oder für den gleichen Kurs, aber bei unterschiedlichen Arten der Nutzung (Vorlesung, Tutorübung, Skript, etc.), verwendet werden können. Hierzu müssen sowohl die primären Eigenschaften der Lernobjekte, wie beispielsweise ihr Inhalt, als auch die verschiedenen beabsichtigten Nutzungen berücksichtigt werden.

Wiederverwendung wird oft sowohl innerhalb der gleichen Domäne (domänenspezifisch) als auch zwischen unterschiedlichen Domänen (domänenübergreifend) betrachtet. Dabei gliedert sich der Begriff Domäne im Kontext dieser Arbeit in *Inhaltsdomäne* (die Themengebiete) und in *Anwendungsdomäne* (die Nutzungsbereiche). Im folgenden wird kurz auf domänenübergreifende und domänenspezifische Wiederverwendung eingegangen.

2.4.2.1 Domänenübergreifende Wiederverwendung

Domänenübergreifende Wiederverwendung von Lehrmaterialien bedeutet hier die Nutzung der Lehrmaterialien in unterschiedlichen Inhalts- oder Anwendungsdomänen, also beispielsweise in Lehrveranstaltungen mit unterschiedlichen Themengebieten. Dies bedeutet, dass ein Lernobjekt *A* sowohl in einer Lehrveranstaltung der Domäne 1 als auch einer Lehrveranstaltung der Domäne 2 genutzt werden kann (vgl. Abbildung 2-3).

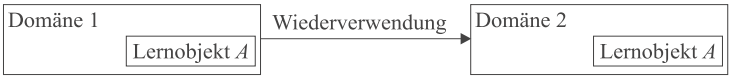


Abbildung 2-3: Domänenübergreifende Wiederverwendung

Damit dies erreicht werden kann, müssen Lernobjekte, die domänenübergreifend eingesetzt werden sollen, in ihren Eigenschaften möglichst allgemein gehalten werden. Ebenso dürfen nur möglichst wenige Annahmen über die zukünftige Nutzung der Lernobjekte gemacht werden. Die Autoren der Lernobjekte müssen darauf achten, möglichst viele Freiheitsgrade (also noch nicht fix belegte und somit noch frei setzbare Eigenschaften) zu spezifizieren, um so einen flexiblen Einsatz der Lernobjekte und damit deren Wiederverwendung überhaupt erst möglich zu machen.

2.4.2.2 Domänenspezifische Wiederverwendung

Domänenspezifische Wiederverwendung von Lehrmaterialien bedeutet in dieser Arbeit die Erstellung und Nutzung von Lehrmaterialien oder Kurse gleicher Inhalte in unterschiedlich spezifizierten didaktischen Funktionen und / oder unterschiedlichen Präsentationsformaten. So würde man von domänenspezifischer Wiederverwendung sprechen, wenn ein und derselbe Kurs einmal als Microsoft Powerpoint-Präsentation zur Unterstützung des Dozenten bei

[5] Auf diesen Aspekt wird in 2.6.1 genauer eingegangen.

der Vorlesung und einmal als vorlesungsbegleitendes Volltext-Skript vorliegt. Die beiden im Beispiel genannten Nutzungen des Kurses wären dann „Präsenzveranstaltung“ und „Selbststudium“. Abbildung 2-4 illustriert diesen Gedanken: Kurs 1 und mit ihm Lernobjekt *A* wird einerseits in einer Präsenzveranstaltung („Nutzung 1“) eingesetzt, andererseits beim Selbststudium („Nutzung 2“).

Abbildung 2-4: Domänenspezifische Wiederverwendung

Einige Eigenschaften von Lernobjekten sind bei domänenspezifischer Wiederverwendung planbar und können damit bereits bei der Konzeption der Lernobjekte berücksichtigt werden, beispielsweise die Kursform oder die Zielgruppe der Lehrveranstaltung. Andere Eigenschaften wiederum hängen von schwer einschätzbaren Ereignissen ab, beispielsweise dem Interaktionsverhalten der Studenten während einer Vorlesung. So kann es während einer Vorlesung nötig sein, mehrmals zwischen verschiedenen Kapiteln hin- und herzuspringen, um bestimmte Fragen zu beantworten, so dass ein streng sequenzielles Layout des Kurses denkbar ungünstig wäre. Für Eigenschaften dieser Art muss der Autor der Lernobjekte Freiheitsgrade ermöglichen, so dass eine jeweils der aktuellen Situation angepasste Nutzung der Lernobjekte möglich wird.

In [JGJ97] wird bemerkt, dass eine domänenübergreifende Wiederverwendung (vgl. 2.4.2.1) kaum erreicht werden kann und auch nicht immer sinnvoll ist. Es ist oft lohnenswerter, sich auf ein paar Domänen zu beschränken und dann für diese Domänen das Domänendesign durchzuführen.

Domänendesign umfasst das Erfassen möglichst aller Umstände und Anforderungen, unter denen Lehrmaterialien später eingesetzt werden sollen. Dies umfasst die Zielgruppe(n), die Art der Lehrveranstaltung, etc. Aus diesen Umständen und Anforderungen lassen sich Eigenschaften ableiten, die Lehrmaterialien für den geplanten Einsatz haben müssen. Folgende Fragen (in Anlehnung an [Gri98]) können helfen, die Eigenschaften der Domäne und darauf aufbauend die geforderten Eigenschaften der Lernobjekte zu identifizieren und genauer zu verstehen[6]:

- Sind die Eigenschaften einer Domäne eine Verallgemeinerung oder Spezialisierung bereits vordefinierter Eigenschaften?
- Wenn die Eigenschaften eine Verallgemeinerung sind, sind sie allgemein genug, um potentiell in anderen Anwendungen der gleichen Anwendungsdomäne verwendet zu werden?
- Kann eine Eigenschaft, für die dies nicht gilt, durch kleine Änderungen (beispielsweise Änderung des Attributwertes) für andere Anwendungen sinnvoll nutzbar gemacht werden?
- Ist eine in der untersuchten Domäne identifizierte Eigenschaft abhängig von anderen Domänen?
- Benötigen verschiedene Anwendungen dieselbe Eigenschaft?

[6] In diesen Fragen spiegelt sich die problematische Frage einer Metrik wider, der Suche und Anwendung von messbaren Größen und Messverfahren, um überhaupt sinnvolle Angaben über Faktoren wie den Grad der Wiederverwendung beziffern zu können [Gri98].

- Wie häufig lässt sich eine bestimmte benötigte Eigenschaft in der untersuchten Anwendungsdomäne finden?

Ergebnis des Domänendesigns ist ein Domänenmodell, in dem die Eigenschaften und Funktionen der Anwendungsdomäne spezifiziert sind. Minimale Anforderung an das Domänenmodell ist dabei die genaue Beschreibung der Anwendungsdomäne.

2.4.3 Möglichkeiten zur Umsetzung der Wiederverwendung

Die domänenspezifische Wiederverwendung wird realisiert, indem die generellen (für alle Domänen gültigen) Eigenschaften der Lernobjekte auf das Domänenmodell, das im gerade beschriebenen Domänendesign identifiziert wurde, übertragen werden. Ergebnis dieses Vorgangs ist eine Menge von Eigenschaften von Lernobjekten, die innerhalb der vorgegebenen Domänen variiert werden können:

- Inhalt

  Lernobjekte gleichen Inhalts können auf unterschiedliche Weise – beispielsweise in unterschiedlichen Detaillierungsgraden – genutzt werden.

- Präsentationsformat

  Lernobjekte können in unterschiedlichen Formaten vorliegen. Beispielsweise könnte ein Lernobjekt in HTML für die Nutzung in der Präsenzveranstaltung und in PDF für die Nutzung im Selbststudium formatiert sein.

- Medientyp

  Mehrere Lernobjekte können den gleichen Inhalt in verschiedenen Medientypen (Animation, Text, Sprache, etc.) wiedergeben.

- Darstellung und Präsentationslayout

  Je nach Nutzungsart können verschiedene Visualisierungen von Lernobjekten genutzt werden. Für die Nutzung während einer Vorlesung beispielsweise sind andere Schriftarten und Farben angebracht als für ein Skript für das Selbststudium. Analog kann auch der Navigationsstil definiert werden.

- Struktur

  Für unterschiedliche Nutzungen können Lernobjekte unterschiedlich strukturiert werden. Beispielsweise lineare, nach Inhalt strukturierte Lernobjekte für das Selbststudium und nach Übungseinheiten strukturierte Lernobjekte für die Tutorübung. Im ersten Fall entspricht die Struktur den Kapiteln, im zweiten Fall den Übungseinheiten.

Um die Wiederverwendung von Lernobjekten hinsichtlich einer oder mehrerer der oben genannten Variationen umzusetzen, wird die beabsichtigte Nutzung mit dem Ergebnis des Domänendesigns abgeglichen. Üblicherweise werden folgende Schritte durchgeführt: Auswahl von Lernobjekten entsprechend bestimmter, vorgegebener Eigenschaften und Anpassung und / oder Evaluation der ausgewählten Lernobjekte im Hinblick auf die beabsichtigte Nutzung. Als Ergebnis erhält man domänenspezifisch angepasste Lernobjekte. Die beabsichtigte Nutzung legt Anforderungen fest, über welche Suchkriterien definiert werden können. Mit-

tels dieser Suchkriterien können gezielt Lernobjekte ermittelt werden. Über die Wahl der Suchkriterien wird zugleich auch festgelegt, welche Art der Variation vorgesehen ist[7].

Eine Anpassung und / oder Evaluation kann auf mehrere Weisen erfolgen. Im folgenden werden einige generelle Möglichkeiten vorgestellt. Diese können auch verschränkt ausgeführt werden:

- Austausch

  Gegeben sei ein Lernobjekt *C*, das aus einem Lernobjekt *A* und einem Lernobjekt *D* besteht. Lernobjekt *A* wird gegen ein anderes Lernobjekt *B* austauscht, wobei sich *A* und *B* in Inhalt, Struktur oder Darstellung unterscheiden können. Auf diese Weise können neue Instanzen *C*' von Lernobjekt *C* erstellt werden, die sich in einem Lernobjekt unterscheiden. Grund für den Austausch könnte eine Verbesserung der Darstellung oder der didaktischen Präsentation von Lernobjekt *C* sein.

- Weiterentwicklung eines Lernobjekts

  Lernobjekte können auch weiterentwickelt werden, indem ihr Inhalt erweitert oder geändert wird. Dies kann mittels Black Box- oder als White Box-Wiederverwendung geschehen. Die Black Box-Wiederverwendung verändert die vollständig gekapselten Inhalte und Strukturen der Lernobjekte nicht. Sie wird lediglich mittels einer „nutzt"-Beziehung zwischen dem neuen Lernobjekt und dem wiederverwendeten Lernobjekt realisiert [Bal98]. Im Gegensatz dazu wird bei der White Box-Wiederverwendung eine echte Datenänderung durchgeführt. Die Weiterentwicklung kann ferner einer Versionsableitung bedeuten. Mehrere Versionen existieren und sind nachzuvollziehen, und je nach Anwendung wird eine passende Version ausgewählt. Für mehr Information zu Versionierung wird an dieser Stelle auf das Zweigmanagement in [BrDu04] verwiesen.

- Integration und Konfiguration

  Eine andere Art der Anpassung ist die Integration und Konfiguration von Lernobjekten, beispielsweise die Kombination von Lernobjekt *A* und Lernobjekt *B* zu einem neuen, komplexer aufgebauten Lernobjekt *C*.

  Die Konfiguration eines Lernobjekts *C* erfolgt, indem Eigenschaften von *C* entsprechend den Anforderungen der beabsichtigten Nutzung gesetzt bzw. geändert werden.

- Adaption

  Eine Adaption von Lernobjekten kann dadurch erreicht werden, dass noch nicht berücksichtigte Aspekte durch neu hinzuzunehmende, bislang nicht betrachtete Eigenschaften nun mit betrachtet werden. Der Unterschied zum letzten Punkt „Integration und Konfiguration" ist, dass nicht nur bereits bestehende Eigenschaften geeignet gesetzt werden, sondern neue Eigenschaften zu einem Lernobjekt hinzukommen.

  Die neu hinzukommenden Eigenschaften können allerdings in vielfacher Weise mit den bereits bestehenden Eigenschaften in Beziehung stehen. Dadurch wird eine Konsistenzprüfung der betroffenen Lernobjekte unumgänglich. Nur so kann vermieden werden, dass ein Lernobjekt durch die neu hinzukommenden Eigenschaften in einen inkonsistenten Zustand durch einander widersprechende Eigenschaften gerät.

[7] Plant man eine Wiederverwendung des Inhalts, so wird eine Suche nach entsprechenden Stichworten stattfinden. Plant man hingegen eine Wiederverwendung von Lernobjekten, die eine bestimmte Struktur aufweisen, so findet eine Suche über Metadaten, die Auskunft über die Struktur geben, statt.

2.5 Mechanismen zur Realisierung wiederverwendbarer Lernobjekte

Nachdem die Planung, auf welche Weise Lernobjekte wiederverwendet werden können und sollen, abgeschlossen ist, können verschiedene Mechanismen eingesetzt werden, um die Wiederverwendbarkeit der Lernobjekte zu unterstützen. In 2.3 wurden einige Ansätze bereits allgemein vorgestellt. Hier werden diese Ansätze nun auf Lernobjekte angewendet.

2.5.1 Wiederverwendung mittels Metadaten

Mit Metadaten können Lehrmaterialien beschrieben und später flexibel hinsichtlich verschiedenster Lehr- und Lernnutzungen eingesetzt und gestaltet werden. Setzen die Autoren konsequent Metadaten ein und werten die Nutzer konsequent Metadaten aus, ist es leichter möglich, Lernobjekte in verschiedenen Anwendungen zu nutzen sowie Lernobjekte anhand bestimmter Eigenschaften zu finden, zu konfigurieren, auszutauschen und zu adaptieren.

Beschreibung und Strukturierung von Lernobjekten

Metadaten werden für ein bestimmtes Objekt meist nur einmal angelegt [Schu03]. Bei der Erstellung eines Lernobjekts können Metadaten definiert werden, die für die Identifikation des Lernobjekts, für administrative Zwecke und für die Beschreibung des Inhalts, der Struktur und der Darstellung des Lernobjekts dienen. Diese Beschreibung kann mittels einer Vielzahl von Datenstrukturen realisiert werden: Von einfachen Attribut-Wert-Paaren bis hin zu komplexen Graphen.

Ferner können Lernobjekte mit Hilfe von Metadaten unterschiedlichen Anwendungsdomänen zugeordnet werden, indem die Metadaten und die eigentlichen Lernobjekte getrennt voneinander gespeichert werden. Dies ermöglicht es, einem einzigen Lernobjekt mehrere Sätze von Metadaten für jeweils verschiedene Anwendungsdomänen bzw. Nutzungen innerhalb einer Anwendungsdomäne zuzuweisen und die Werte entsprechend der Anwendungsdomänen zu spezifizieren.

Suchen und Auffinden von Lernobjekten

Bei großen Beständen von Lernobjekten ist es wünschenswert, die Lernobjekte strukturiert verwalten zu können. Es gibt eine Reihe unterschiedlicher Ansätze zur Unterstützung der Suche nach Lernobjekten anhand von Metadaten. Im Vergleich beispielsweise zur Volltextsuche, die eine syntaktische Suche nach einzelnen Zeichenketten unterstützt[8], kann eine metadatenbasierte Suche stärker auf die Semantik ausgerichtet sein (vgl. analytische Metadaten in 2.3.2.1), wie beispielsweise „Suche nach Schlagwort xyz“: Statt große Datenbestände mit aufwändiger Volltextsuche nach „xyz“ zu durchforsten, ist es effizienter und nachvollziehbarer, Metadaten geeignet anzulegen, die um ein Vielfaches kleiner sind und damit schneller durchsucht und aufgefunden werden können.

Austausch von Lernobjekten

Mit der Erschließung neuer Informations-, und Kommunikationstechnologien werden Techniken zum gezielten system- und institutionsübergreifenden Austausch von Lehrmaterialien immer leichter zugänglich und immer besser nutzbar. Diese Entwicklung bietet hohes Potenzial für den Austausch und die Nutzung von heute größtenteils verteilt vorliegenden Lehrmaterialien.

[8] Einzelne in einem Text genutzte Worte sind selten repräsentativ für die Bedeutung des Textes.

Gerade beim vermehrten Austausch von Lernobjekten muss es eine Möglichkeit geben, wie sich Lernobjektanbieter und Lernobjektnachfrager gegenseitig mitteilen können, wie ein bestimmtes Lernobjekt aufgebaut ist und wie dieses Lernobjekt genutzt werden kann. So können Lernobjekte, die konsequent mit Metadaten ausgestattet sind, systemübergreifend eingesetzt und ausgetauscht werden.

Gewinnung, Auswertung und Anpassung von Metadaten

Metadaten werden heute meistens von Autoren manuell oder halbautomatisch generiert und dann in die Lernobjekte eingebettet. „Halbautomatisch" bedeutet, dass ein Teil der Metadaten manuell erzeugt wird, während der andere Teil durch algorithmische und logische Auswertungen bis zu einem gewissen Grad automatisch generiert werden kann. Eine vollautomatische Erzeugung der Metadaten gibt es bisher in der Regel nicht.

Bei der Auswertung und Anpassung von Metadaten liegt eine analoge Situation vor. Im einfachsten Fall werden Metadaten von den Autoren und Nutzern gelesen. Bei anderen Ansätzen werden die Metadaten maschinell ausgewertet, beispielsweise bei der Suche nach Lernobjekten mit bestimmten, in Metadaten erfassten Eigenschaften. Solche Suchverfahren werden oft um die Auswertung von Regeln, die auf die Metadaten angewendet werden, oder um die Ausführung von Operationen auf den Metadaten ergänzt. Danach wird durch die Analyse des Verhältnisses von Anforderungen und Ergebnissen versucht, die Treffermenge dieser Suche weiter einzuschränken und abzugleichen. An dieser Stelle werden die interessierten Leser auf [Bal98] verwiesen.

2.5.2 Ausgewählte Metadaten-Standards für Lehrmaterialien

Metadaten sollen nicht nur jede Form von Daten beschreiben können, sondern – zumindest innerhalb eines Anwendungsbereichs, beispielsweise Lehrmaterialien – auch einheitlich angewendet werden können. Derzeit weisen die Metadaten, mit denen die Lernobjekte beschrieben werden, eine starke Heterogenität sowie auch Interoperabilitätsprobleme auf, vgl. auch [SICT]. Dies resultiert daraus, dass die Lernobjekte oft von verschiedenen Autoren oder Institutionen, die unterschiedliche Systeme mit jeweils eigenem Metadatenvokabular nutzen, stammen. Damit Metadaten ihren Zweck erfüllen können, müssen die Benennung und die Wertebereiche von Metadaten sowie ihre Strukturierung standardisiert werden, um Lernobjekte austauschen und projekt- und produktübergreifend verknüpfen und integrieren zu können.

Ein Standard stellt bestimmte Mindesteigenschaften von Produkten, Methoden oder Abläufen sicher und dient der Vereinfachung und Vereinheitlichung bei der Erstellung und Nutzung dieser Produkte, Methoden und Abläufe [Mon04]. Er liefert die Voraussetzung für sowohl syntaktische als auch semantische Interoperabilität von Lernobjekten. IEEE[9], AICC[10], IMS[11], DIN[12], ISO[13] und andere Organisationen bemühen sich um Standardisierung, auch im Bereich Lehrmaterialien. Mittlerweile gibt es eine Vielzahl von Metadatenstandards im Lehrmaterialumfeld. Beispiele hierfür sind das Alliance of Remote Instructional Authoring and Distribution Networks of Europe (ARIADNE)-Modell, das Learning Objects Metadata-Modell (LOM), das Instructional Management Systems-Modell (IMS), das Shareable Content Object Reference Model (SCORM) und die Advanced Distributed Learning (ADL) Initiative des US-amerikanischen Verteidigungsministeriums [ADL]. In dieser Arbeit wird kurz

[9] Institute of Electrical and Electronics Engineers
[10] Aviation Industry Computer Based Training Committee
[11] Instructional Management Systems
[12] Deutsche Industrienorm
[13] International Standards Organisation

auf ARIADNE-, LOM- und IMS-Metadaten eingegangen. Diese drei Standards sind mittlerweile untereinander kompatibel.

Das Modell der Alliance of Remote Instructional Authoring and Distribution Networks of Europe (ARIADNE)

ARIADNE ist ein seit 1995 von der EU-Kommission gefördertes Projekt. Ziel des Projekts ist es, den Austausch von Lernobjekten zwischen verschiedenen Bildungseinrichtungen zu fördern. In Kooperation mit 20 europäischen Universitäten und fünf internationalen Firmen wurden Metadatenkategorien speziell für Lehrmaterialressourcen geschaffen.

ARIADNEs Ziel ist es, einen europaweiten Informationspool bildungsbezogener Lehrmaterialien in sogenannten Knowledge Pool Systems (KPS) zu verwalten, die multilingual sind, einfache Indexierungsverfahren benutzen und mit einfachen Tools effizient durchsucht werden können. Jedes KPS wird von mehreren Benutzergruppen genutzt. Auf der einen Seite sind die Presenter und Autoren, die auf das KPS zugreifen und Inhalte erstellen und pflegen. Auf der anderen Seite sind einzelne Lerner und Lerngruppen, die die Inhalte der KPS lesen und durcharbeiten [Schö01], [Tra00].

Für das Erstellen, Bearbeiten und Abrufen von Ressourcen für die KPS wurden eine Reihe von Tools entwickelt: Simulation Authoring Tool, Questionnaire Authoring Tool, oder Pedagogic Hypertext Generator, etc. [Schö01].

ARIADNE definiert für die Beschreibung der in den KPS enthaltenen Ressourcen folgende Metadatenkategorien:

- Allgemeine Informationen über eine Ressource
- Semantische Aspekte einer Ressource
- Pädagogische Attribute zur Spezifikation des pädagogischen Umfelds, in welchem die Lernobjekte eingesetzt werden können.
- Technische Charakteristika , beispielsweise das Betriebssystem
- Nutzungsbedingungen (Rechte, Preise)
- Daten über die Metadatenerstellung (Meta-Metadaten)
- Vom Hersteller autorisierte Anmerkungen und Informationen über die Nutzungsmöglichkeiten der Lernobjekte

Das Learning Objects Metadata-Modell (LOM)

Learning Objects Metadata (LOM) ist ein von einer Arbeitsgruppe des IEEE LTSC (Learning Technology Standardization Committee) [LTSC] auf der Grundlage von Dublin Core [Dub b] und ARIADNE geschaffener Standardisierungsvorschlag mit dem Ziel, Lernobjekte zu verwalten, zu lokalisieren, zu beschreiben, auszutauschen und auszuwerten.

Die Idee hinter LOM ist eine hierarchische Organisation der Metadaten. An der Spitze des Metadatenbaumes steht dabei das Root-Element, das viele Unterelemente enthalten kann. Datenelemente können in LOM jeweils vorgeschrieben, optional oder bedingt angegeben sein. Datenelemente können beliebig tief geschachtelt werden und der Wert eines Datenelements kann wiederum ein Datenelement sein. Unterschiedliche Datenelemente haben einen festen Wertebereich und sind Kategorien –Zusammenfassungen logisch zusammenhängender Metadaten – zugeordnet. Kategorien und Datentypen können erweitet werden.

Derzeit enthält LOM 80 Attribute, durch die teilweise das Umfeld der Lernumgebung, repräsentiert durch Eigenschaften der Lernobjekte, abgedeckt worden ist.

Im Folgenden sind die LOM-Kategorien des Basisschemas dargestellt, mit dem allgemeine Eigenschaften von Lernobjekten beschrieben werden können. Unter den jeweiligen Kategorien sind standardisierte Metadaten mit zugeordneten Wertebereichen definiert. Derzeit besteht das LOM-Basisschema aus neun Kategorien:

- General Category: Grundlegende Informationen, die das Lernobjekt als Ganzes beschreiben
- Lifecycle Category: Merkmale, die sowohl die Historie und den aktuellen Zustand des Lernobjekts beschreiben, als auch die Lernobjekte, die von den gerade betrachteten beeinflusst werden
- Meta-Metadata Category: Informationen über die Metadateninstanz an sich
- Technical Category: Technische Voraussetzungen und Merkmale des Lernobjekts
- Educational Category: Pädagogische Merkmale des Lernobjekts
- Rights Category: Urheberrecht und Nutzungskonditionen des Lernobjekts
- Relation Category: Beziehungen zwischen dem Lernobjekt und anderen verwandten Lernobjekten
- Annotation Category: Anmerkungen über den Bildungsnutzen des Lernobjekts und Informationen über die Entstehung der Kommentare („wann“, „von wem“)
- Classification Category: Einordnung des Lernobjekts in ein Klassifizierungssystem

Mit LOM können nicht nur Daten beschrieben werden, sondern auch Beziehungen zwischen den Lernobjekten. Folgende Beziehungsarten sind möglich:

- Hierarchische Beziehungen:

  Über hierarchische Beziehungen werden Lernobjekte zu einem größeren Lernobjekt miteinander verbunden. Des weiteren können die Lernobjekte dann auch anhand ihrer Beziehungen untereinander mittels eines Information Retrieval-Dienstes abgefragt werden.

  Beispiel: „besteht aus“-Beziehung

- Referenz:

  Mit Referenzen lassen sich inhaltliche Beziehungen zwischen Lernobjekten modellieren.

  Beispiele: „bestimmt“, „neben“, „Gegensatz zu“, „vor“, „Kontext von“, „Prozess von“, „bewertet“, „Mittel von“, „Affinität“, „Voraussetzung“

Wichtigstes Ziel von LOM ist es dem Anwender zu ermöglichen, Lernobjekte zu suchen, auszuwerten und zu nutzen. Des weiteren soll modulares Entwerfen und Kombinieren von Lernobjekten (etwa durch intelligente Agenten, die einen Wissensstand und ein Lernziel genannt bekommen und daraus Lektionen zusammenstellen) gefördert werden. Im Vergleich zu ARIADNE hat LOM seine Stärken in der Modellierung von Inhalten von Lernobjekten und Lernerinformationen. Der wesentliche Vorteil von LOM ist, dass es mittelfristig von der IEEE als Standard verabschiedet werden wird.

Das Instructional Management Systems-Modell (IMS)

Die National Learning Infrastructure Initiative (NLII) ist ein Verband aus Institutionen und Organisationen aus dem Bildungsbereich und aus Industriepartnern. Eines der bekanntesten Arbeitsergebnisse der NLII ist die IMS Metadata Specification zur Beschreibung von Metadaten von Lerninhalten. Diese Spezifikation bildet die Basis für den im Rahmen des IEEE LTSC-Konsortiums entwickelten LOM-Standard, 19 von 80 Elementen der LOM-Metadaten werden identifiziert. Zusätzlich zu LOM beschreibt die IMS-Spezifikation [IMS a], [IMS b] Typisierungen einiger Attribute, wodurch ein weiterer Schritt zur Interoperabilität zwischen Lernobjekten geleistet wird.

Die IMS-Spezifikation ist ein technischer und kein akademischer oder pädagogischer Standard. Grundlage der IMS-Spezifikation ist ein IMS Content Package. Ein IMS Content Package besteht aus zwei Teilen: Einem XML-Dokument (Manifest), welches den Aufbau und den Inhalt des Pakets beschreibt und den Dateien, auf die im Manifest verwiesen wird. IMS definiert die kleinste Einheit als Physical File. Physical Files sind atomare Einheiten wie Bilder oder Texte. Zusammenhängende Physical Files werden zu einem Package Interchange File zusammengesetzt. Die hierzu benötigten Strukturinformationen sind im Manifest gespeichert. Abbildung 2-5 illustriert das Prinzip.

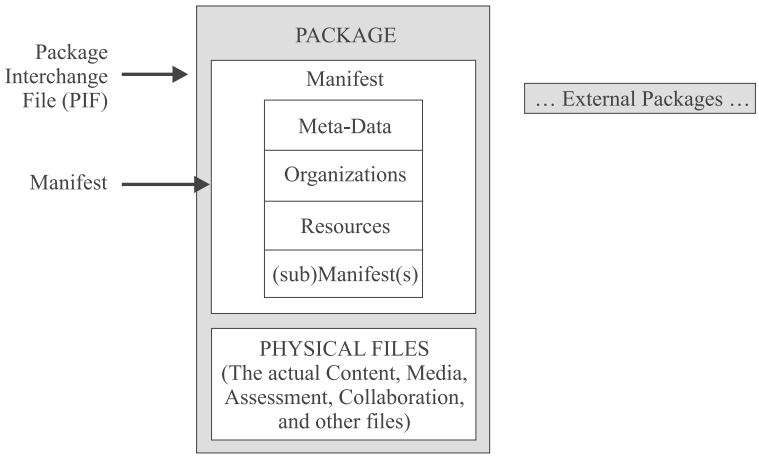


Abbildung 2-5: Aufbau eines IMS Package Interface Files, nach [IMS a]

Beobachtungen

Standards allein genügen nicht den Anforderungen zur domänenübergreifenden und domänenspezifischen Wiederverwendung von Lernobjekten (vgl. 2.4). Die Standards sind mehr auf den Datenaustausch zwischen verschiedenen Werkzeugen ausgerichtet und enthalten nur wenige Metadaten, die helfen würden, das Verständnis der Daten der Lernobjekte für die Wiederverwendung zu verbessern. Dies zeigt zugleich auch eine Übersimplifizierung: Die genannten Standards decken nicht alle geforderten Eigenschaften von Lernobjekten (vgl. 2.2.2) und Beziehungen zwischen ihnen in Bezug auf Erstellung- und Nutzungsszenarien bzw. -arten ab:

- Die beschriebenen Standards umfassen meist nur die Beschreibung des Inhalts. In der Zuordnung zu didaktischen Aktivitäten und der Abbildung auf das Lernen (beispielsweise bei der Beschreibung von Lernpfaden) dagegen zeigen die Standards Schwächen und mangelnde Ausdruckskraft. Ebenfalls fehlen Informationen über die möglichen Nutzungsarten, für die die Daten erzeugt sind und benutzt werden.

- Die genannten Standards bieten keine Möglichkeit, verschiedene subjektive Einordnungen und Beschreibungen von Lernobjekten[14] sowie eine domänenabhängige Semantik von Lernobjekten auszudrücken.
- Metadaten in den beschriebenen Standards sind meistens größeren Lerneinheiten zugeordnet: Es fehlen Metadaten für feingranulare Lernobjekte.

2.5.3 Erweiterungen der Standards

Um den oben genannten Beobachtungen und Problemen zu begegnen und um eine weitgehende Abdeckung aller Lernobjekte und aller Relationen zwischen ihnen bei der Erstellung und Nutzung der Lehrmaterialien sicherzustellen, versuchen zahlreiche Forschungsprojekte aus verschiedenen Aspekten (beispielsweise Nutzungsszenarien) heraus, diese Standards zu erweitern, indem neue Kategorien und Datenelemente zum Basisschema hinzugefügt werden [CaHo01], [SüFr99].

Bei den Erweiterungen wurde meistens auf verfügbare Standards aufgesetzt, um die Offenheit, Erweiterbarkeit und Plattformunabhängigkeit der Lernobjekte nach wie vor zu gewährleisten. Im folgenden werden einige typische Erweiterungen erläutert:

Ergänzung um Ontologien

Bei diesem Ansatz werden die Standards um Kategorien mit referenzierten Wissensquellen angereichert. In diesen Kategorien wird ein Vokabular zu bestimmten Themen definiert und inhaltliche Aspekte auf einheitliche Attributmengen und deren Wertebereiche abgebildet.

Darüber hinaus werden domänenspezifische Kategoriesysteme – Ontologien oder auch semantische Netze – geschaffen, um eine fachsystematische Klassifikation der Lernobjekte zu ermöglichen, vgl. [SemDok04]. Diese Erweiterung trägt dazu bei, dass zwischen Lernobjekten eine vielfältige semantische Vernetzung ermöglicht wird. Zugleich unterstützt sie die Genauigkeit beim Suchen und Auswerten der Metadaten durch die Nutzer.

Beispiel: Teachware on Demand

> Im Rahmen des Projekts „Teachware on Demand“ werden zwei Erweiterungsstufen des LOM-Basisschemas vom Fraunhofer-Institut für Software- und Systemtechnik ISST festgelegt: Erweiterung themenunabhängiger Kategorien und Attribute von LOM, wie beispielsweise die Kategorie „Presentation“ oder „Rights“, und Hinzufügen IT-spezifischer Attribute zu den LOM-Kategorien „Technical und Educational“. Dabei wurden nicht nur Lernobjekte, sondern auch die referenzierten Inhalte in Form von Konzepten, welche Stichworten bzw. Themen gleichgesetzt sind, modelliert. Das vom LOM-Basisschema abgeleitete Teachware-Schema wird um eine Kategorie „index“ erweitert, welche aus einer eindeutigen Identifizierung der für die Konzeptkodierung verwendeten Ontologie und einer Auflistung aller Beziehungen zwischen dem Lernobjekt und Konzepten der Ontologie besteht. Anhand der Ontologie lassen sich potenziell inhaltliche Abhängigkeit zwischen Lernobjekten berechnen. Näheres dazu siehe [CaHo01] und [ToD].

Ergänzung um Inhaltstypen

Die Typisierung von Lernobjekten und die Konfiguration wiederverwendbarer Lernobjekte liefern eine weitere Ausgangsbasis für Erweiterungen. Dazu werden Inhaltstypen auf den

[14] Damit sind Beschreibungen von Lernobjekten gemeint, die nur schwer objektiv fassbar sind. Ein Beispiel wäre die Beschreibung des Schwierigkeitsgrades. Angenommen, es wird eine Skala von 1 (leicht) bis 5 (schwer) der Bewertung zu Grunde gelegt, so ist davon auszugehen, dass verschiedene Autoren ein und dasselbe Lernobjekt unterschiedlich bewerten.

Lernobjekten spezifiziert, über die dann implizit – über Schachtelung („ist enthalten in"- oder „Referenz zu"-Relation) – inhaltliche Beziehungen zwischen den Lernobjekten abgeleitet werden. Über die so entstehenden strukturellen Beziehungen zwischen Lernobjekten werden zugleich auch didaktische Beziehungen realisiert.

Beispiel: Das Passauer Knowledge Management System (Beispiel 1) [SüFr99]

> Das Passauer Knowledge Management System (Pakmas) wird vom Lehrstuhl für Informationsmanagement der Fakultät für Mathematik und Informatik der Universität Passau entwickelt. Es unterstützt die Erstellung von wiederverwendbaren Lehrmaterialien und vor allem die Wiederverwendung von Inhalten und Strukturen. Pakmas ist ein Authoring-System, das Autoren- und Management-Umgebung integriert.
>
> Kernpunkt von Pakmas sind konzeptuell modellierte, Semantik tragende Inhaltsobjekte, welche wiederum aus verschiedenen einfachen und komplexen Daten- bzw. Medientypen bestehen. Auf diesen Inhaltsobjekten werden konzeptuell modellierte (inhaltliche) Beziehungen aufgebaut. So bilden beispielsweise die Inhaltsobjekte „proof" und „proposition" die Beziehung „Proof proves proposition". In diesem Zusammenhang entstand die Learning Material Markup Language (LMML) [LMML05], eine XML-Sprache, um Inhaltsobjekte und deren Struktur syntaktisch mittels DTDs zu repräsentieren. Aufgrund der umfangreichen metadatenbasierten Filter- und Anfragemöglichkeiten können Lernobjekte leicht aufgefunden, eingefügt und neu kombiniert werden. Für Details zu Pakmas sowie LMML wird an dieser Stelle auf [SüFr99] bzw. [SüFr01] verwiesen.

Ergänzung um spezifizierte und personalisierte Nutzungen

Ein anderer Gesichtspunkt von Lehrmaterialien sind nutzer- und nutzungsspezifische Aspekte. Autoren konfigurieren bestehende Lernobjekte anhand von Metadaten, während Nutzer diese Lernobjekte gemäß der spezifizierten Nutzungen adaptieren. Um die Standards für verschiedene Nutzergruppen gemäß ihrer Eigenschaften, Aufgaben und Wünsche zu erweitern, werden Metadaten für typische Nutzergruppen, wie beispielsweise bestimmte, spezifizierte Zielgruppen, entsprechend ergänzt. So lassen sich die bestehenden Standards, die meist nur inhaltliche Aspekte umfassen, um die angeforderten Nutzungen erweitern.

Beispiel: Das Passauer Knowledge Management System (Beispiel 2)

> Das Konzept von Pakmas ermöglicht die lokale Pflege und Aktualisierung sowie die Wiederverwendung und die statische Anpassung an verschiedene Zielgruppen und unterschiedlichste Anwendungen in der Lehre. In Pakmas sind Lehrinhalte und Anwendungen voneinander getrennt. Dies erlaubt eine statische Anpassung von Lehrmaterialien an die individuellen Bedürfnisse der Lerner. Die Präsentations-, Navigations- und die thematische Anwendung werden adaptiv auf Inhaltsobjekten aufgebaut, indem sie anwendungsorientiert ausgewählt, angepasst und kombiniert werden können.

Ergänzung um Präsentation und Visualisierung

Metadaten werden gegenüber den Standards auch um Präsentationsstile wie Darstellungsform, Layout, etc. ergänzt, um so eine Wiederverwendung bezüglich der Darstellung zu erreichen. So wurde versucht, mehrere Präsentationsstile mit Hilfe erweiterter Metadatenstandards für Lehrmaterialien gleichen Inhalts und gleicher Struktur zu definieren und zu spezifizieren, um so in der Präsentation und der Visualisierung der Lehrmaterialien flexibler zu sein. In den meisten Fällen sind alle für die zu unterstützenden Präsentationsformate benötigten Informationen zur visuellen Gestaltung im Modell und / oder in der Grammatik repräsentiert. In diesen Fällen kann das Modell beispielsweise mit XSL- oder CSS-Stylesheets dynamisch in die verschiedenen Präsentationsvarianten transformiert werden.

Beispiel: Automatic Page Layout System (APALO) [KlRe97]

Das Programm APALO (Automatic Page Layout System) generiert automatisch Seitenlayouts für multimediale Elemente wie Text, Video, Grafik, Bild, etc. nach typografischen und kognitiven Gesichtspunkten. Grundlage für die Bestimmung des Layouts sind Standard Generalized Markup Language (SGML)-Dokumente, die die Layoutstruktur der zu präsentierenden Inhalte beschreiben. So ein SGML-Dokument beschreibt beispielsweise, welche Reihenfolge für die darzustellenden multimedialen (Wissens-)Elemente besteht oder welche Elemente wie gruppiert sind. Wissen über die Gruppierung oder die Reihenfolge sind hierbei entscheidend für die Güte des Seitenlayouts.

2.5.4 Wiederverwendung mittels Transformationsregeln

Bei diesem Ansatz werden Regeln definiert, die ausgeführt werden müssen, um Lernobjekte auf der Grundlage konsistenter Metadaten zu modifizieren. Diese Regeln beschreiben Anpassungsschritte, die eine Wiederverwendung der Lernobjekte beispielsweise in einer anderen domänenspezifischen Anwendung erlauben. Derartige Auswertungs- und Anpassungsschritte können mit Algorithmen darstellbar sein und mit Hilfe von Generatoren (beispielsweise XSLT-Prozessoren) implementiert werden.

Beispiel: Task-based Adaptive LearNer Guidance On the WWW (TANGOW) [CPR99]

TANGOW, entwickelt von der Universidad Autonoma de Madrid, konzentriert sich auf Wiederverwendung eines Kurses durch Lernerinteraktionen. TANGOW gehört streng genommen zu den sogenannten Lernsystemen. (Ein Lernsystem besteht hier aus Informationssystem, Selbstlern-, Übungs- und Gruppenlernumgebung. Für Details siehe 6.2.1.) TANGOW erlaubt es, Kurse sowohl anhand statisch analysierbarer Nutzerdaten als auch anhand dynamischer Nutzerinteraktionen anzupassen.

Die Kursentwicklung bei TANGOW beruht auf *Lernaufgaben* (tasks) und *Lernregeln* (rules). Die Lernaufgaben sind auf die Konzepte ausgerichtet, die von den Lernern erlernt werden sollen, während die Lernregeln die Art und Weise festlegen, wie diese Konzepte miteinander verbunden sind. Die Lernaufgaben können auch einander zugeordnete Inhalte eines Konzepts umfassen, die jedes beliebige Multimediaelement enthalten können: Texte, Bilder, Video, Ton oder Simulations-Applets.

In TANGOW werden Nutzerprofile der Lerner aufgebaut, um für den Lerner oder die Lerngruppe personalisierte Lehrmaterialien anzubieten. Neben den vorspezifizierten statischen Inhalten passt das System dynamisch während des Lernprozesses die Informationen an, die jedem Lerner angeboten werden. Die Anpassung der Lehrmaterialien erfolgt hierbei sowohl im Hinblick auf die Eigenschaften, die im Profil des Lerners vermerkt sind, als auch basierend auf den Ergebnissen der Interaktion des Lerners mit dem System. Ein personalisierter Kurs wird dann im Rahmen vorgeplanter und strukturierter tasks und rules diesem Lerner angeboten.

Weitere Adaptierungen sind möglich, sofern sie von den Kursautoren (als Vorlage) vorgesehen wurden. Die Kurse werden dann aus dem Inhalt, der Struktur und den Präsentationselementen, aus denen die Kurse bestehen, zusammengestellt.

2.6 Szenarien der Wiederverwendung von Lernobjekten

Wiederverwendung von Lernobjekten kann in verschiedenen Situationen stattfinden: Bei der Erstellung neuer Lernobjekte und bei der Nutzung der Lernobjekte für verschiedene Lehrveranstaltungen. Beide Situationen werden in den folgenden Abschnitten betrachtet.

2.6.1 Wiederverwendung bei der Erstellung der Lernobjekten

Um ein Lernobjekt zu erstellen, sind der Top-Down-Ansatz und der Bottom-Up-Ansatz nutzbar. In der Praxis werden beide Ansätze gleichberechtigt und oft auch verschränkt eingesetzt; eine Entwicklung von Lernobjekten, die ausschließlich nach dem Top-Down- oder dem Bottom-Up-Ansatz durchgeführt wird, ist vergleichsweise selten.

Szenario 1:

> Autor *A* will Lehrmaterialien zur Vorlesung „Einführung Rechnernetze“ für das Grundstudium erstellen. Als Präsentationsformat für die Lehrmaterialien wird die Hypertext Markup Language (HTML) gewählt. Ziel der Vorlesung ist es, den Studenten einen Überblick über das Thema „Rechnernetze“ zu vermitteln.
>
> Es ist bereits ein alter Kurs „Technische Informatik“ für das Hauptstudium verfügbar, der ebenfalls in HTML vorliegt. In diesem wird neben anderen Themen auch detailliert auf das Thema „Rechnernetze“ eingegangen, hier unter anderem auf Themen wie Ethernet, ISO OSI-Schichtenmodell, Netzmanagement, etc. Autor *A* will nun auf Basis der Lehrmaterialien des Kurses „Technische Informatik“ neue Lehrmaterialien für die Vorlesung „Einführung Rechnernetze“ erstellen.

Folgende Anforderungen an den zu erstellenden Kurs können aus dem Szenario gewonnen werden:

- Der bereits bestehende Kurs ist in seinem Inhalt („Rechnernetze“ im Bereich „Technische Informatik“) und seinem Aufbau (Kursstruktur) abgeschlossen und liegt in einem einheitlichen Präsentationsformat und -stil vor. Ferner ist er nutzungsspezifisch (für das Hauptstudium).
- Der neue Kurs soll einen abgeschlossenen Inhalt haben, welcher durch sein Thema („Einführung Rechnernetze“) vorgegeben ist.
- Der neue Kurs hat eine bestimmte, vorgegebene Nutzung (Grundstudium, das heißt im wesentlichen überblicksartige Wissensvermittlung).
- Beide Kurse gehören zur selben Inhaltsdomäne „Technische Informatik“.
- Beide Kurse gehören zur selben Anwendungsdomäne („Präsenzveranstaltung“), haben aber eine unterschiedliche Nutzung (Grundstudium vs. Hauptstudium).

Im folgenden wird gezeigt, wie die benötigten Lernobjekte bei Verwendung des Top-Down-Ansatzes respektive der Verwendung des Bottom-Up-Ansatzes erstellt werden können.

2.6.1.1 Entwurf von Lernobjekten mit dem Top-Down-Ansatz

Schritt 1: Soll-Analyse

Beim Top-Down-Ansatz wird als erstes eine Anforderungsanalyse der beabsichtigten Nutzung durchgeführt. Ziel ist es, die Soll-Situation akkurat zu beschreiben. Im Fall von Szenario 1 muss vor allem das neue Kursziel „Überblick über das Thema Rechnernetze“ und die Zielgruppe „Studenten im Grundstudium“ festgestellt werden. Diese Randbedingungen sind fest vorgegeben und nicht veränderbar. Sie dienen zur Identifikation von Suchkriterien, anhand derer passende Lernobjekte für den neuen Kurs ermittelt werden.

Schritt 2: Entwurf der Kursstruktur

Auf der Soll-Analyse aufbauend, wird nun die Kursstruktur entsprechend den Anforderungen entworfen. Zuerst wird nur der Kursrahmen, also die reine Struktur ohne die Lernobjekte, die die Inhalte tragen, aufgebaut. Möglich sind hier Hypertextstrukturen, hierarchische

Strukturen (beispielsweise Kapitel – Unterkapitel), etc. In Szenario 1 soll der Kurs für das Grundstudium konzipiert werden. Hierzu genügt es, ihn aus Lernobjekten der Typen „Einführung“, „Motivation“, „Grundlagen“, „Beispiel“ und „Zusammenfassung“ aufzubauen. Komplexere theoretische Beweise oder Algorithmen können entfallen. Dies definiert die Struktur des neuen Kurses. Man beachte, dass hier lediglich die Struktur definiert wird; eine Auswahl konkreter Lernobjekte und damit die Auswahl konkreter Inhalte erfolgt hier noch nicht, sondern erst in Schritt 3.

Schritt 3: Suche und Auswahl von Lernobjekten

Steht die Struktur des Kurses fest, wird der Kurs mit Inhalten gefüllt. Hierzu müssen passende Lernobjekte gefunden und ausgewählt werden. Passende Lernobjekte kann man finden, indem man die gesuchten Lernobjekte Schritt für Schritt genauer spezifiziert. In Szenario 1 könnte man beispielsweise folgende Suchanfragen stellen:

- Suche nach einem bestimmten Datentyp: „Suche nach allen Bildern.“
- Suche nach einem bestimmten Datentyp mit bestimmtem Inhalt: „Suche nach allen Bildern, die das Hot Potato-Verfahren zum Routing darstellen.“

In Szenario 1 würden dann Lernobjekte des bereits bestehenden Kurses gefunden werden. Diese Lernobjekte kämen dann für den neuen Kurs in Frage.

Je genauer die gesuchten Lernobjekte spezifiziert werden, desto präziser können passende Lernobjekte gefunden werden, aber desto kleiner ist die Ergebnismenge. Im Extremfall kann gar kein Lernobjekt gefunden werden, das den Suchkriterien entspricht; die Ergebnismenge ist dann leer.

Werden in diesem Schritt keine passenden Lernobjekte gefunden, müssen passende Lernobjekte neu erstellt werden.

Schritt 4: Anpassung der ausgewählten Lernobjekte

Da der neue Kurs für eine andere Zielgruppe als der alte Kurs erstellt wird, können die bei der Suche gefundenen Lernobjekte nicht einfach übernommen werden. Sie müssen auf ihre Eignung überprüft und gegebenenfalls angepasst werden. Die Anpassung kann auf verschiedene Weisen erfolgen; Möglichkeiten hierzu sind in 2.4.3[15] genannt. Zur beachten ist hierbei, dass diese Möglichkeiten auch kombiniert durchgeführt werden können. Beispielsweise können neben der Konfiguration der bestehenden Lernobjekte auch neue Lernobjekte entwickelt werden, die die in Schritt 2 definierte Kursstruktur erfüllen.

Es ist möglich, dass im Rahmen einer Anpassung eine neue Suche durchgeführt werden muss. Die Schritte „Suche und Auswahl von Lernobjekten“ und „Anpassung der ausgewählten Lernobjekte“ werden so rekursiv durchlaufen, und zwar so lange, bis die in der Soll-Analyse identifizierten Anforderungen mit den Eigenschaften der ausgewählten Lernobjekte abgeglichen sind.

2.6.1.2 Entwurf von Lernobjekten mit dem Bottom-Up-Ansatz

Analog zum Top-Down-Ansatz bestimmt auch beim Bottom-Up-Ansatz die Anforderungsanalyse den Entwurf des neuen Kurses. Die Unterschiede zwischen dem Top-Down-Ansatz und dem Bottom-Up-Ansatz liegen im weiteren Vorgehen: Während beim Top-Down-Ansatz die Strukturierung vom Generellen zu den speziellen Themen und Nutzungen erfolgt, geht der Bottom-Up-Ansatz von den bereits bestehenden Lernobjekten aus. Diese werden zu neuen Lernobjekten kombiniert, falls nötig modifiziert, umkonfiguriert oder um weitere

[15] Zur Erinnerung: Austausch, Weiterentwickelung, Konfiguration, Adaptierung.

Lernobjekte ergänzt, und bauen so Lernobjekt für Lernobjekt den neuen Kurs auf. Die folgenden Schritte werden beim Bottom-Up-Ansatz durchgeführt:

Schritt 1: Soll-Analyse

Die Soll-Analyse wird analog zum Top-Down-Ansatz durchgeführt. Vgl. dort.

Schritt 2: Dekomposition bereits bestehender Kurse

Die bereits bestehenden Kurse werden – falls noch nicht geschehen – bis auf die unterste Hierarchiestufe in einzelne Lernobjekte zerlegt. Als Ergebnis erhält man eine Menge von Lernobjekten mit bereits durch Metadaten oder Transformationsregeln beschriebenen bestimmten Eigenschaften und Nutzungsbedingungen.

Schritt 3: Suche, Auswahl und gegebenenfalls Erstellung von Lernobjekten

Basierend auf den Anforderungen des neu zu erstellenden Kurses einerseits und den bereits verfügbaren Lernobjekten andererseits wird ein üblicherweise inhaltsbasiertes Suchen durchgeführt. In Szenario 1 könnten dies folgende Suchen sein:

- Suche nach Lernobjekten mit bestimmtem Inhalt (Volltextsuche oder metadatenbasierte Suche): „Suche alle Lernobjekte mit dem Thema Rechnernetze."
- Suche nach Lernobjekten mit bestimmtem Inhalt mit gewissen inhaltlichen Merkmalen (Kombination von Volltextsuche und metadatenbasierter Suche oder rein metadatenbasierte Suche): „Suche alle Lernobjekte mit dem Thema Rechnernetze, deren Detaillierungsgrad gering ist".
- Suche nach Lernobjekten mit bestimmtem Inhalt und einer bestimmten syntaktischen oder semantischen Struktur (Kombination von Volltextsuche und metadatenbasierter Suche oder rein metadatenbasierte Suche): „Suche nach Lernobjekten, die eine detaillierte Beschreibung zum Thema Ethernet im Einführungskapitel enthalten".

Falls keine passenden Lernobjekte gefunden werden, müssen Lernobjekte erstellt werden, die die geforderten Eigenschaften besitzen.

Schritt 4: Anpassung der ausgewählten Lernobjekte

Dieser Schritt wird analog zum Top-Down-Ansatz durchgeführt. Vgl. dort.

Schritt 5: Aufbau inhaltlicher Beziehungen zwischen Lernobjekten

In diesem Schritt werden mehrere Lernobjekte, die in den Schritten 3 und 4 ausgewählt und geeignet angepasst wurden, zu einem in sich abgeschlossenen, komplexeren Lernobjekt kombiniert. Die Kombination der einzelnen Lernobjekte erfolgt anhand inhaltlicher oder struktureller Beziehungen zwischen den Lernobjekten. Wenn beispielsweise bekannt ist, dass ein Lernobjekt *A* ein anderes Lernobjekt *B* als Vorwissen benötigt, kann zwischen den beiden Lernobjekten eine Vorgänger-Nachfolger-Beziehung abgeleitet werden. Werden nun Lernobjekte *A* und *B* in einem komplexeren Lernobjekt *C* zusammengefasst, so kann die Vorgänger-Nachfolger-Beziehung zwischen *A* und *B* so realisiert werden, dass *B* in Lernobjekt *C* vor *A* aufgeführt wird.

Die Schritte 3 bis 5 werden nun so lange wiederholt durchlaufen, bis der neue Kurs aufgebaut ist.

2.6.2 Wiederverwendung bei der Nutzung von Lernobjekten

Die Wiederverwendung von Lernobjekten kann nicht nur bei der Erstellung neuer Lernobjekte erfolgen, sondern auch bei der Nutzung der Lernobjekte in Lehrveranstaltungen bzw. für das Selbststudium. Hier werden zwei Arten von Lernobjekten unterschieden: *Adaptierte* Lernobjekte und *adaptierbare* Lernobjekte. Adaptierte Lernobjekte werden im Vorfeld von ihren Autoren an die spätere, beabsichtigte Nutzungen manuell mittels geeigneter Werkzeuge oder (semi)automatisch unter Nutzung des bestehenden Nutzerprofils angepasst und dann von den Nutzern entsprechend verwendet. Die Anpassung der Lernobjekte erfolgt bei diesem Ansatz also vor ihrer tatsächlichen Nutzung. Bei dieser Art der Anpassung sind auch Änderungen an den Lernobjekten selbst möglich; hier steht die ganze Palette der in 2.4.3 beschriebenen Möglichkeiten der Anpassung zur Verfügung.

Bei adaptierbaren Lernobjekten hingegen nehmen die Nutzer – Dozenten, Lerner, etc. – die Anpassung an die von ihnen geplante Nutzung (innerhalb einer Anwendungsdomäne) vor. Bei dieser Art der Anpassung können Eigenschaften der Lernobjekte wie der Schwierigkeitsgrad, der Detaillierungsgrad, das Layout, etc. (vgl. 2.2.2) gesetzt oder verändert werden, um so gezielt verschiedene Nutzungen zu unterstützen; es können hingegen keine Änderungen an den Lernobjekten selbst, beispielsweise hinsichtlich ihrer Struktur, vorgenommen werden. Insofern kann man hier von Black Box-Wiederverwendung sprechen. Die Anpassung kann von den Nutzern manuell mittels entsprechender Werkzeuge oder automatisch durch Auswertung der Nutzerprofile der jeweiligen Nutzer oder beides kombiniert vorgenommen werden.

Szenario 2:

> Professor *P* will den Kurs "Einführung Rechnernetze" für das Grundstudium als Vorlesung präsentieren mit dem Ziel, dass die Studenten so einen Überblick über das Thema „Rechnernetze“ erhalten. Der Kurs soll innerhalb von 16 Wochen mit drei Semesterwochenstunden (SWS) pro Woche durchgeführt werden. Professor *P* bevorzugt es, die Inhalte den Studenten im wesentlichen anhand von Beispielen nahe zu bringen.

Folgende Anforderungen an den zu erstellenden Kurs können aus dem Szenario gewonnen werden:

- Der Kurs hat einen abgeschlossen Inhalt. Dieser ist durch das Thema „Rechnernetze“ vorgegeben.
- Der Kurs hat eine abgeschlossene Nutzung: Präsenzveranstaltung, Einsatz im Grundstudium, 16 Wochen zu drei SWS.
- Der Kurs ist für einen bestimmten Dozenten, dessen Lehrstil bekannt ist (Beispiele werden bevorzugt).

Generell können bereits bestehende Lernobjekte wie folgt adaptiert oder adaptierbar gemacht werden:

Adaption bezüglich Zielgruppe bzw. Wissensstand

Angenommen es existiert bereits ein Kurs „Einführung Rechnernetze“, der in Varianten sowohl für das Hauptstudium als auch für das Grundstudium genutzt werden kann. Die Zielgruppe und deren Wissensstand bilden die Ausgangsbasis dieser beiden Varianten, welche durch unterschiedliche Alternativen von Lernobjekten und / oder unterschiedliche Verlinkung der Lernobjekte realisiert sein können. In der Folge haben die Varianten den selben Inhalt, aber je nach Konfiguration in unterschiedlichem Detaillierungsgrad bzw. in unterschiedlicher Strukturierung. In Szenario 2 adaptiert der Dozent diejenigen Lernobjekte, um

die Kursvariante für das Grundstudium zu entwickeln bzw. er bekommt die Kursvariante für das Grundstudium adaptiert. Genau genommen führt der Dozent die Projektion vom domänenunabhängigen Kurs auf eine domänenabhängige Kursvariante aus. Dies geschieht, indem der Dozent die Eigenschaften des Kurses so wählt, wie er sie für den Einsatz des Kurses in seiner Lehrveranstaltung benötigt. Liegt eine passende Kursvariante bereits vor, wird diese ausgewählt. Anderenfalls führt das Setzen der Eigenschaften zur Erstellung einer neuen, passenden Kursvariante.

Adaption im Präsentationsformat

Für den Kurs „Einführung in Rechnernetze“ für das Grundstudium bieten sich zwei Präsentationsformate an: HTML und PDF. Im allgemeinen eignet sich HTML aufgrund der nichtlinearen Struktur besser für die Vorlesung, da so leichter auf Fragen und Anregungen der Studenten eingegangen werden kann. Der Kurs wird dann im HTML-Format erstellt (adaptiert).

Die Adaptierung von Lernobjekten ist nicht völlig frei; je nachdem, welches Ausgabemedium eingesetzt werden soll, müssen bestimmte Nebenbedingungen eingehalten werden. Wird beispielsweise geplant, ein vorlesungsbegleitendes Skript in PDF auf Papier gedruckt herauszugeben, ist man an eine sequenzialisierbare Struktur (Sequenz, Baum, etc.) der Lernobjekte gebunden. Ebenso sind in diesem Fall dynamische Inhalte wie Animationen oder Videos nicht möglich. Eine entsprechende Adaption des Kurses bezüglich der verwendeten Medien und bezüglich der Kursstruktur muss hier gegeben sein.

Adaption bezüglich der Individualität des Lehrers

Lehre ist ein methodischer Vorgang; man beschäftigt sich insbesondere mit der Fragestellung, wie der Lehrstoff eines Kurses den Studenten am besten beigebracht werden kann. Jeder Lehrer hat hier seine persönliche Vorlieben, was bedeutet, dass dem Lehrer eine gewisse Freiheit im Umgang mit dem ausgewählten Kurs gegeben werden muss, beispielsweise was die Geschwindigkeit, die Wahl der verwendeten Lernobjekte oder die Aufenthaltsdauer bei einem Lernobjekt betrifft. Im vorliegenden Szenario möchte Professor *P* seine Vorlesung entlang der im Kurs enthaltenen Beispiele gestalten; die Navigationspfade durch den Kurs sind also entsprechend zu adaptieren. Die Adaption kann von den Nutzern manuell mittels entsprechender Werkzeuge oder teilweise automatisch durch Auswertung der Nutzerprofile der jeweiligen Nutzer vorgenommen werden.

Adaption bezüglich der Benutzerinteraktion

Benutzerinteraktion wird hier im Sinne von Kommunikation des Dozenten mit den Lernern in der Lehrveranstaltung verstanden. Auch bezüglich der Interaktion mit den Nutzern muss der Kurs aus Szenario 2 adaptiert werden: Erwartet der Dozent einen hohen Grad an Interaktion mit den Studenten, muss die Navigation so gestaltet werden, dass er schnell zwischen den relevanten Inhalten hin und herspringen kann – es sind viele Querverweise zu nutzen. Zum Beispiel bietet es sich für den Kurs an, Querverweise zwischen den Beispielen anzulegen, Animationen anzuzeigen oder Selbsttestaufgaben durchzuführen.

2.7 Eignung der Ansätze für die Wiederverwendung von Lernobjekten

2.7.1 Bewertung der Ansätze

Objektorientierung, Entwurfsmuster, Frameworks, Metadaten und Regeln

Objektorientierte und komponentenbasierte Ansätze helfen, Lehrmaterialien zu modularisieren und zu strukturieren. Hier steht nicht die Kapselung von Verhalten und Eigenschaften im Vordergrund – eine zentrale Eigenschaft der Objektorientierung – sondern die Unterteilung und Strukturierung komplexer Datenbestände (hier: Lehrmaterialien) in kleinere, einfacher zu beherrschende Objekte (hier: Lernobjekte). Insofern sind für die Zwecke dieser Arbeit objektorientierte Ansätze und komponenten- bzw. Framework-basierte Ansätze gleichwertig. Durch die Aufteilung und Strukturierung, die diese Ansätze leisten, entstehen Lernobjekte, die allein schon aufgrund ihres deutlich abgegrenzten Umfangs deutlich leichter wiederverwendet werden können als umfangreiche Kurse oder ähnlich große Lehrmaterialien.

Metadaten und Regeln sowie ihre Repräsentation in Form von XML oder RDF bieten die technische Unterstützung, um Lernobjekte und Domänen zu beschreiben bzw. anwendungsspezifisch zu konfigurieren und zu adaptieren (vgl. auch [HeHe00]). Mit diesen Mitteln wird es möglich, aus der Menge der erstellten Lernobjekte gezielt solche auszuwerten bzw. zu nutzen, die für eine bestimmte geplante Nutzung in einer Lehrveranstaltung oder einem Kurs benötigt werden.

Beide Arten von Ansätzen gehen Hand in Hand: Die Lernobjekte, die mittels objektorientierter oder komponentenbasierter Ansätze identifiziert werden, können nur dann sinnvoll genutzt werden, wenn sie anhand verschiedener Aspekte identifiziert und miteinander kombiniert werden können. Dies ist durch Metadaten und Regeln möglich. Umgekehrt sind Metadaten und Regeln aber nur dann gewinnbringend einsetzbar, wenn eine ausreichend große Zahl an nicht zu speziellen und damit gut wiederverwendbaren Lernobjekten bereitsteht, denn nur wenn die verfügbaren Lernobjekte auch tatsächlich wiederverwendet werden können – und das ist nur möglich, wenn die Lernobjekte nicht zu groß und nicht zu speziell sind – lohnt es sich, die Lernobjekte mit Metadaten, etc. zu beschreiben.

Standards und Erweiterung

Standards decken längst nicht alle Eigenschaften und Szenarien zur Erstellung und Nutzung von Lernobjekten ab (vgl. Beobachtungen in 2.5.2). Durch verschiedene Erweiterungen (Ontologien, Inhaltstypen, etc.) der Standards konnte es erreicht werden, dass nun reichhaltige konzeptionelle Darstellungen der Eigenschaften wiederverwendbarer Lernobjekte durch erweiterte Metadaten bzw. Gruppen von Metadaten sowie durch die entwickelten Grammatiken vorliegen. Die Wiederverwendung dergestalt beschriebener Lernobjekte geht jedoch meist von einer ausgewählten und spezifizierten Anwendungsdomäne (eine bestimmte Nutzungsart, ein bestimmtes Nutzungsszenario bzw. eine bestimmte Nutzergruppe) aus mit dem Ergebnis, dass meist für bestimmte Anwendungen benötigte Eigenschaften und Funktionen der bestehenden Lernobjekte adaptiert werden. Ein generelles Konzept zur Wiederverwendung wird in den Standards sowie ihren Erweiterungen jedoch nicht angestrebt.

2.7.2 Motivation eines allgemeinen Ansatzes

Lernobjekte sowie die Art und Weise, auf die sie wiederverwendet werden, sind stark heterogen. Es ist oft nicht ausreichend, Lernobjekte nur unter einem bestimmten Erweiterungsaspekt oder aus einer Anwendungsdomäne heraus zu untersuchen, denn so können nur sehr

spezifische und damit möglicherweise für einige Anwendungsbereiche nur schlecht wiederverwendbare Lösungen abgeleitet werden. Eine Konzentration auf Erweiterungen von Standards, um so noch nicht erfasste Anwendungs- und Inhaltsdomänen abdecken zu können, ist daher nicht zielführend. Vielmehr ist es nötig, sowohl modelltechnisch als auch prozesstechnisch die Wiederverwendung von Lernobjekten zu untersuchen und gezielt zu verbessern. Diese Tatsache motiviert einen allgemeinen Lösungsansatz zur Wiederverwendung von Lernobjekten: Gesucht ist ein integrierter Lösungsweg, um die Probleme, die bei der Lernobjekterstellung und –nutzung auftreten, zu lösen:

- Modelltechnische Integration

  Ziel der modelltechnischen Integration ist es, einen für die Nutzer (Autoren, Dozenten und Lerner) transparenten Informationsaustausch auf der Basis eines gemeinsam genutzten Datenmodells zu ermöglichen. Dies kann mittels eines Metadatenstandards wie LOM, versehen mit geeigneten Erweiterungen zum Zwecke des Informationsaustausches zwischen den Nutzern (nicht zum Zwecke einer Spezialisierung der Lernobjekte auf eine bestimmte Inhalts- oder Nutzungsdomäne!), gewährleistet werden. Eine Verwaltung von Modellen von Lernobjekten – nicht mehr von Lernobjekten selbst – könnte den Entwicklungsaufwand deutlich senken, indem Manipulationen der Lernobjekte in der Modellebene, also auf Gruppen von Lernobjekten, stattfinden und nicht wie bisher auf Lernobjektebene.

- Prozesstechnische Integration

  In 2.4.3 wurden unter anderem Systeme zum Entwurf von Lehrmaterialien vorgestellt. Abgesehen von der technischen Umsetzung der Systeme sind die verwendeten Konzepte sowie das Vorgehen zur Erstellung der Lehrmaterialien recht ähnlich. Es soll daher generell die Vorgehensweise, die die Grundlage einer großen Zahl derzeit implementierter Systeme bildet, abstrahiert und kritisch untersucht werden. Auf der prozesstechnischen Ebene werden daher die Rollen, die Aufgaben, die Aktivitäten und der Datenfluss während der gesamten Prozesskette zur Erstellung und Nutzung von Lehrmaterialien festgehalten und die Ausführung wird kontrolliert durchgeführt. Zentrale Forderung ist es, einen durchgängigen Datenfluss sicherzustellen und Medienbrüche weitgehend zu vermeiden.

Diese Arbeit zeigt einen Weg, wie die modelltechnische Integration und die prozesstechnische Integration auf Basis bestehender Ansätze geleistet werden kann. Daneben wird gezeigt, wie dieser Weg zugleich auch die Qualität und die Wiederverwendbarkeit der Lehrmaterialien erhöht.

2.8 Zusammenfassung

In diesem Kapitel wurden die Grundlagen für diese Arbeit gelegt: Es wurden zentrale Begriffe, insbesondere der Begriff des Lernobjekts, eingeführt und gegeneinander abgegrenzt.

Anschließend wurden zwei Ansätze beschrieben, die die Wiederverwendbarkeit von beliebigen Objekten – nicht notwendigerweise Lehrmaterialien – unterstützen: Der Ansatz, komplexe Objekte mittels den Konzepten der Objektorientierung oder auf der Grundlage von Komponenten- und Framework-Gedanken zu strukturieren, und der Ansatz, die so entstehenden, kleineren und einfacheren Objekte mittels verschiedener Mechanismen (Metadaten, etc.) zu beschreiben. Es wurde erkannt, das beide Ansätze in Kombination genutzt werden müssen, um Wiederverwendbarkeit effizient zu unterstützen.

Im folgenden wurden diese Ansätze auf Lehrmaterialien angewandt. Zuerst wurden allgemeine Anforderungen vorgestellt, denen Lernobjekte folgen müssen, wenn sie sinnvoll wiederverwendbar sein sollen. Hier wurde auch auf die Unterschiede zwischen domänenübergreifender und domänenspezifischer Wiederverwendung, inklusive der daraus resultierenden Anforderungen an die Lernobjekte, eingegangen. Anschließend wurde vorgestellt, wie Metadaten und Transformationsregeln gezielt zur Wiederverwendung von Lernobjekten angewendet werden können. Es wurden Beispielsysteme vorgestellt, in denen diese Ideen umgesetzt wurden. Schließlich wurde in zwei Szenarien beispielhaft gezeigt, wie der Gedanke der Wiederverwendung bei der Erstellung der Lernobjekte (Szenario 1) und bei ihrer Nutzung in der Lehre (Szenario 2) umgesetzt werden kann.

Abschließend wurden die bisher vorgestellten Ansätze, um Wiederverwendbarkeit von Lernobjekten zu unterstützen, einer kritischen Betrachtung unterzogen. Es hat sich gezeigt, dass der oftmals praktizierte Ansatz, Standards so zu erweitern, dass noch nicht abgedeckte Themen oder Anwendungsbereiche nun erfasst werden, nicht zielführend ist. Außerdem wurden Prozessaspekte wie beispielsweise der Informationsfluss zwischen verschiedenen Nutzern der Lernobjekte bislang nicht betrachtet. Diese Defizite machen es nötig, über Wiederverwendung in einem größeren Rahmen nachzudenken, und rechtfertigen daher einen neuen Ansatz. Dieser Ansatz verfolgt das Ziel, die Wiederverwendung von Lernobjekten durch modelltechnische und prozesstechnische Integration gezielt zu verbessern.

3 Ein systemübergreifendes Lehrmaterialmodell

3.1 Überblick

In diesem Kapitel wird ein Datenmodell für Lehrmaterialien vorgestellt, wobei besondere Aufmerksamkeit auf die Wiederverwendbarkeit der einzelnen Lernobjekte gelegt wird. Das Datenmodell, das in diesem Kapitel vorgestellt wird, wurde stark vom Datenmodell von TArgeted Reuse and GEneration of TEAching Materials (Targeteam)[16] [Tee04] beeinflusst; die Eigenschaften der Lernobjekte orientieren sich an LOM (vgl. 2.5.2, [LTSC], [LOM]).

Ziel dieses Kapitels ist es, ein einheitliches Datenmodell für Lehrmaterialien bereitzustellen. Dieses Modell soll es erlauben, dass für einmal erstellte Lernobjekte domänenübergreifende, aber auch domänenspezifische Eigenschaften (vgl. 2.4.2) weitgehend einheitlich verwaltet werden können. Außerdem sollen alle Schritte, die zur Erstellung und Nutzung von Lehrmaterialien nötig sind, auf der Grundlage dieses Modells durchgeführt werden können, so dass auf der Grundlage dieses Modells ein für alle Beteiligten transparenter Daten- und Informationsaustausch erreicht werden kann. Dieses Datenmodell bildet somit die Basis für den Lehrmaterialerstellungs- und -nutzungsprozess, wie er in Kapitel 4 beschrieben wird.

Hierzu umfasst das Modell neben dem Inhalt und der Struktur der Lehrmaterialien auch deren Präsentations- und Navigationsmöglichkeiten sowie die verschiedenen Arten der Nutzung in der Lehre und beim Lernen. Diese Aspekte finden sich auch im Prozess der Lehrmaterialerstellung und –nutzung wieder; bestimmte Prozessschritte greifen auf bestimmte Teile und Daten des Modells zurück. Dies wird in Kapitel 4 näher betrachtet.

Die Erstellung wiederverwendbarer Lehrmaterialien ist ein hoch komplizierter Vorgang. Mit Hilfe einer geeigneten einheitlichen Strukturierung der zu bearbeitenden Daten kann es geschafft werden, die Komplexität dieser Aufgabe unter Kontrolle zu halten. In den folgenden Abschnitten wird ein für alle Autoren und Nutzer gemeinsames Datenmodell für Lehrmaterialien vorgeschlagen. Die aus dem Software-Engineering und der komponentenbasierten Software-Entwicklung bekannten Konzepte der Modularisierung, Parametrisierung und Spezialisierung werden verwendet, um eine weitgehende Wiederverwendung der Lehrmaterialien auf flexible Weise zu unterstützen.

---

[16] Targeteam ist ein gemeinsames Forschungsprojekt des Lehrstuhls für Angewandte Informatik / Kooperative Systeme der TU München und des Instituts für Informationstechnische Systeme, Universität der Bundeswehr München. Targeteam ähnelt in seinem Aufbau einem Autorensystem, das für Lehrmaterialien eine eigene XML-basierte Sprache (TeachML) verwendet.

Zuerst wird in 3.2 die Struktur des Datenmodells vorgestellt: Das Datenmodell wird grob in Abstraktionsebenen und Nutzungsebenen strukturiert. Die Unterteilung in verschiedene Abstraktionsebenen erlaubt eine einfache und flexible Erstellung gut wiederverwendbarer Lernobjekte, während es die Unterteilung in verschiedene Nutzungsebenen ermöglicht, unterschiedliche Arten der Wiederverwendung der Lernobjekte im Modell zu erfassen.

Nach diesem Überblick über die Struktur des Datenmodells werden in 3.3 die einzelnen Bausteine von Lehrmaterialien vorgestellt und modelliert: Lernobjekte (Atome und Module) und wie diese mittels Beziehungen (Kompositionen und Assoziationen) in Zusammenhang gesetzt werden können.

Alle Eigenschaften der Lernobjekte und Beziehungen werden mittels Parametern modelliert, die sich an LOM orientieren, diesen Standard aber an einigen Stellen erweitern. Dies wird in 3.4 vorgestellt. Da auch Parameter gewissen Einschränkungen unterliegen, werden in diesem Abschnitt auch Constraints vorgestellt, ein Mechanismus, mit dem Bedingungen und auf den Bedingungen aufbauende Regeln modelliert werden.

Bislang wurden die verschiedenen Arten der Nutzung von Lehrmaterialien nicht betrachtet. Ob ein Kurs für eine bestimmte Lehrform beispielsweise in HTML oder als Postscript-Datei für eine bestimmte Nutzungsklasse genutzt wird, ist bislang durch das Modell nicht abgedeckt. Die hierfür nötigen Formalismen werden in 3.5 geliefert: Nutzungsklassen, die die verschiedenen Möglichkeiten der Nutzung modellieren, werden eingeführt und ausführlich beschrieben. Damit ist die Beschreibung des Datenmodells abgeschlossen.

In 3.6 wird nun beschrieben, wie in diesem Datenmodell konkrete Lehrmaterialien aus Lernobjekten aufgebaut und instanziiert werden. Es folgt eine Betrachtung, wie das Geflecht der Lernobjekte entsprechend der geplanten Nutzung dargestellt und wie passende Navigationsmöglichkeiten integriert werden.

Abschließend wird das Datenmodell in 3.7 einer kritischen Betrachtung unterzogen. Als wesentlicher Schwachpunkt werden Parameter identifiziert, die mehr oder weniger subjektiv gefärbte, stark nutzungsabhängige oder schwer in einem diskreten Wertebereich darstellbare Werte annehmen.

3.2 Struktur des Datenmodells: Abstraktionsebenen und Nutzungsebenen

Das Datenmodell kann auf zweierlei Arten strukturiert werden: Eine Schichtung in drei verschiedene Abstraktionsebenen (horizontale Schichtung) erlaubt es, gut wiederverwendbare und anpassbare Lehrmaterialien flexibel zu erstellen. Mittels einer anders gestalteten Schichtung in verschiedene Aufgabenbereiche (vertikale Schichtung in Nutzungsebenen wie Struktur, Präsentation, etc.) kann die Komplexität der Erstellung der Lehrmaterialien deutlich reduziert werden, indem man immer nur bestimmte Aspekte – Inhalt und Struktur, beabsichtigte Nutzung in der Lehre, etc. – betrachtet. Dieser Gedanke wird mit den „Sichten“ in 4.2.3.2 nochmals aufgegriffen.

3.2.1 Horizontale Schichtung: Abstraktionsebenen

Das Datenmodell ist in drei Schichten unterteilt (vgl. Abbildung 3-1). Der Abstraktionsgrad der Objekte in der jeweiligen Schicht nimmt von unten nach oben zu. Jede Ebene – mit Ausnahme der untersten – definiert dabei die Konzepte, die zur Beschreibung der Objekte der nächst niedrigeren Ebene benötigt werden (Pfeil „beschreibt“). Die auf einer Ebene modellierten Objekte sind damit stets Instanzen der auf der nächst höheren Ebene gelegenen Konzepte, welche umgekehrt den Objekten auf der niedrigeren Ebene als Rahmen dienen (Pfeil

„instanziiert“). Diese Abstraktion der Modellierung ist eine Voraussetzung für die Strukturierung der Lernobjekte sowie Ausgangspunkt für die Erstellung wiederverwendbarer Lernobjekte (vgl. Kapitel 2.3.1.1).

3.2.1.1 Instanzebene

Auf der untersten Ebene, der *Instanzebene*, sind die konkreten Lehrmaterialien, die *Instanzkomponenten*, zu finden. Diese sind in Inhalt und gegebenenfalls auch gemäß ihrer Nutzung und Nutzer(gruppe) abgeschlossen. Instanzkomponenten können aus Texten, verschiedenen Multimedia-Objekten oder rekursiv wiederum Instanzkomponenten bestehen.

Beispiel:

> Ein Kapitel zum Kurs „CSCW“ (Instanzkomponente) besteht aus den drei Unterkapiteln „Kommunikation“, „Koordination“ und „Kooperation“ (jeweils Instanzkomponenten), die jeweils wieder Texte (Instanzkomponenten) und Bilder (Instanzkomponenten) beinhalten.
>
> Alle Unterkapitel enthalten keine Platzhalter, sondern konkreten Inhalt und sind daher Instanzkomponenten.

Den Instanzkomponenten stehen die Rahmenkomponenten der Rahmenebene gegenüber: Im Gegensatz zu Instanzkomponenten haben Rahmenkomponenten keinen Inhalt, und damit existieren auch keine konkreten Ausprägungen.

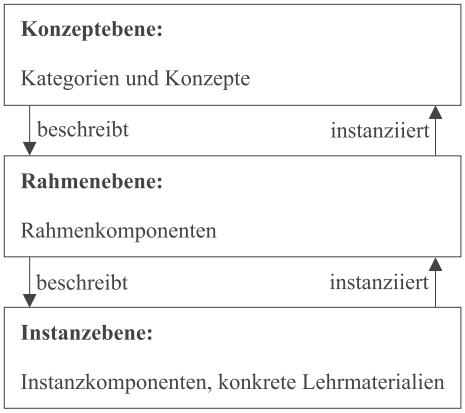


Abbildung 3-1: Strukturierung des Datenmodells in Abstraktionsebenen

3.2.1.2 Rahmenebene

In der mittleren Ebene, der *Rahmenebene*, wird beschrieben, aus welchen Elementen ein komplexeres Lernobjekt (eine *Rahmenkomponente*) aufgebaut ist und welche Eigenschaften diese Elemente haben müssen; in dieser Ebene wird die Struktur eines Lernobjekts definiert. Wesentlich an der Rahmenebene ist, dass in dieser Ebene Lernobjekte lediglich beschrieben werden; die Elemente, aus denen diese Lernobjekte bestehen, sind nicht selbst („inline“) in den Lernobjekten enthalten, sondern sind ausgelagert und existieren als eigenständig nutzbare und wiederverwendbare Objekte. An den Stellen der Rahmenkomponenten, an denen Elemente stehen sollen, sind Platzhalter eingefügt, die beschreiben, welche Elemente an dieser einen Stelle eingefügt werden dürfen[17]. In der Rahmenebene werden somit keine konkreten Lernobjekte definiert, sondern Schablonen, die keine konkreten Lehrinhalte enthalten.

[17] Auf welche Weise die Elemente beschrieben werden und wie die Beziehungen zwischen einem Lernobjekt und seinen Elementen realisiert und aufgelöst werden, wird in 3.3 genauer beschrieben.

In Rahmenkomponenten werden konkrete Lehrmaterialien der Instanzebene oder wiederum Rahmenkomponenten eingefügt. Konkrete Lehrmaterialien – die Instanzkomponenten der Instanzebene – entstehen durch Auswahl und Auswertung der vordefinierten Rahmenkomponenten. (Details hierzu in 3.6.) Dies ermöglicht beispielsweise einen unproblematischen Austausch eines Lernobjekts, das in eine Rahmenkomponente eingebettet ist, gegen ein anderes: Das fragliche Lernobjekt wird einfach gegen ein anderes ausgetauscht, das ebenfalls in den jeweiligen Platzhalter passt.

Der Übergang von Rahmen- zu Instanzkomponenten geschieht bei der Zuordnung von konkreten Instanzkomponenten zu Platzhaltern der Rahmenkomponenten: Eine Rahmenkomponente wird zur Instanzkomponente, wenn rekursiv alle Subelemente der Rahmenkomponente Instanzkomponenten sind.

3.2.1.3 Konzeptebene

Motivation der Einführung der Konzeptebene ist eine Zuordnung von allgemeinen Strukturmustern zu Rahmenkomponenten sowie eine flexible Klassifizierung der Rahmenkomponenten. In der *Konzeptebene*, der obersten Ebene, werden Kategorien (wie Strukturmuster) definiert, welche eine Zusammenfassung von Lernobjekttypen der Rahmenkomponenten anhand ihrer Gemeinsamkeiten erlauben: Rahmenkomponenten mit ähnlichen Eigenschaften werden in der gleichen Kategorie zusammengefasst.

Die Strukturmuster, nach denen die Rahmenkomponenten kategorisiert werden, können unterschiedliche Gestalt haben. Sie können das Thema von Rahmenkomponenten umfassen, aber auch deren Aufbau (die Struktur) oder deren Nutzung. Die einzelnen Strukturmuster sind jeweils unabhängig voneinander. Dies erlaubt es, dass Rahmenkomponenten mehreren Kategorien unterschiedlicher Strukturmuster zuordenbar sind, so dass sie genauer beschrieben werden können. So ist es beispielsweise möglich, eine gegebene Rahmenkomponente anhand ihres Inhalts „Grove-Algorithmus“ der Kategorie „CSCW“ (Strukturmuster „Thema“) und zugleich anhand ihres Aufbaus in „Einleitung“ – „Sequenz von Unterkapiteln“ – „Zusammenfassung“ der Kategorie „Linearer Aufbau“ (Strukturmuster „Aufbau“) zuzuordnen. Die Rahmenkomponente „Grove-Algorithmus“ gehört dann beiden Kategorien an.

Folgende Strukturmuster[18] werden in dieser Arbeit betrachtet:

- Inhaltskategorien: Themengebiete, beispielsweise „Technische Informatik“, aber auch feingranularere Themen wie „Rechnernetze“, „Verteilte Anwendungen“, „Betriebssysteme“
- Strukturkategorien: Inhaltliche Strukturierungseinheiten, wie beispielsweise „Definition“, „Beispiel“ oder „Zusammenfassung“
- Didaktische Kategorien: Didaktische Ausprägungen wie beispielsweise „Frage“, „Übung“ oder „Literatur“

Weitere Kategorien sind denkbar.

Diese Kategorisierung nach Strukturmustern erleichtert auf Grund der Vorstrukturierung die Auswahl und Einordnung ähnlicher Rahmenkomponenten und vereinfacht so die Wiederverwendung einmal definierter, klassifizierter Rahmenkomponenten.

[18] Kategorien und Strukturmuster sind in dieser Arbeit eine Art der Strukturierung von Lernobjekten und werden daher nicht explizit formal modelliert und vertieft. Interesierte Leser seien an dieser Stelle auf die Literatur zu Ontologien verwiesen, wie beispielsweise die Konzepthierarchien in Teachware on Demand [CaHo01] oder [Kre02].

3.2.2 Vertikale Schichtung: Nutzungsebenen

Neben einer Strukturierung der Lehrmaterialien in verschiedene Abstraktionsebenen ist eine Strukturierung entsprechend der intendierten Nutzung sinnvoll. Jedes Lernobjekt besteht aus einer Menge von Ebenen, die jeweils bestimmte Sichten auf die Wiederverwendung widerspiegeln. Diese Ebenen werden *Nutzungsebenen* genannt. Indem von Nutzungsebene zu Nutzungsebene mehr Aspekte der Nutzung berücksichtigt werden, kann die Erstellung flexibel anwendbarer Lehrmaterialien spürbar vereinfacht werden. Gemäß Abbildung 3-2 wird eine Abstraktion in die nutzungsfreien Ebenen („Inhalt“ und „Struktur“), Nutzungs-, Präsentations- und Navigationsebene vorgenommen.

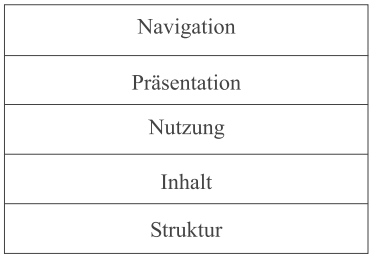


Abbildung 3-2: Strukturierung des Datenmodells in Nutzungsebenen

Diese Sicht der Schichtung wird auf jedes Lernobjekt – Rahmen- als auch Instanzkomponenten – angewandt. Jede Nutzungsebene umfasst bestimmte, auf die vorgesehene Nutzung des Lernobjekts ausgerichtete Aspekte des Lernobjekts; je nach Art der Nutzung sind bestimmte Nutzungsebenen des Lernobjekts besonders von Interesse.

Ausgangspunkt sind die *nutzungsfreien Ebenen* „Inhalt“ und „Struktur“: Die Lernobjekte beinhalten keinerlei Informationen über die spätere Nutzung. Sie enthalten lediglich inhaltliche und strukturelle Informationen.

Auf diesen Ebenen aufbauend entsteht eine *Nutzungs-* bzw. *Nutzerebene*, die bereits Aspekte der Lehre, der verschiedenen Lernformen und der Nutzer berücksichtigt. In dieser Ebene erfolgt beispielsweise die Anpassung der Lernobjekte an die Zielgruppe, was bei der Auswahl der für die Nutzung relevanten Lernobjekte eine Rolle spielt. Beispielsweise muss der Schwierigkeitsgrad der ausgewählten Lernobjekte dem Leistungsvermögen der Zielgruppe entsprechen. Es ist möglich, auf bestehende nutzungsfreie Ebenen mehrere, jeweils verschiedene Nutzungs- und Nutzerebenen zu legen. Dies entspräche beispielsweise der Nutzung eines Lernobjekts (gleiche Inhalts- und Nutzungsebene) für unterschiedliche Zielgruppen (je Zielgruppe eine unterschiedliche Nutzungs- und Nutzerebene).

Darauf aufbauend werden in der nächsten Ebene, der *Präsentationsebene*, die verschiedenen Präsentationsformate der Lernobjekte spezifiziert. Auch hier ist es wieder möglich, auf eine gegebene Nutzungs- und Nutzerebene mehrere unterschiedliche Präsentationsebenen zu legen. Dies entspräche dem Fall, dass ein Lernobjekt für eine bestimmte Zielgruppe konfiguriert ist und nun für mehrere verschiedene Lehrformen (beispielsweise Präsenzvorlesung und Selbststudium) genutzt werden soll.

In der obersten Ebene, der *Navigationsebene*, werden schließlich je nach Präsentationsform die erlaubten Darstellungen und Navigationsstile innerhalb bzw. zwischen den Lernobjekten spezifiziert.

3.3 Modularisierung von wiederverwendbaren Lehrmaterialien

3.3.1 Module und Atome

Alle Lehrmaterialien im Datenmodell dieser Arbeit bestehen aus zwei Arten von Objekten: Modulen und Atomen, wobei Module aus Atomen und wiederum aus Modulen bestehen können.

Atome sind nicht weiter sinnvoll unterteilbar, ohne dass ihre Eigenständigkeit und Wiederverwendbarkeit verloren ginge. Beispiele für Atome sind Texte, Animationen, Bilder, etc. Atome sind daher immer Instanzkomponenten. Atome sind geeignet, bereits bestehende Lehrmaterialien im Binärformat, beispielsweise Microsoft Powerpoint-Präsentationen, Flash-Animationen, etc., zu kapseln.

Module dagegen sind komplexe Objekte. Sie bestehen aus Atomen und / oder wiederum aus Modulen (strukturelle Rekursion). Sie sind in sich thematisch abgeschlossen und eigenständig wiederverwendbar. Ein Modul kann auch einen konkreten Kurs umfassen, welcher ein bestimmtes Themen- oder Fachgebiet behandelt. Dieses Modul wäre darüber hinaus auch in seiner Nutzung abgeschlossen, zum Beispiel für den Einsatz in einer bestimmten, in ihren Eigenschaften spezifizierte Lehrveranstaltung. Module können Instanz- oder Rahmenkomponenten sein, je nachdem, ob alle Elemente (Submodule oder Atome), aus denen das Modul besteht, selbst Instanzkomponenten sind.

Soll nicht zwischen Atomen und Modulen unterschieden werden, wird in dieser Arbeit der Begriff des *Lernobjekts* verwendet. Der Begriff der *Lehrmaterialien* bezeichnet in dieser Arbeit allgemein konkrete, elektronisch vorliegende Lehrmittel, die die Lerner zum Wissenserwerb nutzen.

3.3.2 Beziehungen

Komplexe Lernobjekte werden aus einfacheren Lernobjekten mittels Beziehungen aufgebaut. In dieser Arbeit werden zwei verschiedene Arten von Beziehungen unterschieden: *Kompositionen*, die die Struktur der Lernobjekte aufbauen, und *Assoziationen*, welche beliebige, nicht die Struktur betreffende Beziehungen zwischen Lernobjekten modellieren.

Der hierarchische Aufbau von Elementen aus einfacheren und/oder komplexen Lernobjekten wird mittels Kompositionen zwischen den Atomen und Modulen realisiert. Referenzen zwischen Lernobjekten erfolgen mittels Assoziationen, beispielsweise Vorgänger-Nachfolger-Relationen oder Äquivalenzen.

Beziehungen können auf zwei Weisen realisiert werden: „Inline“ und mittels Referenzen (vgl. Abbildung 3-3). Bei der Realisierung als Inline-Elemente werden die Subelemente direkt in das Lernobjekt eingebunden (vgl. Beispiel unten). Die Subelemente werden so unmittelbar Teil der Beschreibung des komplexen Lernobjekts. Bei der Realisierung mittels Referenzen bleiben die Subelemente eigenständig und werden nur über ihren Identifikator angesprochen.

Beispiel:

> Anmerkung: Die folgenden Code-Fragmente nutzen eine fiktive, XML-basierte Grammatik.
>
> Einbindung von Subelementen „inline“:

```
<component>
   Dies ist ein Lernobjekt.
```

```
      <component>
         Dies ist Subelement 1.
      </component>
      <component>
         Dies ist Subelement 2.
       </component>
   Jetzt ist das Lernobjekt zu Ende.
</component>
```

Einbindung von Subelementen mittels Referenzen:

```
<component>
   Dies ist ein weiteres Lernobjekt.
      <ref target="Subelement1"/>
      <ref target="Subelement2"/>
   Jetzt ist auch dieses Lernobjekt zu Ende.
</component>
```

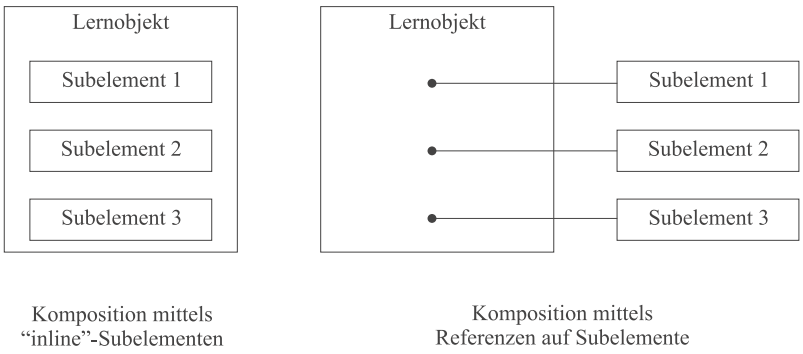


Abbildung 3-3: Realisierung von Beziehungen „inline" und mittels Referenz

Die Realisierung von Beziehungen mittels Referenzen hat den Vorteil, dass die Subelemente des Lernobjekte eigenständig bleiben und so auch unabhängig von evtl. bereits bestehenden Beziehungen wiederverwendet werden können. Dies ist nicht möglich, wenn die Lernobjekte bereits ihre Subelemente enthalten. Daher werden in dieser Arbeit Beziehungen über Referenzen realisiert.

Komposition

Von *Komposition* wird gesprochen, wenn ein Lernobjekt aus mehreren Subelementen aufgebaut ist. Das übergeordnete Lernobjekt steht dann implizit (über die im Platzhalter definierten Anforderungen) mit seinen Subelementen in einer Kompositionsbeziehung. Komposition bezeichnet damit eine hierarchische strukturelle Beziehung zwischen einander über- bzw. untergeordneten Lernobjekten und wird bei der Erstellung von Lernobjekten aus bereits bestehenden Lernobjekten angewendet.

Im folgenden werden typische Kompositionstypen genannt (vgl. auch [CaHo01]). Diese sind als Kategorien in der Konzeptebene definiert:

- Strukturelle Komposition:

  Die Komposition der Lernobjekte spiegelt die existierende Struktur der Lernobjekte wider.

- Inhaltliche Komposition:

  Semantisch zusammenhängende Lernobjekte (beispielsweise Motivation und Definition) werden zu einem größeren Lernobjekt (als Grundlage) zusammengefasst. Die

hierarchische Struktur der Lernobjekte untereinander wird unmittelbar durch die inhaltliche Komposition abgebildet.

- Ordinale Komposition:

  Lernobjekte, die durch eine Anordnung auf einer Skala (beispielsweise einer Zeitleiste, einem didaktischen Vorgang wie dem Ablauf einer Lehrveranstaltung) in Zusammenhang stehen, werden zu einer größeren Komponente zusammengefasst, beispielsweise die Komposition von Lernobjekten entsprechend einer didaktischen Strukturierung aus Konzepten, Fragen, Übungen und Literaturverweisen.

Assoziation

Eine *Assoziation* bezeichnet allgemein eine Beziehung zwischen zwei Lernobjekten, die nicht notwendigerweise in einer hierarchischen Beziehung stehen müssen. Mit Assoziationen werden zum Beispiel semantische Beziehungen zwischen Lernobjekten ausgedrückt. Beispiele für Assoziationen sind „Vorwissen zu einem Modul“, „Beispiel zu einer Definition“ oder „Aufgabe zu einem Kapitel“. Im Gegensatz zu Kompositionen werden Assoziationen nicht bereits bei der Erstellung der Lehrmaterialien über die Schachtelung der Elementen definiert, sondern müssen explizit (beispielsweise anhand von Metadaten oder Annotationen, vgl. 3.4.2.1 und 5.4.1.2) auf der Rahmen- oder Instanzebene definiert werden. Dies kann bei der Erstellung der Lehrmaterialien durch den Autor geschehen.

3.3.3 Parameter

Die Beschreibung der Eigenschaften von Lernobjekten und Beziehungen wird mit *Parametern* realisiert[19]. Parameter sind Metadaten. Sie sind typisiert und werden Wertebereichen (Datentypen) zugewiesen. Die Auswertung der Parameter und die Zuordnung von Parameterbelegungen zu konkreten Instanzkomponenten, die in die Platzhalter der Rahmenkomponenten eingesetzt werden dürfen, erfolgt erst auf der Instanzebene.

Neben einem eindeutigen Identifikator, dem Parameternamen, verfügen Parameter stets über genau einen Datentyp, der die Grundmenge für den Parameterwert vorgibt, wie beispielsweise Zahlen oder Zeichenketten. Der Wertebereich von Parametern orientiert sich an den Datentypen, die in LOM definiert sind (nach [LOM]):

- LangString: Zeichenkette
- DateTime und Duration[20]
- Vocabulary: Sammlung von gleichen, gleichwertigen Typen, wie Mengen von Daten. Diesen Datentyp findet man zum Beispiel bei der Beschreibung des Inhaltstyps eines Lernobjekts: Problembeschreibung, Definition, Zustand, etc.
- Aufzählungen: Der Wertebereich von Aufzählungen wird durch explizite Aufzählung der Elemente einer endlichen Menge gleichartiger Werte festgelegt. Die beschriebenen Kategorien sind konform zu Aufzählungen, also einer Menge erlaubter Werte in Form von Vokabeln oder einer Referenz zu einem anderen Standard. Ein Beispiel ist hier die Bestimmung des Detaillierungsgrades eines Lernobjekts: niedriger, mittlerer, hoher, sehr hoher Detaillierungsgrad.

Insgesamt stellen die beschriebenen LOM-Datentypen das Fundament dar, auf dem das Datenmodell aufbaut. Sie geben das Werteuniversum vor, das benutzt werden kann, um Eigen-

[19] Ein den Parametern vergleichbares Konzept sind Attribute in der objektorientierten Datenmodellierung.
[20] Für das Datum und die Dauer wird der ISO 8601:2000-Standard verwendet.

schaften von Elementen darzustellen. Das LOM-Basisschema (vgl. 2.5.2 und [LOM]) kann durch Hinzufügen von Kategorien, Parametern und Datentypen beliebig erweitert werden: Aus dem Basisschema kann damit ein neues Schema abgeleitet werden.

3.3.4 Nutzung der grundlegenden Bausteine: Aufbau von Lehrmaterialien und Kursen

Lehrmaterialien werden aus Lernobjekten – Modulen und Atomen – aufgebaut. Die Struktur der Lehrmaterialien spiegelt sich dabei unmittelbar in ihrem Aufbau aus den Lernobjekten wider. Lehrmaterialien haben damit eine Baumstruktur: Die inneren Knoten des Baums sowie dessen Wurzel sind Module, die Blätter sind Atome. Damit folgt die Struktur der Lehrmaterialien dem Composite-Pattern (vgl. [GHJV96] und Abbildung 3-4).

Die Tiefe des Lernobjektbaums variiert von Wurzel zu Wurzel. Ist beispielsweise die Wurzel ein kompletter Kurs, ist der Baum tiefer als wenn die Wurzel ein einfaches Modul ist, das lediglich ein Atom kapselt.

Kurse sind besondere Lernobjekte: Strukturell sind sie zwar identisch mit einem gewöhnlichen Modul, das ein oder mehrere Submodule bzw. Atome kapselt. Sie sind allerdings nicht nur inhaltlich abgeschlossen, sondern auch in ihrer Nutzung in der Lehre. Kurse können begleitend zu Lehrveranstaltungen eingesetzt werden, das heißt, Kurse haben eine bestimmte Nutzung sowie evtl. ein bestimmtes Präsentationsformat. Dies ist bei beliebigen komplexen Lernobjekten im allgemeinen nicht der Fall. Zu Aspekten bzgl. der Nutzung sei auf 3.5 verwiesen.

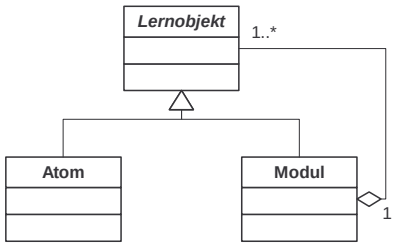


Abbildung 3-4: Composite-Pattern zwischen Atomen und Modulen

3.3.5 Modellierung der grundlegenden Bausteine

Im folgenden wird ein formales Datenmodell für die grundlegenden Bausteine hergeleitet. Aus Gründen der Übersicht wird immer nur der für den jeweiligen Abschnitt relevante Teil des Datenmodells gezeigt; ein vollständiger Überblick in UML (Unified Modeling Language) ist in Anhang B zu finden.

Parameter

Parameter bilden die Grundlage zur Beschreibung von Atomen, Modulen und Beziehungen. Sie legen auch fest, aus welchen Subelementen ein Modul aufgebaut ist.

Parameter werden folgendermaßen definiert:

```
<!ELEMENT parameter (id, domain, value, default, option)>,
```

wobei

```
<!ELEMENT id #CDATA>
<!ELEMENT domain #CDATA>
<!ELEMENT value #CDATA>
<!ELEMENT default #CDATA>
```

```
<!ELEMENT option („MANDATORY“|„OPTIONAL“|„CONDITIONAL“)>
```

Id ist der Identifikator, mit dem der Parameter eindeutig identifiziert werden kann. Domain ist die Wertemenge, aus der der konkrete Wert value des Parameters stammen darf. Default ist ein Default-Wert und option gibt an, ob der Parameter verpflichtend beachtet werden muss.

Beispiel:

```
<parameter>
   <id>MeinParameter</id>
   <domain>Integer</domain>
   <value>0815</value>
   <default>0815</default>
   <option>optional</option>
</parameter>
```

Einschränkung von Parameterwerten mittels Constraints

In vielen Fällen ist es sinnvoll, Vorgaben für die Werte von Parametern zu machen und nicht den gesamten Wertebereich zuzulassen. Dies wird mittels *Constraints* realisiert. Constraints definieren Nebenbedingungen oder Vorgaben, denen Werte für Parameter folgen müssen: Die Lernobjekte werden präziser beschrieben, was Missverständnisse bezüglich der Verwendung der Lernobjekte vermeiden hilft und so deren Wiederverwendung erleichtert. Aufgrund ihrer Komplexität werden Constraints meist von Autoren von Lernobjekten und weniger von Lernern, die eigene Lehrmaterialien aus bestehenden aufbauen, eingesetzt.

Constraints können für jede Art von Parameter verwendet werden – für Parameter von Atomen und Modulen ebenso wie für Parameter von Beziehungen. Abbildung 3-5 zeigt den Zusammenhang zwischen Constraints und Parametern.

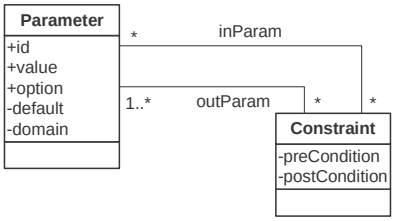


Abbildung 3-5: Zusammenhang zwischen Constraints und Parametern

Constraints können eine Vielzahl funktionaler Ausdrücke enthalten:

- arithmetische Grundoperationen auf Parametern bzw. ihren Werten (+, -, *, /, modulo)
- prädikatenlogische Ausdrücke auf Parametern bzw. ihren Werten:

  Vergleichsoperationen auf numerischen Werten (=, ≠, <, >, ≤, ≥),
- logische Grundoperationen (and, or, not)
- bedingte Ausdrücke auf Parametern bzw. ihren Werten:

  if (Bedingung, $\text{Ausdruck}_{\text{true}}$, $\text{Ausdruck}_{\text{false}}$)

Dabei kann auch auf andere Parameter oder deren Werte bezug genommen werden, wodurch man Abhängigkeiten von Parametern oder Parameterwerten formulieren kann.

Beispiel:

> Um zu fordern, dass der Abstraktionsgrad eines Lernobjekts immer niedrig sein muss, wenn ihr Schwierigkeitsgrad niedrig ist, müsste man ein Constraint auf dem Parameter „Abstraktionsgrad“ des Lernobjekts wie folgt definieren:
>
> value(„Schwierigkeitsgrad“) = „niedrig“ ⇒ value(„Abstraktionsgrad“) = „niedrig“

Formal werden Constraints wie folgt definiert:

```
<!ELEMENT constraint #CDATA>
```

Diese bewusst unstrukturiert gehaltene Definition von Constraints trägt der Vielzahl der Bedeutungen Rechnung, die Constraints ausdrücken können. Diese Definition erlaubt es, verschiedene Sprachen zur Beschreibung von Abhängigkeiten, Einschränkungen, etc. einzusetzen. Eine denkbare Sprache wäre die Object Constraint Language (OCL, vgl. [OMG]).

Atome und Module

Atome und Module sind die Bausteine, aus denen alle Lehrmaterialien aufgebaut sind. Atome enthalten die konkreten Inhalte, Module integrieren Atome oder wiederum Module, vgl. 3.3.1.

Jedes Lernobjekt – egal ob Atom oder Modul – wird eindeutig über einen Identifikator bestimmt. Die Beschreibung der Eigenschaften eines Lernobjekts erfolgt mittels Parametern.

Der Inhalt von Atomen wird mittels Uniform Resource Locators (URLs) referenziert[21]. Dies ermöglicht es, flexibel auf beliebige Daten zuzugreifen.

Die Struktur von Modulen wird mittels Kompositionen realisiert. Da Module auf verschiedenen Hierarchieebenen (auf verschiedenen Ebenen im Kursbaum) wiederverwendet werden können, ist es nicht sinnvoll, die Kompositionsbeziehungen zwischen Lernobjekten (Atomen wie Modulen) im Modul selbst zu definieren. Statt dessen sind in den Modulen an den Stellen, an denen Subelemente eingesetzt werden sollen, Platzhalter[22] eingesetzt, die eindeutig innerhalb des Moduls identifiziert werden können und die die geforderten Eigenschaften der Subelemente beschreiben. In diesen Platzhaltern kann auch die Semantik der Beziehung zwischen dem Modul und dem jeweiligen Subelement beschrieben werden, beispielsweise Subelement ist „Übung“, „Erklärung“, „Beispiel“, etc. Die konkrete Kompositionsbeziehung zwischen einem Platzhalter in einem Modul und einem in den Platzhalter einzusetzenden Subelement wird mittels einer Beziehung, die nicht im Modul selbst definiert ist, realisiert. Hierzu mehr im nächsten Abschnitt.

Ein Atom wird damit folgendermaßen definiert:

```
<!ELEMENT atom (ID, contentURL, parameter*, constraint*)>
```

wobei

```
<!ELEMENT ID #CDATA>
<!ELEMENT contentURL #CDATA>
```

und parameter sowie constraint wie im vorherigen Abschnitt.

---

[21] Hier wird deutlich, dass jedes Atom immer eine Instanzkomponente ist, denn in jedem Atom ist ein Verweis auf konkreten Inhalt vorhanden.

[22] Module sind damit in erster Linie Rahmenkomponenten; sie enthalten keinen konkreten Inhalt, sondern nur Beschreibungen, wie die Inhalte gestaltet sein müssen, die in einen gegebenen Platzhalter eingesetzt werden dürfen. Ein Modul wird erst dann zu einer Instanzkomponente, wenn rekursiv alle Subelemente konkrete Inhalte enthalten, also selbst Instanzkomponenten sind.

Module sind etwas komplizierter aufgebaut:

```
<!ELEMENT module (ID, placeholder+, parameter*, constraint*)>
```

wobei

```
<!ELEMENT ID #CDATA>
<!ELEMENT placeholder (placeholderID, parameter+, constraint*)>
<!ELEMENT placeholderID #CDATA>
```

und parameter sowie constraint wie im vorherigen Abschnitt. Abbildung 3-6 verdeutlicht die Zusammenhänge.

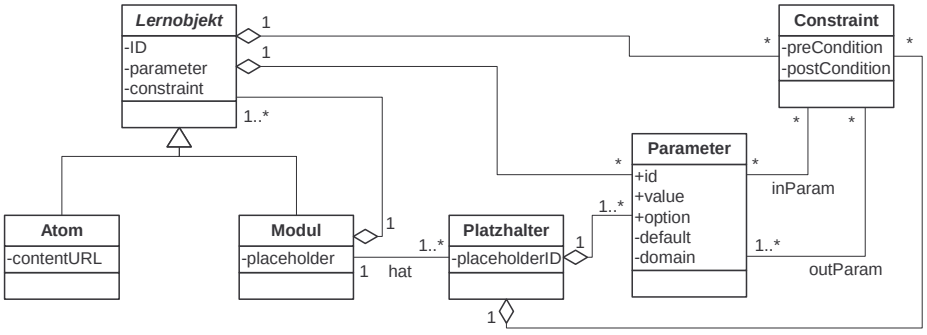


Abbildung 3-6: Zusammenhang Atome, Module, Constraints und Parameter

Die Module und Atome sind konform mit den Learning Objects in LOM [LOM]:

- Atome entsprechen folgendem LOM-Datenelement:

  LOM.General.Aggregationlevel = 1

- Module analog:

  LOM.General.Aggregationlevel $\in \{2, ... 4\}$[23]

In Unterschied zu dem hier vorgestellten Datenmodell stellt LOM keine Informationen über die Zusammensetzung der Lernobjekte bereit. Statt dessen wird in LOM die Datenstruktur implizit über die eingeführte Modellabstraktion beschrieben. Aus diesem Grund wird der Eintrag LOM.General.Aggregationlevel, über den dies in LOM realisiert wird, in dieser Arbeit nicht berücksichtigt.

Beziehungen

In den Modulen selbst sind keine expliziten Beziehungen zwischen einem Platzhalter und einem einzusetzenden Subelement realisiert, sondern diese Beziehungen werden implizit über die Parameter des Platzhalters bzw. der einzusetzenden Subelemente beschrieben. Assoziationen werden ebenfalls nicht in den betreffenden Lernobjekten selbst abgelegt. Alle Beziehungen haben wieder Eigenschaften, beispielsweise Angaben darüber, wie die Beziehung bei bestimmten Präsentationsformaten (HTML, PDF, etc.) des Lernobjekts abgebildet werden sollen. Diese Eigenschaften werden wieder mittels Parametern realisiert.

Dadurch dass die Beziehungen zwischen einem Modul und seinen Subelementen getrennt voneinander definiert sind, können in einem Modul auf einfache Weise Beziehungen geän-

[23] LOM sieht einen Maximalwert für LOM.General.Aggregationlevel von 4 vor.

dert werden. Es ist dann nicht nötig, Änderungen am Modul oder am Subelement durchzuführen, um Beziehungen zu ändern.

Assoziationen sind Beziehungen zwischen zwei Lernobjekten – egal ob Modul oder Atom. Assoziationen haben folgende Gestalt:

```
<!ELEMENT assoziation (sourceID, targetID, parameter*,
  constraint*)>
```

wobei

```
<!ELEMENT sourceID #CDATA>
<!ELEMENT targetID #CDATA>
```

und parameter sowie constraint wie weiter oben.

SourceID ist der Identifikator des Lernobjekts, von dem die Assoziation ausgeht, und targetID ist der Identifikator des Moduls oder Atoms, auf das die Assoziation hinweist. Die Angabe von sourceID und targetID ist nötig, da Module und Atome auf der einen Seite und Beziehungen auf der anderen Seite getrennt verwaltet werden und in einem Modul oder Atom kein Bezug auf die mit dem Modul oder Atom verbundenen Beziehungen besteht.

Assoziationen sind je nach ihrer Semantik teilweise gerichtet (beispielsweise im Fall der Semantik „ist Detaillierung zu“), teilweise ungerichtet (beispielsweise im Fall der Semantik „ist äquivalent zu“).

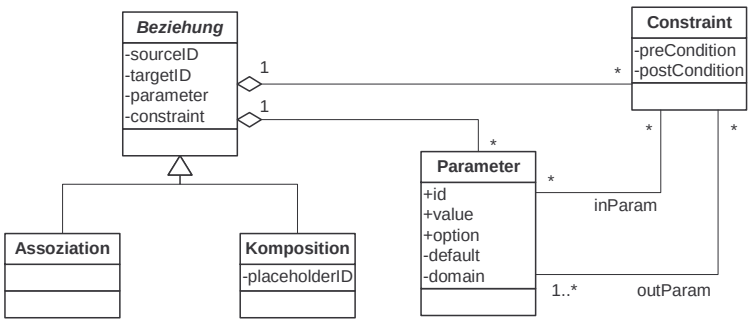


Abbildung 3-7: Assoziationen und Kompositionen

Kompositionen realisieren Beziehungen zwischen einem Modul und einem Subelement, das in einen konkreten Platzhalter eingesetzt werden soll. Kompositionen haben daher im Vergleich zu Assoziationen eine leicht andere Gestalt:

```
<!ELEMENT composition (sourceID, placeholderID, targetID,
  parameter*, constraint*)>
```

wobei

```
<!ELEMENT sourceID #CDATA>
<!ELEMENT targetID #CDATA>
<!ELEMENT placeholderID #CDATA>
```

und parameter sowie constraint wie weiter oben.

SourceID ist der Identifikator des Lernobjekts, von dem die Komposition ausgeht, placeholderID ist der Identifikator des Platzhalters in Lernobjekt sourceID, in den das Subelement eingesetzt werden soll, und targetID ist der Identifikator des Subelements (Modul oder Atom), das in den Platzhalter eingesetzt werden soll. Abbildung 3-7 liefert die Modellierung von Assoziationen und Kompositionen in UML-Notation.

Im Gegensatz zu Assoziationen, deren Semantik darüber entscheidet, ob sie gerichtet sind oder nicht, sind Kompositionen immer gerichtet; dies liegt an der „part-of“-Beziehung zwischen dem Modul und dem entsprechenden Subelement.

Beziehungen zwischen Lernobjekten werden damit unmittelbar durch Assoziationen und Kompositionen ausgedrückt. Die Beziehung zwischen Modulen und Lernobjekten, wie sie in Abbildung 3-6 eingezeichnet ist, besteht somit nur indirekt. Abbildung 3-8 zeigt, wie Lernobjekte nun über Assoziationen und Kompositionen verbunden sind.

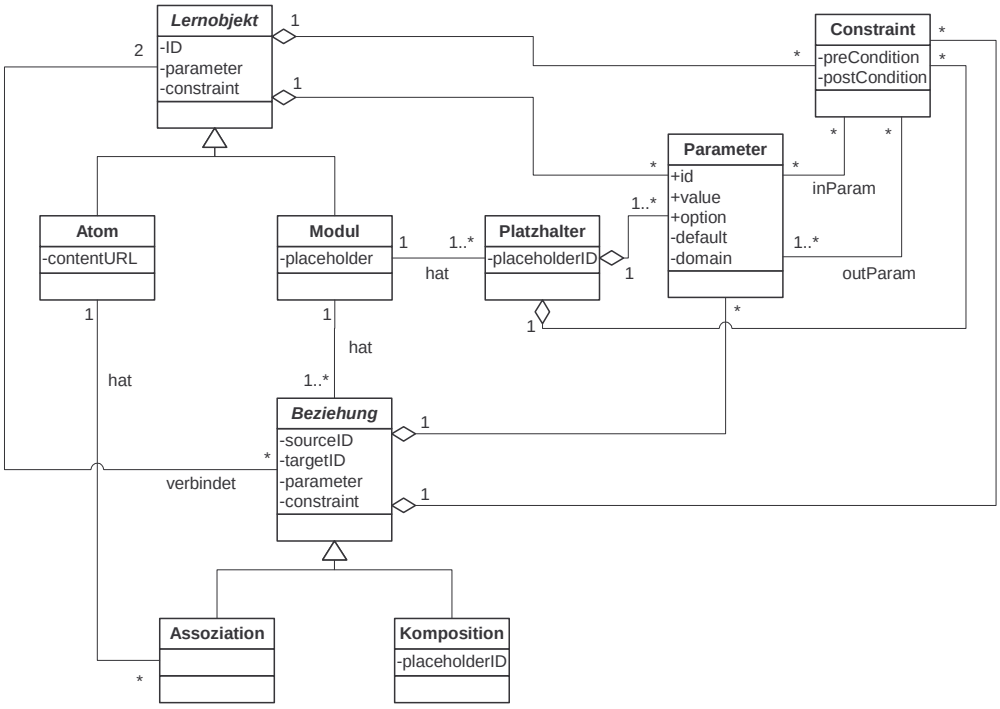


Abbildung 3-8: Beziehungen zwischen Lernobjekten mittels Assoziationen und Kompositionen

Neben dem Erstellen von gerichteten, zusammenhängenden und zyklenfreien Graphen (hierarchische, baumartige Struktur) mittels Kompositionen sind in dieser Arbeit auch Querverweise zwischen den Lernobjekten (Assoziationen) erlaubt. Allerdings können diese Beziehungen nicht mit dem LOM-Standard ausgedrückt werden. Dennoch können Kompositionen und Assoziationen, als Teil der „Relation“-Kategorie des LOM-Basisschemas [LOM], [Dub b] kodiert bzw. exportiert werden, welche das Formulieren von gerichteten Relationen ausgehend vom LOM-Lernobjekt ermöglicht. Dies ist in Tabelle 3-1 illustriert.

LOM	Abbildung auf Beziehungen in dieser Arbeit
LOM.Relation.kind=“HasPart“	Komposition
LOM.Relation.kind=“IsReferencedBy“ LOM.Relation.kind=“References“	Assoziation

Tabelle 3-1: Zuordnung "Relationen in LOM – Beziehungstypen"

3.4 Parametrisierung der Lernobjekte und Beziehungen

In 3.3.5 wurde beschrieben, wie komplexe Module (in der Rahmenebene) aus einfacheren Lernobjekten (Rahmen- oder Instanzkomponenten) aufgebaut werden können. Platzhalter, die mit Parametern versehen waren, haben hierbei eine zentrale Rolle gespielt: An die Stelle der Platzhalter werden Lernobjekte eingesetzt, die bestimmten, in den Parametern der Platzhalter formulierten Eigenschaften entsprechen. Der Beschreibung von Lernobjekten mittels Parametern kommt damit zentrale Bedeutung zu.

In diesem Abschnitt werden die Parameter näher betrachtet. Es wird insbesondere gezeigt, wie das Konzept der parametrisierten Platzhalter genutzt werden kann, um Module flexibel in ihrem Aufbau und somit besser wiederverwendbar zu gestalten.

3.4.1 Wiederverwendung von Lernobjekten mit Hilfe von Parametern

Parametrisierung ist ein wesentlicher Mechanismus zur Wiederverwendung von Lernobjekten, um zum Beispiel aus einer einzigen Rahmenkomponente mehrere verschiedene Instanzkomponenten ableiten zu können. Grundlage für die Wiederverwendung von Lernobjekten ist die Unterteilung in Rahmenkomponenten und Instanzkomponenten, wie sie in 3.2 beschrieben wurde: Rahmenkomponenten definieren anhand von Parametern, welche Eigenschaften die Lernobjekte haben sollen, aus denen sie aufgebaut sind. Man kann sich Rahmenkomponenten als eine Folge von Platzhaltern, an denen Subelemente (Instanz- oder wiederum Rahmenkomponenten) eingefügt werden, vorstellen, wobei an den Platzhaltern die Eigenschaften in Form von Parametern vermerkt sind, die die Subelemente erfüllen müssen.

Das Prinzip, Subelemente anhand ihrer Eigenschaften zu fordern, kann sehr flexibel genutzt werden: Es können in den Platzhaltern beliebige, die Subelemente beschreibende Eigenschaften verwendet werden. Die Rahmen- oder Instanzkomponenten, die diese Eigenschaften erfüllen, werden dann in den entsprechenden Platzhalter in die jeweilige Rahmenkomponente eingefügt[24]. Dieser Vorgang wird wiederholt durchlaufen, wenn in eine Rahmenkomponente nicht nur Instanzkomponenten, sondern weitere Rahmenkomponenten eingesetzt werden.

Beispiel:

> Ein Modul ist so definiert, dass es aus einer Sequenz folgender Subelemente besteht: Zuerst kommt ein Platzhalter, der mittels Parametern festlegt, dass das einzusetzende Lernobjekt kurz und anschaulich den Begriff „Rechnernetz“ definieren muss (Inhalt = „Rechnernetz“, Lernobjekttyp = „Definition“, Detaillierungsgrad = „leicht“). Dann folgt ein Platzhalter für ein Lernobjekt, das als Beispiel das Tokenring-Verfahren nennen und grob erklären muss (Inhalt = „Tokenring“, Lernobjekttyp = „Beispiel“, Detaillierungsgrad = „mittel“). Der letzte Platzhalter ist für ein Lernobjekt, das Verweise auf vertiefende Lernobjekte (Lernobjekttyp = „Vertiefung“, Detaillierungsgrad = nicht definiert) enthält.
>
> In jeden dieser Platzhalter können nun alle Lernobjekte eingesetzt werden, die die geforderten Eigenschaften erfüllen. In diesem Fall liegt die Auswahl eines bestimmten Lernobjekts beim Autor bzw. beim Lerner[25]. Eine feste Zuordnung eines bestimmten Lernobjekts zu einem Platzhalter ist weiterhin möglich, wobei man dadurch aber Flexibilität verschenkt.

---

[24] Dies wird durch einen Abgleich der Parameter des Platzhalters und der Lernobjekte realisiert. In 3.6 wird dieser Vorgang im Detail beschrieben.

[25] Dies wird in 3.6 anhand eines Beispiels verdeutlicht.

Um zu verstehen, wie Wiederverwendung über Parameter von Rahmen- und Instanzkomponenten realisiert wird, müssen zwei Themenkreise betrachtet werden: Die Parametrisierung der Lernobjekte, die in Rahmenkomponenten eingefügt werden sollen (Beschreibung der Eigenschaften der einzufügenden Lernobjekte), und die flexible Zuordnung von Lernobjekten zu den Platzhaltern, in die sie eingefügt werden können (das tatsächliche Einfügen der Lernobjekte). Letzteres wird über ein Matching von Parametern sowie die Parametrisierung von Beziehungen realisiert.

3.4.1.1 Beschreibung von Eigenschaften von Lernobjekten und Beziehungen durch Parameter

Jedes Lernobjekt (Instanz- sowie Rahmenkomponente) wird durch eine Menge von Parametern beschrieben. Jedes Lernobjekt hat einige generelle Eigenschaften, die durch Parameter entsprechend LOM.General beschrieben werden. Eine Übersicht liefert Tabelle 3-2.

Parameter	Datentyp	Bedeutung
LOM.General.Title	LangString	Name des Datenobjekts
LOM.General.Description	LangString	Textuelle Beschreibung
LOM.General.Language	CharacterString	Sprache
LOM.General.Keywords	LangString	Stichworte, die das Lernobjekt beschreiben

Tabelle 3-2: Parameter zur Beschreibung genereller Eigenschaften von Lernobjekten (Ausschnitt)

Im folgenden wird beschrieben, welche Gruppen von Parametern für die Beschreibung von Lernobjekten und / oder Beziehungen definiert sind:

- Beschreibung des Inhalts

  Um den konkreten Inhalt eines Lernobjekts zu beschreiben, werden die Parameter „Abstraktionsgrad“, „Schwierigkeitsgrad“, „Detaillierungsgrad“, „Thema“ und „Vorwissen“ benötigt, welche folgende Wertebereiche erhalten:

  ```
  domain(„Abstraktionsgrad“) = {„niedrig“,„mittel“,„hoch“}
  domain(„Detaillierungsgrad“) = {„niedrig“,„mittel“,„hoch“}
  domain(„Schwierigkeitsgrad“) = {0, 1, ..., 5}[26]
  domain(„Thema“) ⊆ Inhaltskategorie
  domain(„Vorwissen“) ⊆ Inhaltskategorie[27]
  ```

  Während die Parameter „Abstraktionsgrad“ und „Detaillierungsgrad“ einen einfachen Wert (beispielsweise „mittel“ oder „4“) annehmen können, wird das dem Lernobjekt assoziierte Thema und Vorwissen in Form von Inhaltskategorien modelliert. Eine Inhaltskategorie umfasst in der Konzeptebene alle Lernobjekte eines bestimmten Themengebiets. Beispielsweise gehört das Modul „Ethernet“ oder „Tokenring-Verfahren“

[26] Man beachte, dass die Angabe eines Schwierigkeitsgrads nur im Zusammenhang in einer gegebenen Nutzungsart sinnvoll ist. Dies liegt daran, dass die Bewertung eines Schwierigkeitsgrads von der vorgesehenen Zielgruppe abhängt.

[27] Hierzu kann das Vorwissen auch über eine Assoziation „ist Vorwissen von“ modelliert werden.

zur Inhaltskategorie „Rechnernetze“, welche wiederum zur Inhaltskategorie „Technische Informatik“ gehört (vgl. Abbildung 3-9).[28]

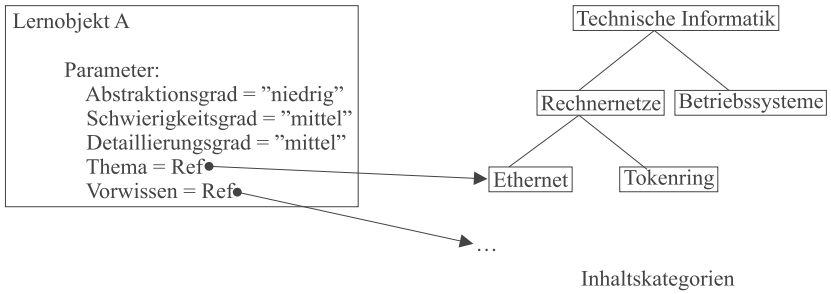


Abbildung 3-9: Zusammenspiel zwischen Parametern und Inhaltskategorien

Die Parameter „Thema“ und „Vorwissen“ gehören eng zusammen, denn es ist leicht denkbar, dass mehrere Lernobjekte zur selben Inhaltskategorie gehören, aber gänzlich verschiedenes Vorwissen benötigen. Dieser Fall läge beispielsweise bei einer Behandlung des Themas „Ethernet“ vor, für das zwei Lernobjekte existieren: Eines, das in der Vorlesung „Einführung in die Informatik für Fachfremde“ verwendet wird, und eines, das im Praktikum „Rechnernetze“ eingesetzt wird. Im ersten Fall kann kaum von Vorwissen ausgegangen werden – weder was grundlegende Konzepte der Informatik angeht, noch in bezug auf Rechnernetze, wohingegen im zweiten Fall zumindest fundierte Grundlagen der Informatik vorausgesetzt werden können. Um derlei zu vermeiden, muss immer sowohl das Thema als auch das Vorwissen eines Lernobjekts spezifiziert werden. Dies erlaubt es darüber hinaus auch, dass ein Lernobjekt in mehrere verschiedene Kategorien aufgenommen und so flexibler genutzt werden kann.

Parameter	Datentyp	Bedeutung
LOM.Content.Difficulty	Vocabulary	Schwierigkeitsgrad auf einer Skala von 0 (leicht) bis 5 (schwer)
LOM.Content.Abstraction	Vocabulary	Abstraktionsgrad auf einer Skala von 0 (konkret) bis 5 (abstrakt)
LOM.Content.Detail	Vocabulary	Detaillierungsgrad auf einer Skala von 0 (wenig detailliert) bis 5 (sehr detailliert)
LOM.Content.Subject	Vocabulary	Thema des Lernobjekts
LOM.Content.PreviousKnowledge	Vocabulary	Benötigtes Vorwissen für das Lernobjekt

Tabelle 3-3: Parameter zur Beschreibung des Inhalts von Lernobjekten

Teilweise ähneln die gerade beschriebenen Parameter denen, die in LOM.Educational definiert sind. Im Unterschied zu LOM.Educational betreffen die hier genannten Para-

[28] Ontologien stellen eine gern genutzte Möglichkeit dar, Inhaltskategorien zu definieren. Dieser Ansatz ist flexibler und mächtiger – aber auch deutlich komplexer – als die Teilmengenbeziehung, die hier beispielhaft verwendet wird.

meter aufgrund der strengen Trennung zwischen Inhalt und Lehrmaterialeinsatz vorerst nur den Inhalt der Lernobjekte. Daher werden die Schemata von LOM um die Kategorie „LOM.Content“ entsprechend Tabelle 3-3 erweitert.

- Beschreibung der inhaltlichen Aufgabe eines Lernobjekts

  Die inhaltliche Aufgabe beschreibt die „Rolle“ eines Lernobjekts im Kurs, beispielsweise ob ein Lernobjekt eine Definition oder ein Beispiel ist. Sie kann über die Lernobjekttypen beschrieben werden. Folgende Typen von Lernobjekten sind definiert: „Aspekt“, „Beispiel“, „Motivation“, „Einleitung“ und „Zusammenfassung“. Der Lernobjekttyp „Aspekt“ kann seinerseits weiter unterteilt werden in „Definition“, „Erklärung“, „Beschreibung“ und „Hintergrund“. Der Lernobjekttyp wird mit dem Parameter „kind“ beschrieben, wobei „kind“ auf eine Strukturierungskategorie (vgl. 3.2.1.3) verweist.

  Dieser Parameter ist abgesehen von Lernobjekten ebenso für Beziehungen definiert. Er definiert dann, welche Rolle das Lernobjekt, zu dem die Beziehung besteht, im aktuellen Kontext einnimmt. Dies erlaubt auch Konstrukte, bei denen Lernobjekte auf unübliche Weise eingesetzt werden, beispielsweise wenn in einem Anfängerkurs ein Lernobjekt vom Typ „Erklärung“ als „Definition“ eingesetzt wird (vgl. Abbildung 3-10). In diesem Fall hat die Beziehung einen Parameter „kind = Definition“, wodurch die Rolle des Lernobjekts eindeutig geklärt wird.

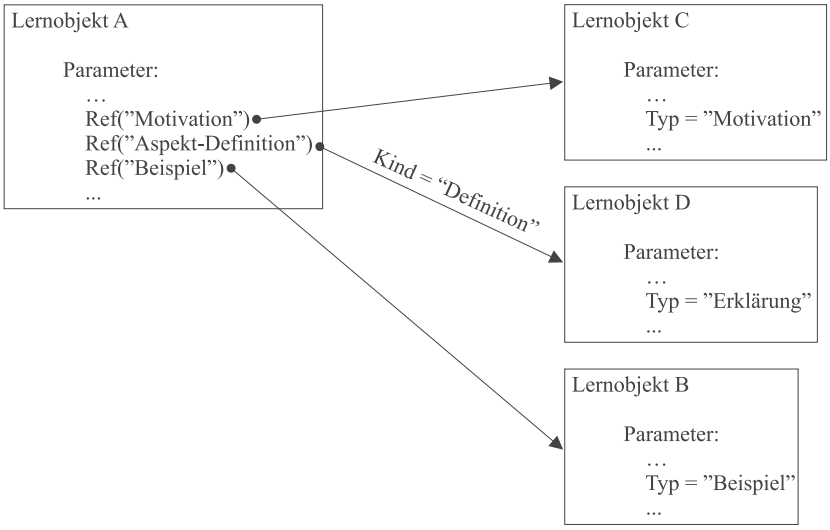


Abbildung 3-10: Typen von Lernobjekten und Beziehungen

In LOM wird über die Kategorie „Relation“ die Art der Beziehung zu anderen Lernobjekten definiert. Diese wird ebenfalls in das Datenmodell übernommen. In LOM.Relation.Resource.Identifier wird der Identifikator des Ziel-Lernobjekts der Relation gespeichert. Der Parameter „Kind“ beschreibt die Rolle des Ziel-Lernobjekts.

Parameter	Datentyp	Bedeutung
LOM.Relation.Kind	Vocabulary	Art der Beziehung
LOM.Relation.Target	LangString	Bezeichnung des Ziels

Tabelle 3-4: Parameter zur Beschreibung der Beziehung von Lernobjekten

- Beschreibung der didaktischen Aufgabe eines Lernobjekts

  Diese Parameter beschreiben, wie ein Lernobjekt in der Lehre genutzt wird. Es können hier folgende Typen von Lernobjekten unterschieden werden: „Wissensvermittlung“, „Übung“ und „Verweis“. Diese verweisen auf eine didaktische Kategorie (vgl. 3.2.1.3).

  Diese Parameter sind ebenso für Beziehungen hinsichtlich der Kursstrukturierung definiert. Sie definieren dann, welche didaktische Funktion das Lernobjekt, zu der die Beziehung besteht, in der Anwendung in der Lehre einnimmt. So könnte beispielsweise in einem Lehrbuch über das Schreiben wissenschaftlicher Arbeiten eine besonders gut gelungene Definition eines Begriffs als lobendes Beispiel, wie eine gute Definition aussehen kann, aufgeführt sein. Dies würde in einem Lernobjekt der Kategorie „Wissensvermittlung“ realisiert sein; die gut gelungene Definition (Strukturkategorie „Definition“) würde über eine Beziehung der Strukturkategorie „Beispiel“ an das Lernobjekt gebunden sein.

  Zur Beschreibung dieser Aspekte können die in Tabelle 3-4 beschriebenen Parameter verwendet werden.

Diese Arten von Parametern sind orthogonal zueinander und können kombiniert werden. So ist es möglich, verschiedene Aspekte eines Lernobjekts im selben Satz von Parametern zu beschreiben und so eine kompakte Darstellung der Beschreibung zu erreichen.

Die gerade beschriebenen Parameter müssen mindestens vorhanden sein, damit eine ausreichend flexible Wiederverwendbarkeit der Lernobjekte sichergestellt werden kann. Daneben sind natürlich weitere Parameter denkbar, die den Lernobjekten von den Autoren zugeordnet werden können. Beispiele für weitere Parameter sind in [LOM] und [LTSC] genannt.

Im weiteren Fortgang dieser Arbeit – insbesondere in den Kapiteln 6 und 7 – werden weitere Parameter eingeführt, die im wesentlichen technische Funktion haben. Diese Parameter können hier an dieser Stelle ignoriert werden.

3.4.1.2 Parametrisierung der Platzhalter von Rahmenkomponenten

Rahmenkomponenten enthalten keinen eigenen Inhalt. Statt dessen definieren sie eine Folge von Platzhaltern, in die passende Lernobjekte – Instanzkomponenten oder selbst wieder Rahmenkomponenten – eingesetzt werden. Welche Lernobjekte in einen Platzhalter passen, wird wiederum über Parameter festgelegt: Die Parameter, die für ein Lernobjekt zwingend vorgeschrieben sind, werden dabei in den jeweiligen Platzhaltern der Rahmenkomponente festgelegt.

Es ist bei den Parametern eines Platzhalters einer Rahmenkomponente natürlich möglich, direkt den Identifikator des Subelements (Instanzkomponente oder Rahmenkomponente) anzugeben und damit vorzuschreiben, dass nur ein ganz bestimmtes Lernobjekt in den Platzhalter eingesetzt werden darf. Damit verliert man allerdings die Möglichkeit, ein Lernobjekt gegen ein anderes mit gleicher Charakteristik auszutauschen. Beschreibt man die erlaubten Lernobjekte jedoch ausschließlich über Parameter, so bleibt diese Flexibilität erhalten.

Die Parameterwerte, die in den Platzhaltern einer Rahmenkomponente gesetzt werden, hängen von den Parametern der Rahmenkomponente selbst ab. Die Parameterwerte in den Platzhaltern, in die Subelemente eingesetzt werden sollen, dürfen im allgemeinen strenger als die entsprechenden Parameterwerte der Rahmenkomponente sein, nicht hingegen schwächer. Wird beispielsweise für eine Rahmenkomponente der Parameter LOM.Content.Difficulty=3 gesetzt um zu beschreiben, dass die Rahmenkomponente maximal mittelschwer ist, so muss für alle Platzhalter in der Rahmenkomponente LOM.Content.Difficulty ≤ 3 gelten: Die Sub-

elemente dürfen leichter oder genauso schwer wie die Rahmenkomponente selbst sein, nicht jedoch schwerer. Diese Abhängigkeiten zwischen den Parametern der Rahmenkomponente und den Parametern der Platzhalter werden explizit vom Autor der Rahmenkomponente mittels Constraints festgeschrieben. So bleibt es möglich, in bestimmten Ausnahmefällen auch Subelemente mit anderen Eigenschaften (festgeschrieben in Parameterbelegungen) zu verwenden.

3.4.1.3 Constraints zur Erleichterung der Wiederverwendung von Lehrmaterialien

Parameter existieren nicht isoliert voneinander, sondern sind in einigen Fällen voneinander abhängig. Diese Abhängigkeiten von Parametern untereinander können verwendet werden, um zusätzliche Freiheiten beim Aufbau von Lernobjekten zu schaffen. Hierzu können Constraints verwendet werden.

Ohne Constraints müssen alle Parameter von Platzhaltern eines Moduls mit Werten belegt sein, damit passende Subelemente gesucht werden können[29]. So wird die Menge der Subelemente, die in einen Platzhalter eingesetzt werden können, beschränkt, denn für einen konkreten Parameter wird dann nur ein Wert angegeben, nicht eine Menge möglicher Werte. Werden bei der Erstellung und Beschreibung von Lernobjekten durch Autoren jedoch Constraints eingesetzt, müssen bei jeweils zwei Parametern, die voneinander abhängen, nicht mehr alle Parameter belegt werden. Durch die Constraints, die die Beziehungen zwischen den Parametern formalisieren, können die Werte der Parameter aneinander gekoppelt sein, so dass es ausreicht, wenn einer der beiden Parameter belegt wird. Für den anderen Parameter können Werte aus einer Menge eingesetzt werden, die durch die Constraint zwischen den beiden Parametern definiert ist.

Dies hat weitreichende Folgen für die Auflösung von Beziehungen zwischen Rahmenkomponenten und ihren Subelementen, die in sie eingesetzt werden sollen: Durch die so realisierte Vergrößerung des Wertebereichs der möglichen Parameterwerte kann die Zuordnung einer Beziehung zu ihren möglichen Ziel-Lernobjekten flexibler gestaltet werden, denn nun können entsprechend der Constraint *n* verschiedene Werte[30] angenommen werden, nicht mehr nur ein bestimmter Wert. Die Zahl der möglichen Ziel-Lernobjekte erhöht sich auf diese Weise, so dass den Autoren beim Aufbau komplexer Lernobjekte mittels Komposition mehr Subelemente zur Auswahl stehen. Auf den Vorgang, wie aus Rahmenkomponenten konkrete Lernobjekte (Instanzkomponenten) werden und insbesondere, wie die Auflösung der Beziehungen zwischen einem Platzhalter und den passenden Subelementen geschieht, wird in Abschnitt 3.6 im Detail eingegangen.

3.4.2 Alternativenbildung

Neben einer Flexibilisierung der Beziehungen zwischen Modulen und ihren Subelementen, wie sie in 3.4.1 beschrieben wurde, stellt die Parametrisierung der Lernobjekte den Ausgangspunkt zur Erstellung von Alternativen von Lernobjekten dar. Lernobjekte, die inhaltlich äquivalent sind, die also bezüglich der Parameter, die den Inhalt des Lernobjekts beschreiben, identisch sind, werden als *Alternativen* bezeichnet. Alternativen stellen den gleichen Inhalt auf verschiedene Weise dar. Dies betrifft die Darstellung des Inhalts (textuelle Darstellung gegenüber Animationen, etc.), den Detaillierungsgrad oder den Abstraktionsgrad

[29] Die Verwendung von Wildcards bei Parameterwerten von Platzhaltern wäre auch denkbar. So könnte für einen Parameter auch die Menge der passenden Subelemente erhöht werden. Mit Wildcards kann allerdings nur geprüft werden, ob der Parameterwert einer Subkomponente in seiner Syntax einem bestimmten Muster entspricht, nicht jedoch, ob der Parameterwert bestimmten logischen Bedingungen genügt. Es bestünde daher die Gefahr, dass so Lernobjekte kombiniert werden, die aufgrund von Thema, Struktur, etc. nicht zueinander passen.

[30] Die *n* verschiedenen Inhalte der Menge von Parameterwerten, die entsprechend der Constraint erlaubt sind

– respektive die dies beschreibenden Parameter. In diesen Punkten unterscheiden sich die einzelnen Alternativen.

Alternativenbildung ist eine Möglichkeit zur Erhöhung der Wiederverwendbarkeit, denn sie ermöglicht vor allem die Austauschbarkeit von inhaltlich und / oder strukturell äquivalenten Lernobjekten und die Konfigurierbarkeit von Beziehungen.

Alternativen gestatten ferner den Autoren gewisse Freiheitsgrade, indem sie alle möglichen sichtbaren Ausprägungen von Inhalt und Struktur der Lernobjekte als austauschbar spezifizieren, wobei die endgültige Auswahl erst in der Instanzebene erfolgt. Ein Beispiel hierzu sind die verschiedenen möglichen Werte des Parameters „Detaillierungsgrad“ zu ein und demselben Inhalt: Die Anzahl der Alternativen entspricht dem Wertebereich von „Detaillierungsgrad“.

3.5 Betrachtung von Nutzungsaspekten

3.5.1 Motivation

Wiederverwendung umfasst nicht nur das Einsetzen passender Instanzkomponenten (anhand von Parametern), deren Austausch (durch Alternativen, wie im letzten Abschnitt beschrieben) und (Re-)Konfiguration. Wiederverwendung umfasst ebenso die nutzungsbezogene Spezialisierung von Instanzkomponenten (vgl. 2.4), im folgenden Adaption genannt. Beispielsweise soll ein und derselbe Kurs für verschiedene Zielgruppen eingesetzt werden können. In diesem Fall gäbe es von einem Kurs mehrere Adaptierungen: eine Adaptierung je Zielgruppe.

Lernobjekte und Beziehungen können durch eine weitgehende Spezifizierung und Belegung der Parameter sehr präzise beschrieben werden, wodurch ihr Zweck und ihre möglichen Einsatzgebiete klar umrissen und irrtümliche Verwendungen in Lehrmaterialien vermieden werden. Dies erleichtert die Nutzung der Lernobjekte und erhöht so deren Wiederverwendbarkeit. Aber ist die Wiederverwendung von Lernobjekten wirklich beliebig? Unter welchen Bedingungen darf ein Lernobjekt überhaupt wiederverwendet werden? Gibt es bestimmte Nutzungsarten bzw. -szenarien, die einander ausschließen? Ziel muss sein, die Menge der Parameter und deren Werte für jeweils eine bestimmte Nutzung sukzessiv herauszuarbeiten und / oder anzupassen. Dies erreicht man dadurch, dass man anhand der Nutzungen die Verwendung der betreffenden Lernobjekte immer detaillierter beschreibt (vgl. 2.4.2.2).

3.5.2 Nutzungsklassen

Die Erstellung von Lernobjekten steht stets mit gewissen Nutzungsszenarien in enger Verbindung. Anhand dieser Szenarien werden Parameter von Lernobjekten identifiziert, falls nötig neu definiert und die Werte der Parameter möglichst treffend belegt.

Viele Nutzungsszenarien sind einander sehr ähnlich und erfordern ähnliche oder sogar identische Parameter und Parameterwerte. Solche Nutzungsszenarien können daher als äquivalent betrachtet werden. Zur Strukturierung der Nutzungsszenarien werden *Nutzungsklassen* eingeführt: Äquivalente Nutzungsszenarien werden in einer Nutzungsklasse zusammengefasst. Nutzungsklassen sind ein Strukturierungsmittel der Konzeptebene.

Entsprechend der verschiedenen Nutzungsszenarien gibt es mehrere verschiedene Nutzungsklassen. Nutzungsklassen dienen zur Klassifizierung der Nutzung von Lernobjekten und zu ihrer Einordnung gemäß der unterschiedlichen Nutzungen. Beispielsweise können Nutzungsklassen anhand ihrer Zielgruppe oder anhand der Art der Lehrveranstaltung, in der sie eingesetzt werden, spezifiziert werden. Die Nutzungsklassen sind damit die zentralen Strukturie-

rungselemente für Rahmen- und Instanzkomponenten hinsichtlich ihrer Nutzung. Im folgenden werden die für die Erstellung von Lernobjekten bedeutsamen Nutzungsklassen vorgestellt.

3.5.2.1 Nutzungsklasse „Lehre“

Die Nutzungsklasse „Lehre“ beschreibt Anwendungen von inhaltlich und strukturell abgeschlossenen Lernobjekten in der Lehre, beispielsweise die Lehrform, in deren Rahmen das Lernobjekt eingesetzt wird. Die Lehrmaterialien werden hierbei aus der Sicht des Dozenten betrachtet.

Die Nutzungsklasse „Lehre“ spezifiziert folgende Parameter:

- Parameter „Lehrform“

  Beschreibt, wie die Lehrveranstaltung ausgestaltet wird, in der die Lernobjekte eingesetzt werden sollen. Derzeit definierte Werte sind „Vorlesung“, „Übung“, „Selbststudium“.

- Parameter „Lehrziel“

  Beschreibt, welche Lehrinhalte den Lernern in den Lehrveranstaltungen mittels der Lernobjekte vermittelt werden. Die Lehrinhalte werden, wie in 3.4.1.1 beschrieben, mittels Inhaltskategorien modelliert.

- Parameter „Lehrmethode“

  Beschreibt die pädagogische Ausrichtung der Lernobjekte, beispielsweise ob sie eher theorieorientiert, übungsorientiert, etc. sind. Dementsprechend sind folgende Werte definiert: „Theorie“, „Übung“, „Beispiel“.

- Parameter „Wissensstand“

  Beschreibt das Vorwissen der Lerner. Das Vorwissen ist wieder, wie in 3.4.1.1 beschrieben, mittels Inhaltskategorien modelliert.

Um die Nutzungsklasse „Lehre“ beschreiben zu können, wird die LOM-Kategorie „Educational“ um eine Reihe von Elementen erweitert, die für den Einsatz von Lernobjekten in der Lehre bzw. die Zuordnung von Inhalten für die Lehre von Bedeutung sind. In Tabelle 3-5 werden die benötigten nutzungsklassenspezifischen Parameter dargestellt.

Parameter	Datentyp	Bedeutung
LOM.Educational.Form	Vocabulary	Art der Lehrveranstaltung
LOM.Educational.Target	Vocabulary	Lehrinhalte der Lehrveranstaltung
LOM.Educational.Approach	Vocabulary	Pädagogische Ausrichtung
LOM.Educational.Knowledge	Vocabulary	Vorwissen der Lerner

Tabelle 3-5: In der Nutzungsklasse „Lehre“ spezifizierte Parameter

3.5.2.2 Nutzungsklasse „Lernen“

Die Nutzungsklasse „Lernen“ beschreibt die Anwendung von inhaltlich und strukturell abgeschlossenen Lernobjekten im Selbststudium, wobei Merkmale des Lerners berücksichtigt werden. Die Lernobjekte werden hierbei aus der Sicht der Lerner betrachtet.

Die Nutzungsklasse „Lernen“ spezifiziert folgende Parameter:

- Parameter „Lerntyp“

  Beschreibt die Möglichkeiten der bevorzugten Informationsaufnahmen der Lerner oder Gruppe, beispielsweise „Sehen“, „Hören“ oder „Eigene Handlungen“ [31].

- Parameter „Lernmethode“

  Beschreibt die pädagogische Ausrichtung der Lernobjekte, beispielsweise ob sie eher theorieorientiert, übungsorientiert, etc. sind. Dementsprechend sind folgende Werte definiert: „Theorie“, „Übung“, „Beispiel“.

- Parameter „Wissensstand“

  Beschreibt das Vorwissen der Lerner. Das Vorwissen ist wieder, wie in 3.4.1.1 beschrieben, mittels Inhaltskategorien modelliert.

Um die Nutzungsklasse „Lernen“ beschreiben zu können, wird die LOM-Kategorie „Educational“ um eine Reihe von Elementen erweitert, die für den Einsatz von Lernobjekten in der Lehre bzw. die Zuordnung von Inhalten für das Lernen von Bedeutung sind. In Tabelle 3-6 werden die benötigten nutzungsklassenspezifischen Parameter dargestellt. Einige dieser Parameter sind bereits in der Nutzungsklasse „Lehre“ erwähnt, vgl. Tabelle 3-5.

Parameter	Datentyp	Bedeutung
LOM.Educational.Type	Vocabulary	Informationsaufnahme
LOM.Educational.Approach	Vocabulary	Pädagogische Ausrichtung
LOM.Educational.Knowledge	Vocabulary	Vorwissen der Lerner

Tabelle 3-6: In der Nutzungsklasse „Lernen“ spezifizierte Parameter

3.5.2.3 Nutzungsklasse „Präsentation“

Die Nutzungsklasse „Präsentation“ umfasst alle Parameter, die die beabsichtigte Nutzungs- und Präsentationsform der Lernobjekte beschreiben. Diese Nutzungsklasse ist insbesondere für Kurse und Kursteile von Bedeutung, hingegen weniger für einzelne Module. Die Nutzungsklasse „Präsentation“ spezifiziert dabei folgende Parameter:

- Parameter „Navigation“

  Beschreibt die Struktur, entsprechend der die Navigation für die Lerner stattfindet. Derzeit sind die Werte „Sequenz“ und „Hypertext“ definiert.

- Parameter „Medientyp“

  Beschreibt, ob statische oder auch dynamische Medientypen verwendet werden dürfen. Derzeit definierte Werte sind „statisch“ und „dynamisch“.

- Parameter „Datenformat“

  Beschreibt das Datenformat sowie das Layout[32], in dem die konkreten Lehrmaterialien erzeugt werden sollen, beispielsweise PDF, Postscript, HTML, XML, etc.

---

[31] Aus der Hirn- und Gedächtnisforschung, die Frederic Vester in seinem Buch „Denken, Lernen, Vergessen“ vorstellt, sind vier Lerntypen bekannt: Der visuelle Lerntyp, der auditive Lerntyp, der praktische Lerntyp und der gemischte Lerntyp [Ves04].

[32] Wie zum Beispiel die Schriftart bei Texten, die Größe und Farbtiefe bei Grafiken oder die frame rate bei Videos.

Hier kann die LOM-Kategorie „Presentation“ erweitert werden, vgl. Tabelle 3-7.

Parameter	Datentyp	Bedeutung
LOM.Presentation.Navigation	Vocabulary	Vom Autor vorgesehene Struktur der Navigation
LOM.Presentation.Mediatype	LangString	Erlaubter Medientyp (statisch oder dynamisch)
LOM.Presentation.Dataformat	LangString	Ausgabeformat

Tabelle 3-7: In der Nutzungsklasse „Präsentation“ spezifizierte Parameter

3.5.2.4 Zusammenhang zwischen Nutzungsklassen und Constraints

Die Beziehungen und Abhängigkeiten zwischen den Parametern innerhalb einer oder verschiedener Nutzungsklassen müssen präzise modelliert und dokumentiert werden. Andernfalls werden die Folgen einer „einfachen“ Parameteränderung in einer Nutzungsklasse für die Autoren der Lernobjekte nicht mehr beherrschbar – durch widersprüchliche oder unlogische Parameterbelegungen könnten inkonsistente oder pädagogisch unsinnige Lernobjekte entstehen.

Eine Lösung für dieses Problem bieten Constraints auf mehreren Parametern: Die Werte von voneinander abhängigen Parametern werden über Constraints so verknüpft, dass nur solche Werte möglich sind, die in Kombination über alle Nutzungsklassen hinweg sinnvoll sind. Constraints werden damit nicht mehr nur für die Kontrolle der Werte von (voneinander isolierten) Parametern eingesetzt, sondern sie sorgen dafür, dass die Abbildung von nutzungsunabhängigen Lernobjekten in nutzungsabhängige Lernobjekte ohne große Überraschungen für die Autoren abläuft.

Beispiel:

> Um die Abhängigkeiten der Parameter „Lehrform“ („form“) und „Lehrmethode“ („approach“) zu modellieren und korrekte Werte für beide Parameter sicherzustellen, könnten folgende Constraints definiert werden:
>
> ```
> value(„form“)=„Vorlesung“ ⇒ value(„approach“)∈{„Theorie“,
> „Erklärung“}
> ```

3.5.3 Nutzung von Nutzungsklassen für die Wiederverwendung von Lernobjekten

Lernobjekte können nicht nur hinsichtlich ihres Inhalts und ihrer Struktur, sondern auch hinsichtlich ihrer Nutzung parametrisiert werden. Hierzu werden sie mit Parametern der oben beschriebenen Nutzungsklassen versehen. So ist es beispielsweise durch Variation der Parameter „Lehrform“ und „Lerntyp“ möglich, aus ein und demselben Lernobjektbaum Kurse mit unterschiedlichem Detaillierungsgrad oder unterschiedlichem Abstraktionsgrad für verschiedene Nutzungen, repräsentiert durch unterschiedliche Lehrform und Lerntyp, zu erstellen, wobei die in den Kursen vermittelten Inhalte aber immer die selben bleiben. Ebenso ist es möglich, die so erstellten Kurse je nach Belegung der Parameter der Nutzungsklasse „Präsentation“ in verschiedene Darstellungen in verschiedenen Präsentationsformaten zu transformieren, was den Einsatz ein und desselben Kurses auf unterschiedliche Weise – vorlesungsbegleitend, als Skript zum Selbststudium, für die Präsentation einer Vorlesung mit einem Beamer, etc. – ermöglicht.

Voraussetzung für eine gewinnbringende Parametrisierung ist allerdings, dass zum Zeitpunkt der Parametrisierung die Nutzung der Lernobjekte hinsichtlich der Nutzungsklassen „Lehre“

und „Lernen“ größtenteils festgelegt und somit überhaupt mittels Parametern beschreibbar ist. Dies ist im allgemeinen kein Problem, da normalerweise die Art der Lehrveranstaltung feststeht, wenn die Lehrmaterialien für die Lehrveranstaltung zusammengestellt werden. Die Parameter der Nutzungsklasse „Präsentation“ werden von den Autoren erst dann gesetzt, wenn der Kurs fertig erstellt und zur Transformation ins endgültige Präsentationsformat bereit ist.

Nutzungsklassen und die durch sie definierten Parameter steuern so die Abbildung von nutzungsunabhängigen Lernobjekten in nutzungsabhängige Lernobjekte inklusive aller ihrer Subelemente (vgl. Abbildung 3-11).

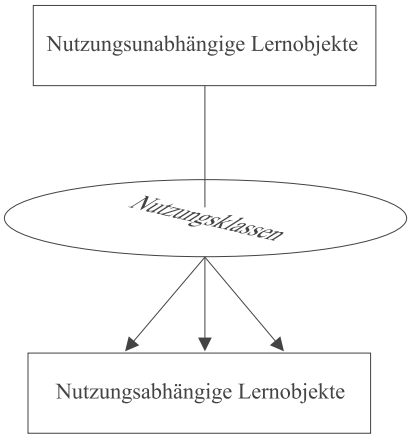


Abbildung 3-11: Abbildung von nutzungsunabhängigen Lernobjekten in nutzungsabhängige Lernobjekte

In diesem Zusammenhang ist der Begriff der *Variante* zu sehen: Varianten sind Lernobjekte, die zwar die selben Inhalte behandeln, die aber in verschiedenen Nutzungen angewandt werden. Beispielsweise wären zwei Lernobjekte, die beide das *CSMA/CD-Verfahren* beschreiben, aber verschiedene Präsentationsformen (HTML und PDF) haben, Varianten. *Kursvarianten* sind eine besondere Art von Varianten: Sie haben eine größere Granularität, denn sie betreffen nicht Lernobjekte, sondern Kurse. Kursvarianten sind Kurse desselben Inhalts, die aber – analog zum Begriff der Varianten auf Lernobjektebene oben – in verschiedenen Nutzungen der Nutzungsklassen angewandt werden.

Im Gegensatz zu Alternativen, wie sie in 3.4.2 beschrieben sind, werden Varianten (von Lernobjekten wie von Kursen) Nutzungen zugeordnet; sie sind damit von ihrer jeweiligen Nutzung in der Lehre abhängig. Beim Begriff der Alternative werden Nutzungsaspekte vor der Auswertung von Nutzungsklassen nicht berücksichtigt.

3.6 Von Lernobjekten zu konkreten Lehrmaterialien

Nachdem nun definiert ist, welchen Aufbau Lernobjekte haben und auf welche Weise sie beschrieben werden können, soll nun betrachtet werden, wie aus einem Geflecht parametrisierter Lernobjekte konkrete Lehrmaterialien (hier: Kurse), die in Lehrveranstaltungen eingesetzt werden können, erzeugt werden. Für die Autoren soll der Aufwand hierzu so gering wie möglich gehalten werden, und gleichzeitig muss so viel Flexibilität bezüglich der Nutzungsmöglichkeiten der Lernobjekte wie möglich erhalten bleiben.

Hierbei wird davon ausgegangen, dass der Autor einen Kurs für eine Lehrveranstaltung erstellen will[33]. Die Zielgruppe, die Art, wie der Kurs bzw. der Kursteil den Lernern nahegebracht werden soll (als Skript zum Selbststudium, als vorlesungbegleitende Präsentation, etc.), evtl. der Schwierigkeitsgrad und weitere Anforderungen an die fertigen Lehrmaterialien sind hierbei bekannt.

Um von einer Menge von Atomen und Modulen zu einem einsatzfähigen Kurs zu kommen, sind mehrere Schritte nötig. Diese werden im folgenden vorgestellt. Dabei werden auch einige in diesem Kapitel bereits genannte Konzepte und Vorgehensweisen verdeutlicht.

3.6.1 Schritt 1: Parametrisierung der initialen Rahmenkomponente

Die Erzeugung konkreter Lehrmaterialien aus Rahmenkomponenten geht immer von einer „initialen" Rahmenkomponente RK aus, die vom Autor entsprechend seinen Vorstellungen und Zielen geeignet parametrisiert wird. Diese Rahmenkomponente ist meist in der Lernobjekthierarchie an recht hoher Stelle angesiedelt. Beispielsweise modelliert sie einen kompletten Kurs.

Der Autor legt nun die Parameter und Constraints der Rahmenkomponente fest, also insbesondere

- alle allgemeinen Parameter, beispielsweise Autor, etc. gemäß Tabelle 3-2
- alle Parameter zur Beschreibung des Inhalts und der inhaltlichen bzw. didaktischen Aufgabe des zu erstellenden Kurses bzw. Kursteils, beispielsweise Schwierigkeitsgrad, Detaillierungsgrad, etc. gemäß Tabelle 3-3
- alle Parameter zur Beschreibung der Lehrveranstaltung und des Selbststudiums, in dessen Rahmen der zu erstellende Kurs bzw. Kursteil genutzt werden soll (Nutzungsklassen „Lehre" und „Lernen"), gemäß Tabelle 3-5
- alle Parameter, die zur Darstellung der zu erstellenden, konkreten Lehrmaterialien nötig sind (Nutzungsklasse „Präsentation"), gemäß Tabelle 3-7
- alle Constraints auf den in den ersten vier Punkten genannten Parametern.

Damit ist die initiale Rahmenkomponente (RK) in allen ihren Parametern p_1, ... p_n hinsichtlich ihrer generellen Eigenschaften, aber auch hinsichtlich ihrer vom Autor vorgesehenen Nutzung beschrieben; Einschränkungen und Abhängigkeiten der Parameter sind mittels Constraints definiert. Um nach Anwendung der in den folgenden Abschnitten beschriebenen Schritte möglichst hochwertige Lehrmaterialien zu erhalten, müssen diese Parameter vollständig und so treffend wie möglich gesetzt werden.

Als nächstes werden die Constraints, die auf den Parametern von RK definiert sind, aufgelöst. Für jede Constraint mit einem Ausdruck Expr(p_1, ..., p_i) mit Parametern p_1, ..., p_i auf der linken Seite und mit Parametern q_1, ..., q_j auf der rechten Seite werden, falls Expr(p_1, ..., p_i) wahr ist, die q_1, ..., q_j entsprechend gesetzt:

Eine Constraint

$$\mathtt{Expr}(p_1, \ldots, p_i) \Rightarrow (q_1 = \mathtt{Expr}_1, \ldots, q_j = \mathtt{Expr}_j)$$

wird somit folgendermaßen aufgelöst:

[33] Die Schritte zur Erstellung von Modulen sind im wesentlichen die selben. Die Erstellung eines Kurses wurde lediglich aus Gründen der Anschaulichkeit gewählt: Kurse sind für die Vorstellung etwas greifbarer und konkreter als Module beliebiger Granularität.

$(Expr(p_1, ..., p_i) == true) \Rightarrow q_1 = Expr_1, ..., q_j = Expr_j,$

wobei $Expr_1$, ... $Expr_j$ auch Mengenzuweisungen, beispielsweise LOM.Content.Detail = {1, 2, 3}, oder wieder Ausdrücke, beispielsweise LOM.Content.Detail = {x | x<4}, sein können.

Falls hierbei festgestellt wird, dass ein Parameter bei der Auflösung der Constraints überschrieben würde, der bereits vom Autor gesetzt wurde, wird dieser Parameter nicht angetastet, sondern es wird lediglich geprüft, ob der vom Autor gesetzte Wert die Constraint erfüllt. Dies geschieht, indem die Auswertung der Constraint wie oben beschrieben durchgeführt wird und dann geprüft wird, ob in der berechneten Ergebnismenge der vom Autor gesetzte Wert vorhanden ist. Falls dies nicht der Fall ist, falls also der vom Autor gesetzte Parameterwert die Constraint nicht erfüllt, wird der Autor mit einer entsprechenden Fehlermeldung informiert. Der Autor muss nun entscheiden, ob er den Parameterwert korrigieren oder beibehalten will.

3.6.2 Schritt 2: Erstellung der Lernobjektstruktur

Schritt 2 besteht aus drei Teilschritten: Im ersten Teilschritt wird für jeden Platzhalter eine Menge potenziell passender Lernobjekte identifiziert. Im zweiten Teilschritt werden für jeden Platzhalter die passenden Lernobjekte in eine Rangfolge gebracht, um leichter Lernobjekte auswählen zu können, die dann konkret in die Platzhalter eingesetzt werden. Im dritten Teilschritt schließlich werden die Ergebnisse der Teilschritte 1 und 2 genutzt, um ein Geflecht von Lernobjekten aufzubauen, das in den weiteren Schritten für die endgültige Präsentation genutzt wird.

Alle diese Teilschritte werden für die initiale Rahmenkomponente RK sowie rekursiv für alle Module unterhalb von RK durchgeführt.

3.6.2.1 Teilschritt 1: Identifikation passender Subelemente

In Teilschritt 1 werden alle Subelemente identifiziert, die in die Platzhalter der Rahmenkomponente eingebunden werden können. Da die Subelemente im allgemeinen nicht direkt genannt sind, sondern lediglich die geforderten Eigenschaften mittels Parametern beschrieben werden, müssen zuerst für jeden Platzhalter passende Subelemente identifiziert werden. Für jeden Platzhalter $P(p'_1, ... p'_n)$ mit Parametern $p'_1, ... p'_n$ wird hierzu folgendermaßen vorgegangen:

1. Übernehmen der Werte der Parameter p_i der Rahmenkomponente in den Platzhalter, falls nötig, und Auflösen der Constraints:

   ```
   foreach (p'1, ... p'n)
     if (p'i == undefined) p'i = pi;
   ```

   Die Constraints werden analog zu Schritt 1 aufgelöst, vgl. 3.6.1.

   Dieser Schritt ist nötig, da für die zu ermittelnden Subelemente die gleichen Eigenschaften gelten sollen wie für die Rahmenkomponente. Wenn die Rahmenkomponente beispielsweise höchstens den Schwierigkeitsgrad 3 hat, dürfen die Subelemente keinen höheren Schwierigkeitsgrad haben.

   Bleibt hierbei ein Parameter undefiniert, so wird angenommen, dass dieser Parameter für die Verwendung des Kurses nicht von Bedeutung ist und so für die weiteren Schritte mit Hinblick auf eine Vergrößerung des Suchraums beliebige Werte annehmen kann.

2. Ermittlung passender Lernobjekte fittingObjs aus der Menge Pool aller verfügbaren Lernobjekte durch Vergleich der Parameter $obj.p_1$, ..., $obj.p_n$ des Lernobjekts obj mit

den Parametern p'_1, ... p'_n des Platzhalters. Hierbei wird davon ausgegangen, dass ein Objekt obj dann in einen Platzhalter passt, wenn für alle i, $i \in \{1, ..., n\}$, gilt:

p'_i == undefined or ($p'_i \neq$ undefined and p'_i == obj.p_i)

Damit kann folgender Algorithmus definiert werden:

```
fittingObjs = EmptySet;
foreach obj in Pool
  isFitting = true;
  foreach (obj.p1, ... obj.pn)
    if (p'i ≠ undefined and obj.pi ≠ p'i)
      isFitting = false;
  if (isFitting) fittingObjs.append(obj);
```

Es wird wieder angenommen, dass nicht gesetzte Parameter („undefined“) des Platzhalters für die spätere Anwendung des Kurses bzw. Kursteils keine Bedeutung haben.

Die Menge fittingObjs enthält nun alle Lernobjekte, die in den gegebenen Platzhalter eingesetzt werden können.

Für jeden Platzhalter gibt es nun je eine Menge fittingObjs, die alle in den Platzhalter passenden Lernobjekte beinhaltet. Man beachte, dass hier bereits sichergestellt ist, dass nur solche Lernobjekte in fittingObjs enthalten sind, die mit der späteren, geplanten Nutzung vereinbar sind (beispielsweise keine Animationen in einem zu druckenden Skript), denn das geplante Präsentationsformat spiegelt sich in den Werten der Parameter der Nutzungsklasse „Präsentation“[34] wider.

Im nächsten Teilschritt muss nun eine Rangfolge auf diesen Lernobjekten definiert werden um entscheiden zu können, welches Lernobjekt konkret in den Platzhalter eingesetzt wird.

3.6.2.2 Teilschritt 2: Festlegung einer Rangfolge unter gleichwertigen Subelementen

Jeder Platzhalter hat im allgemeinen mehrere passende Lernobjekte, die alle ähnliche Parameterwerte haben und die daher – eine saubere Parametrisierung vorausgesetzt – als strukturell und semantisch äquivalent betrachtet werden können. Diese Lernobjekte werden als Ergebnis des letzten Teilschritts (vgl. 3.6.2.1) für jeden Platzhalter in einer Menge fittingObjs gespeichert. Die Auswahl eines konkreten Lernobjekts findet nicht in diesem Schritt statt, sondern erst später in Teilschritt 3, vgl. 3.6.2.3. So bleibt man bis zuletzt offen für Änderungen in Darstellung und Stil des zu erstellenden Kurses bzw. Kursteils.

Es muss jedoch eine gewisse Vorauswahl getroffen werden, welche Lernobjekte aus der Menge fittingObjs für einen Platzhalter besser geeignet sind als andere. Dies ist nötig, da die Menge der passenden Lernobjekte potenziell sehr groß werden kann und ohne eine Einschätzung der Eignung einer Lernobjekts eine fundierte Auswahl für den Autor sehr schwer oder gar unmöglich wird. Hierzu müssen mehrere Kriterien berücksichtigt werden:

- Die Eignung eines Lernobjekts für die vom Autor geplante Nutzung: Hier ist zu prüfen, ob die Parameter der Nutzungsklassen „Lehre“ und „Lernen“ mit den Parametern des Lernobjekts vereinbar sind. So ist es beispielsweise nicht möglich, dynamische Lernobjekte, die Videos, Animationen, Klänge oder aktive Inhalte beinhalten, in einem vorlesungsbegleitenden Skript, das in PDF oder Postscript vorliegen soll, zu verwenden.

[34] Zur Erinnerung: Die Parameter der Nutzungsklasse „Präsentation“ legen fest, ob dynamische Medien erlaubt sind, ob die konkreten Lehrmaterialien linear oder als Hypertext genutzt werden, etc.

Dies kann über Constraints realisiert werden, die die geplante Nutzung mit dem Medientyp des Lernobjekts verknüpfen. Solche Constraints müssen vom Autor in Schritt 1 (vgl. 3.6.1) definiert werden, da er weiß, wie der fertige Kurs bzw. Kursteil verwendet werden soll.

- Die Anzahl der im Platzhalter nach Teilschritt 1, Punkt 1 gesetzten Parameter im Vergleich zur Anzahl der gesetzten Parameter des Lernobjekts: Je näher beide Zahlen beieinander liegen, desto besser ist ein Lernobjekt als Subelement geeignet.

Diese Kriterien bestimmen eine Rangfolge unter den möglichen Lernobjekten, die in den Platzhalter eingesetzt werden können.

Es ist jedoch möglich, dass in der festgelegten Rangfolge mehrere Lernobjekte gleich gut für die Nutzung in einem Platzhalter geeignet sind (mehrere Lernobjekte haben den höchsten Rang). Lernobjekte, für die dies zutrifft, sind Alternativen gemäß 3.4.2 und können daher als gleichwertig behandelt werden. Welches Lernobjekt nun konkret in einen Platzhalter eingesetzt werden soll, kann in diesem Fall programmatisch nicht mehr entschieden werden; die Auswahl muss der Autor des Kurses bzw. des Kursteils treffen.

Als Ergebnis dieses Teilschritts gibt es nun für jeden Platzhalter innerhalb der Menge fittingObjs von passenden Lernobjekten eine Rangfolge bezüglich der Eignung der Lernobjekte für die spätere Nutzung.

3.6.2.3 Teilschritt 3: Aufbau eines Geflechts von Lernobjekten

Als Ergebnis der Teilschritte 1 und 2 ist nun für jeden Platzhalter in und unterhalb der initialen Rahmenkomponente RK bekannt, welches Lernobjekt in den jeweiligen Platzhalter eingefügt werden soll: Es wird entweder das Lernobjekt aus fittingObjs mit dem höchsten Rang genommen oder, falls mehrere Lernobjekte aus fittingObjs gleichwertig den höchsten Rang haben, das vom Autor ausgewählte Lernobjekt. Zwischen jedem Platzhalter und dem jeweils einzusetzenden Lernobjekt wird dann eine Kompositionsbeziehung realisiert. In der Summe erhält man auf diese Weise einen Baum von Lernobjekten mit der Rahmenkomponente RK als Wurzel.

Assoziationen sind Beziehungen zwischen Lernobjekten, die vom Autor explizit definiert werden und parallel zum existierenden Lernobjektbaum existieren. Assoziationen zwischen Lernobjekten definieren nicht wie Kompositionen eine Struktur zwischen Lernobjekten, sondern modellieren Verweise zwischen Lernobjekten. Lernobjekte, die über Assoziationen mit anderen Lernobjekten verbunden sind, werden daher auch nicht in eine gemeinsame Rahmenkomponente, wie beispielsweise ein übergeordnetes Kapitel, integriert. Statt dessen wird eine Assoziation über ein eigenes Objekt realisiert, das die jeweiligen Lernobjekte in Beziehung setzt, vgl. hierzu 3.3.2 und 3.3.5.

Kompositionen und Assoziationen definieren gemeinsam ein Geflecht von Lernobjekten, das, ausgehend von der initialen Rahmenkomponente RK, den Kurs bzw. den Kursteil definiert. Dieses Geflecht repräsentiert die logische Struktur des Kurses bzw. Kursteils. Bis zu diesem Zeitpunkt ist man – abgesehen von der Wahl, ob dynamische Inhalte genutzt werden können oder nicht – noch frei in der Art und Weise, wie der Kurs den Lernern präsentiert werden soll. In den nächsten Schritten wird das Präsentationsformat des Kurses bzw. Kursteils fixiert und das Geflecht von Lernobjekten zu einem konkreten Kurs transformiert.

3.6.3 Schritt 3: Anpassung der Struktur an das Präsentationsformat

Nachdem die logische Struktur des Kurses bzw. Kursteils aufgebaut ist, muss diese den Nutzern präsentiert werden. Die hierfür nötigen Einstellungen (Hypertext oder Sequenz von Lernobjekten, etc.) sind nicht in den Lernobjekten vermerkt, sondern werden vom Autor des

Kurses in den Anwendungsprogrammen angegeben, die die Umsetzung des Lernobjektgeflechts in konkrete Lehrmaterialien durchführen.

Präsentationsformate von Lehrmaterialien und die Möglichkeiten zur Navigation in ihnen hängen eng zusammen: Bei Präsentationsformaten, die ein beliebiges Hin- und Herspringen zwischen verschiedenen Lernobjekten nicht erlauben, ist beispielsweise die Verwendung von Hyperlinks nicht zielführend. Umgekehrt nutzt eine rein sequentielle Navigation „von Kapitel zu Kapitel“ die Möglichkeiten einer Hypertextpräsentation bei weitem nicht aus.

Generell müssen zwei Arten von Präsentationsformaten unterschieden werden: Präsentationsformate, die durch Sequenzialisierung von Lernobjekten gebildet werden (beispielsweise Bücher oder druckbare Skripte), und Präsentationsformate, die Geflechte von Lernobjekten erlauben (beispielsweise Hypertexte).

3.6.3.1 Präsentation als Geflecht von Lernobjekten

Wird vom Autor festgelegt, dass der Kurs bzw. der Kursteil als Geflecht präsentiert werden soll, beispielsweise als Menge verlinkter HTML-Seiten, kann das Lernobjektgeflecht, das den Kurs bzw. Kursteil definiert, unmittelbar in das gewünschte Präsentationsformat abgebildet werden: Die Lernobjekte bleiben als eigenständige Präsentationseinheiten, beispielsweise als HTML-Seite, erhalten; eine Sequenzialisierung ist nicht nötig. Kompositionen und Assoziationen, die zwischen zwei Lernobjekten definiert sind, werden unmittelbar als Links übersetzt.

3.6.3.2 Präsentation als Sequenz von Lernobjekten

Die Präsentation eines Lernobjektgeflechts als Sequenz von Lernobjekten ist komplexer: Einerseits muss eine Sequenzialisierung des Lernobjektbaums (vgl. 3.6.2.3) vorgenommen werden, und andererseits müssen Assoziationen in alternative Verweisformen (Fußnoten, etc.) umgesetzt werden.

Sequenzialisierung des Lernobjektbaums

Maßgeblich für die Sequenzialisierung des Lernobjektbaums sind die Kompositionsbeziehungen zwischen einem Modul und seinen Subelementen. Assoziationen zwischen Lernobjekten dürfen hier nicht berücksichtigt werden. Die Sequenzialisierung des Lernobjektbaums mit der initialen Rahmenkomponente RK als Wurzel erfolgt mittels einer einfachen Tiefensuche entlang den Kompositionsbeziehungen, wobei die einzelnen Lernobjekte, die beim Durchlaufen des Baums angetroffen werden, aneinandergefügt werden.

Auflösung von Assoziationen

Assoziationen sind Beziehungen zwischen Lernobjekten, die parallel zum existierenden Lernobjektbaum existieren. Assoziationen zwischen Lernobjekten modellieren Verweise zwischen verwandten Lernobjekten.

Falls das Präsentationsformat keine Möglichkeiten unterstützt, von einem Lernobjekt direkt zu einem anderen Lernobjekt zu springen, wie das beispielsweise der Fall ist, wenn ein Buch erzeugt werden soll, können Assoziationen nicht wie in 3.6.3.1 durch Links aufgelöst werden. Statt dessen bieten sich Querverweise oder Fußnoten an.

Damit das Lernobjekt, auf das eine Assoziation weist, als Fußnote realisiert werden kann, müssen folgende Bedingungen erfüllt sein:

- Das Lernobjekt, auf das die Assoziation verweist, enthält ausschließlich Text.
- Das Lernobjekt, auf das die Assoziation verweist, ist ein Atom bzw. nicht Teil des Lernobjektbaums.

- Von dem Lernobjekt, auf das die Assoziation verweist, gehen keine weiteren Beziehungen – egal ob Assoziationen oder Kompositionen – aus.

Durch diese Bedingungen wird sichergestellt, dass Fußnoten auch wirklich nur dann eingesetzt werden, wenn das assoziierte Lernobjekt klein, übersichtlich und nur einem einzigen anderen Lernobjekt zugeordnet ist.

In allen anderen Fällen kommen Querverweise zum Zug. Eine Assoziation kann damit beispielsweise durch folgenden Einschub realisiert werden: „(vgl. auch Abschnitt x.y)“ Tabelle 3-8 liefert eine Auswahl möglicher Assoziationen.

Geplante Anwendung	Realisierung
Gedrucktes Skript (als PDF, Postscript, etc.)	Als Verweis auf Kapitelnummer, Seitenzahl, etc.
Präsentation für Vorlesung	Assoziationen nicht in die Lehrmaterialien aufnehmen, sondern lediglich vom Dozenten in der Vorlesung nennen

Tabelle 3-8: Mögliche Realisierung von Assoziationen

3.6.4 Schritt 4: Festlegung des Layouts mittels Style Sheets

Nachdem das Lernobjektgeflecht, wie in 3.6.3 beschrieben, in seiner Struktur dem endgültigen Präsentationsformat angepasst wurde, müssen den einzelnen Lernobjekten noch Eigenschaften zugewiesen werden, wie sie konkret im Zielmedium (Skript auf Papier, virtueller Hypertext, etc.) dargestellt werden sollen. Dies betrifft beispielsweise (Text-) Farbe, Schriftgröße, Auflösung des Zielmediums (Bildschirm, Drucker oder Satzbelichter), Dynamik des Zielmediums (Animationen und Videos können beispielsweise nur in virtuellen Skripten, beispielsweise im Hypertext-Format, verwendet werden), Layout, etc.

Hierzu werden den einzelnen Lernobjekten je nach ihrer Art (LOM.Presentation.Dataformat) Style Sheets zugewiesen, die alle visuellen Eigenschaften des Lernobjekts festlegen. All diese Einstellungen werden wieder vom Autor in den Anwendungsprogrammen vorgenommen, die die Umsetzung des Lernobjektgeflechts in konkrete Lehrmaterialien durchführen.

3.7 Kritik: Schwächen des Datenmodells

Ziel des in diesem Kapitel vorgestellten Datenmodells ist es, Lehrmaterialien auf möglichst einfache und flexible Weise modellieren zu können. Die hierzu verwendeten Lernobjekte sollen dabei auf unterschiedlichste Weise einsetzbar bleiben – sowohl was ihre Präsentation und Darstellung betrifft, als auch hinsichtlich der Nutzungsklassen, unter denen sie eingesetzt werden sollen.

Während sich die Modellierung der Struktur und die Spezifikation der visuellen Eigenschaften der Lernobjekte als weitgehend unproblematisch erweist und in diesen Problembereichen auf bewährte Verfahren (beispielsweise Style Sheets) zurückgegriffen werden kann, bleiben die Rahmenbedingungen des Einsatzes der Lernobjekte problematisch; die Nutzungsklassen sind schwer zu greifen. Es bietet sich an, die Parameter der Nutzungsklassen als Metadaten der einzelnen Lernobjekte zu modellieren und so einzelne Aspekte der Rahmenkomponenten zu beschreiben.

Dieses Vorgehen ist heute allgemein anerkannt und wird in vielen Bereichen praktiziert (vgl. [LOM], [SüFr99], [HoCa01], etc.). Allerdings bringt der Einsatz von Metadaten einige Probleme mit sich, die gerade im Umfeld von Lehrmaterialien unangenehme Auswirkungen ha-

ben: Das Vokabular der Metadaten muss festgelegt werden und es muss sichergestellt sein, dass ein gemeinsames Verständnis über Bedeutung und Einsatz der Metadaten bei allen an der Erstellung und Nutzung von Lehrmaterialien Beteiligten existiert. Dies kann beispielsweise über Ontologien oder, wie hier vorgeschlagen, mit Hilfe von (Inhalts-)Kategorien realisiert werden. Ein weiteres Problem betrifft die Werte von Metadaten: Metadaten enthalten konkrete, in ihrer Semantik eindeutig und klar definierte Werte. So ist beispielsweise der Wert des Parameters (des Metadatums) „Autor“ eindeutig festgelegt. Aber wie sieht dies beim Wert des Parameters „Schwierigkeitsgrad“ aus? Den Schwierigkeitsgrad eines Lernobjekts kann man genau genommen erst dann angeben, wenn das Lernobjekt konkret eingesetzt werden soll, denn erst dann sind Zielgruppe, Vorwissen, konkrete Situation in der Lehrveranstaltung, etc. klar. Es kann dann zwar ein konkreter Wert angegeben werden, aber dieser hängt von vielerlei Einflüssen, die teilweise nicht formal beschreibbar und damit auch nicht mittels Constraints abbildbar sind, ab. Beispielsweise bedeutet ein Schwierigkeitsgrad von beispielsweise „leicht“ für jeden Autor und für jeden Lerner etwas anderes; ein Lernobjekt, das Autor A für „leicht“ hält, wäre für Autor „B“ vielleicht „mittel schwer“. Kurz: Es ist nicht möglich, für diesen Parameter einen allgemein anerkannten Wert festzuschreiben.

Die Beobachtungen mit dem Parameter „Schwierigkeitsgrad“ lassen sich auch für andere Parameter machen, und insbesondere sind hierbei Parameter betroffen, die in den Nutzungsklassen Verwendung finden. Dies liegt an zweierlei: Metadaten können dann ihre gesamte Stärke ausspielen, wenn für den Aspekt, den sie beschreiben, eindeutige, klar definierte Werte angegeben werden können. Parameter, die nicht wirklich gut auf eine konkrete Skala abgebildet werden können oder deren Werte vom jeweiligen Betrachtungspunkt aus unterschiedlich sind, sind hier generell problematisch. Dieses Problem wird verschärft, wenn die Werte aus der subjektiven Erfahrung von Beteiligten belegt werden und nicht davon ausgegangen werden kann, dass ein allgemein anerkannter Wert gefunden werden kann. An dieser Stelle soll lediglich auf diese Problematik hingewiesen werden. In Kapitel 5 wird diese Problematik erneut aufgegriffen und es wird eine Lösung vorgeschlagen.

3.8 Zusammenfassung

In diesem Kapitel wurde ein Datenmodell für Lehrmaterialien vorgestellt. Alle Lehrmaterialien bestehen in diesem Datenmodell aus Lernobjekten und Beziehungen zwischen den Lernobjekten. Lernobjekte können nicht weiter unterteilbar („Atome“) oder aus mehreren Atomen oder wiederum Lernobjekten zusammengesetzt („Module“) sein. Mittels Kompositionen wird eine hierarchische Struktur zwischen den Lernobjekten aufgebaut, mittels Assoziationen werden Querbeziehungen zwischen den Lernobjekten realisiert. So entsteht ein Geflecht von Lernobjekten.

Eigenschaften der Lernobjekte – egal ob Module oder Atome – werden mittels Parametern realisiert. Einschränkungen bezüglich der Werte oder Abhängigkeiten zwischen Parametern werden mit Constraints beschrieben. Constraints werden dabei durch prädikatenlogische oder arithmetische Ausdrücke auf den Werten eines oder mehrerer Parameter ausgedrückt.

Anschließend wurden Nutzungsklassen eingeführt, um die verschiedenen Arten der Nutzung der Lernobjekte zu modellieren. Jede Nutzungsklasse umfasst dabei mehrere Parameter und Constraints, die eine ganz bestimmte Art der Nutzung und eine ganz bestimmte Sichtweise auf ein Lernobjekt beschreiben.

Damit waren alle Konzepte des Datenmodells eingeführt, so dass gezeigt werden konnte, wie diese Konzepte bei der konkreten Instanziierung von Kursen zusammenspielen. Je nach Art der beabsichtigten Präsentationsform kann es nötig sein, das Geflecht der Lernobjekte zu sequenzialisieren. Es wurde beschrieben, wann so eine Transformation nötig ist und wie sie

durchgeführt wird. Ebenso wurde gezeigt, wie Beziehungen – insbesondere Assoziationen – je nach gewünschter Präsentationsform aufgelöst werden. Die Formatierung der einzelnen Lernobjekte erfolgt am Ende mittels Style Sheets.

Abschließend wurden Schwächen des Datenmodells diskutiert. Hierbei wurde festgestellt, dass die Verwendung von Metadaten als Realisierung der Parameter zwar nötig ist, dass sie aber auch einige konzeptuelle Probleme mit sich bringt. Probleme treten in zwei Kategorien auf: Bei Parametern, deren Werte sich nicht in eine diskrete Wertemenge einpassen lassen, und bei Parametern, deren Werte von der subjektiven Einschätzung des Autoren des Lernobjekts abhängen.

Im folgenden Kapitel wird der Prozess betrachtet, in dem Lehrmaterialien auf der Grundlage des gerade vorgestellten Datenmodells erstellt werden.

4 Prozess der Lehrmaterialienerstellung und –nutzung

4.1 Überblick

In 2.4.3 und 2.5.3 wurden unter anderem auch Systeme zum Entwurf von Lehrmaterialien vorgestellt. Abgesehen von der technischen Umsetzung der Systeme sind die verwendeten Konzepte sowie das Vorgehen zur Erstellung der Lehrmaterialien recht ähnlich.

Jedes dieser Systeme implementiert mehrere Stufen, in denen die Lehrmaterialien erstellt werden. Im wesentlichen handelt es sich immer um Entwurf der Lehrmaterialien, dann deren konkrete Entwicklung und Integration, ihre Adaption, dann eine Transformation in eine bestimmte Präsentationsform und schließlich die Hinzunahme von Navigationsinformationen. Immer sind die Lehrmaterialien modular aufgebaut, wobei für die Umsetzung gern XML [XML] sowie eng verwandte Technologien wie XSLT [XSLT] und XPath/XPointer [XPATH] verwendet werden.

Auch bezüglich der abgedeckten Themenkreise sind sich die Systeme sehr ähnlich: Sie decken Inhalt, Struktur und Nutzung von Lehrmaterialien ab, sie berücksichtigen verschiedene Arten der Präsentation und der Navigation.

Übereinstimmung gibt es ebenfalls bei den Rollen, die an der Lehrmaterialerstellung beteiligt sind. Die Systeme unterscheiden fast immer zwischen Autoren auf der einen Seite und Lerner und / oder Dozenten als (End-)Nutzer der Lehrmaterialien auf der anderen Seite.

Andere, in 2.4.3 und 2.5.3 nicht betrachtete Systeme zur Erstellung von Lehrmaterialien nutzen ähnliche Konzepte und Mechanismen.

Trotz dieser vielen Gemeinsamkeiten hat jedes System seine Eigenheiten. Von diesen Eigenheiten soll nun abstrahiert werden, denn es sollen an dieser Stelle nicht Stärken und Schwächen bestimmter, vorgegebener Systeme untersucht werden, sondern es soll generell die Vorgehensweise, die die Grundlage einer großen Zahl derzeit implementierter Systeme bildet, kritisch betrachtet und geeignet optimiert werden.

In diesem Kapitel wird daher ein generischer Prozess zur Erstellung von Lehrmaterialien hergeleitet, der von den Vorgehensweisen, wie sie die jeweiligen Systeme vorschreiben, abstrahiert und so möglichst weit einsetzbar ist. Grundlage bilden hierbei die Vorgehensweisen der in Kapitel 2.4.3 vorgestellten Systeme.

Zuerst wird in 4.2 der generische Prozess im Überblick vorgestellt und die beteiligten Rollen werden eingeführt. Es werden der Daten- und der Kontrollfluss aufgezeigt und Möglichkeiten der Strukturierung des Prozesses genannt. In 4.3 werden dann die einzelnen Teilprozesse im Detail vorgestellt und analysiert. Die Beobachtungen, die bei der Analyse der einzelnen Teilprozesse gemacht wurden, werden anschließend in 4.4 zusammengefasst und konkretisiert, bevor in 4.5 und 4.6 Problemfelder und Verbesserungspotenziale identifiziert werden.

4.2 Ein generischer Prozess

4.2.1 Segmentierung des Prozesses

Der Lehrmaterialerstellungs- und -nutzungsprozess kombiniert viele Schritte in einer interdisziplinären Umgebung, wobei Spezialisten verschiedener Disziplinen sowie verschiedene IT-Plattformen, und -werkzeuge zum Einsatz kommen. Pädagogen, Mediendesigner und Autoren arbeiten zusammen mit Fachleuten der verschiedenen Fächer an dem Design, der Erstellung und der Nutzung von Lehrmaterialien.

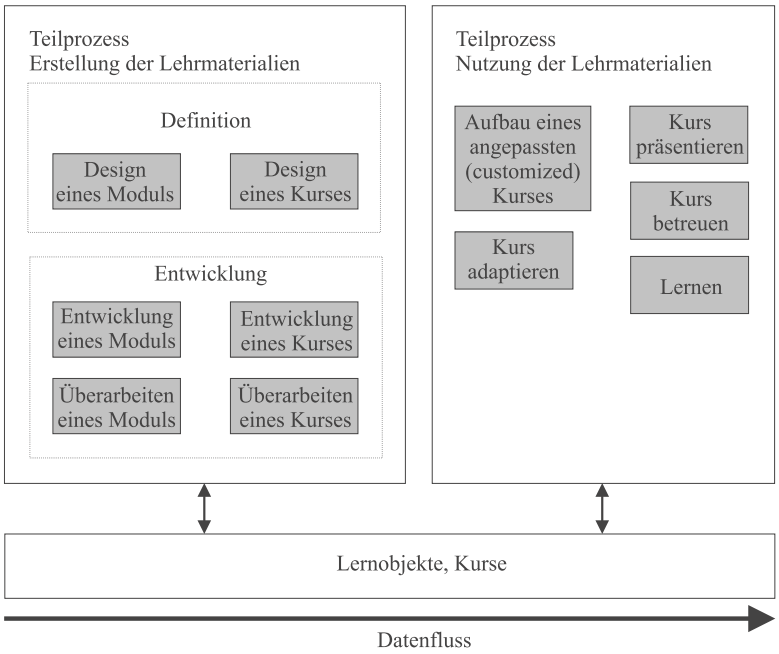


Abbildung 4-1: Prozessschritte und Datenfluss im Überblick

Angefangen mit dem ersten Entwurf der Module und Atome bis hin zum Aufbau eines nutzungsspezifischen Kurses werden viele Phasen der Erstellung und Nutzung der Lehrmaterialien durchgelaufen. In jeder Phase werden (Teile der) Lehrmaterialien definiert, entwickelt oder genutzt, wobei je nach Phase bestimmte Aspekte der Lehrmaterialien – Inhalt, Struktur, Präsentation und Navigation, sowie die beabsichtige Nutzung (vgl. auch 3.5.5.2) – berücksichtigt werden. Ausgehend von der Lehrmaterialerstellung und der Lehrmaterialnutzung sowie den Prozessteilnehmern, von denen die jeweiligen Aufgaben und die Interaktion zwischen den Prozessteilnehmern abhängen, wird der Prozess segmentiert und Schritt für Schritt modelliert. Dadurch wird die Komplexität des Prozesses reduziert; jeder Schritt in der Prozessausführung führt als Ergebnis zu einem Teilprodukt oder einem Modell des Teilprodukts (Lernobjekte, Kurse), welches auf das gemeinsame Datenmodell (vgl. Kapitel 3) abgebildet werden muss und über das Datenmodell den unmittelbar nachfolgenden Prozessschritten als Eingabe dient. In allen Schritten des Teilprozesses „Erstellung der Lehrmaterialien“ wird

zwischen Design und Entwicklung unterschieden: In den Designphasen werden die Typen der Lernobjekte definiert und die Spezifikationen der Instanzen für die Entwicklung festgelegt, in den Entwicklungsphasen werden die Instanzen dann konkret implementiert. In Abbildung 4-1 werden die einzelnen Aufgaben inklusive dem Datenfluss dargestellt.

Jede dieser Phasen und jeder der in Abbildung 4-1 genannten Prozessschritte wird in diesem Kapitel im Detail betrachtet und es werden Potenziale zur Erhöhung der Wiederverwendbarkeit von Lehrmaterialien, aber auch Probleme und Schwachstellen des Prozesses identifiziert.

4.2.2 Rollen und Aufgaben

Die in Abbildung 4-1 genannten Prozessschritte können einzelnen Prozessteilnehmern zugeordnet werden. Wie in Abbildung 4-1 dargestellt, unterteilt sich der gesamte Prozess in Erstellungsprozess und Nutzungsprozess. Analog lassen sich die Rollen der Prozessteilnehmer grob in Designer und Nutzer unterscheiden.

Zu den Designern gehören Rollen wie Atomdesigner, Moduldesigner und Kursdesigner, die für die Definition bzw. das Design und die Entwicklung bzw. Erstellung eines Atoms, eines abgeschlossenen Moduls bzw. eines Kurses verantwortlich sind.

Mögliche Aufgaben des Atomdesigners[35]:

- Erstellung von Atomen (Grafiken, Texte, Animationen, Videosequenzen, etc.)
- Initiale Bereitstellung der Atome für ihre spätere Nutzung in Modulen[36]
- Überarbeitung bereits existierender Atome hinsichtlich Inhalt und Parametern

Mögliche Aufgaben des Moduldesigners:

- Definition der Modulstruktur und Festlegen der Parameter, die das Modul beschreiben
- Erstellen bzw. Zusammenstellen des Modulinhalts (umfasst die Konfiguration sowie die Auswahl von bereits existierenden Atomen oder Modulen)
- Initiale Bereitstellung des Moduls für spätere Nutzung
- Überarbeitung bereits existierender Module hinsichtlich Inhalt, Struktur und Parameter

Mögliche Aufgaben des Kursdesigners:

- Definition der Kursstruktur und Festlegen der Parameter, die den Kurs beschreiben
- Auswahl und Integration von Modulen, die im Kurs verwendet werden sollen
- Initiale Bereitstellung des Kurses für spätere Nutzung in der Lehre und im Selbststudium
- Überarbeitung bereits existierender Kurse hinsichtlich Inhalt, Struktur und Parameter

Designer können einerseits parametrisierte Lernobjekte definieren und andererseits bei der Entwicklung ausgewählte Parameter mit passenden Werten belegen. Kursdesigner betrachten hierbei Kurse als White Box-Komponenten und Module sowie Atome als Black Box-Kom-

[35] Diese Rolle soll hier nur am Rande betrachtet werden, vgl. 4.2.3.
[36] Nutzung von Atomen in Modulen bedeutet die Einbettung von Atomen in sie umfassende Module.

ponenten, Moduldesigner entsprechend Module als White Box- und Atome als Black Box-Komponenten.

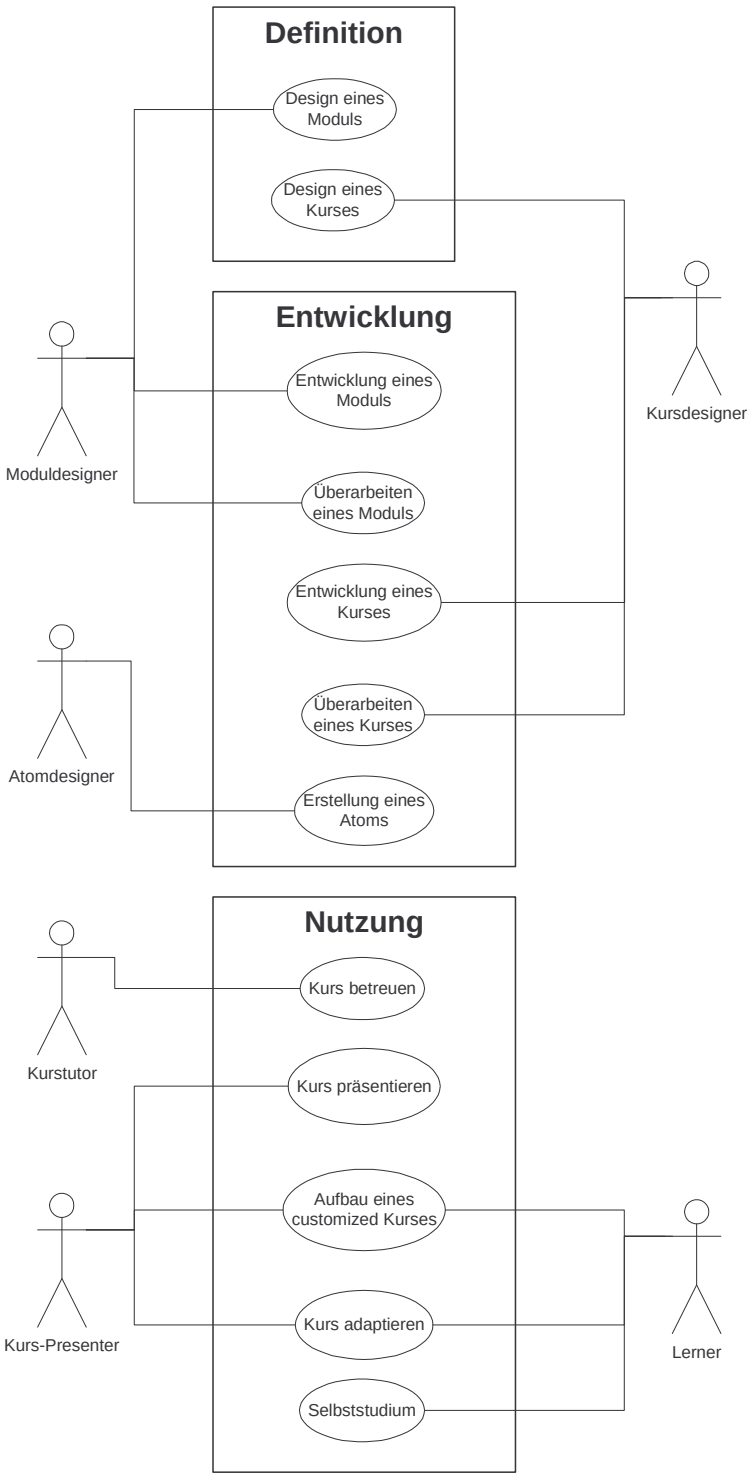


Abbildung 4-2: Use Cases im Lehrmaterialerstellungsprozess (vereinfacht)

Nutzer von Lehrmaterialien sind Kurs-Presenter, Kurstutor und Lerner, die bereits mehr oder weniger vorgefertigte konkrete Lehrmaterialien in verschiedenen Anwendungen nutzen,

ohne deren Inhalt oder Struktur zu verändern. Im Gegensatz zu Kurs- und Moduldesigner haben die Nutzer im allgemeinen nur eine Black Box-Sicht auf die Lehrmaterialien. Sie können die Lehrmaterialien anhand ihrer Außenschnittstellen (spezifizierte Instanzkomponente inklusive der sie beschreibenden Parameter) und adaptieren: Sie können Kursvarianten auswählen sowie – falls es sich um einen customized Kurs handelt – die Belegung bestimmter Parameter von Modulen bestimmen oder Alternativen von Modulen auswählen.

Mögliche Aufgaben des Kurs-Presenters:

- Vorbereiten (Adaptieren) des Kurses für die Präsentation durch Belegung bestimmter Parameter der Nutzungsklasse „Lehre“ und durch Festlegen eines passenden Layout sowie passenden Navigationspfaden
- Nutzung des Kurses in der Lehrveranstaltung
- Nachbearbeitung der Lehrveranstaltung

Mögliche Aufgaben des Kurstutors:

- Nutzung des Kurses während der Interaktion[37] mit den Lernern (beispielsweise in einer Tutorübung)[38]
- Nachbearbeitung der Tutorübung

Mögliche Aufgaben des Lerners:

- Vorbereiten (Adaptieren) des Kurses für das Lernen durch Belegung bestimmter Parameter der Nutzungsklasse „Lernen“ und Festlegen eines passenden Layout sowie passenden Navigationspfaden
- Im Falle eines customized Kurses Anpassen des Kurses entsprechend den vom Kursdesigner erlaubten Freiheitsgraden (in Form von Parametern, Constraints und / oder bereits vorgegebenen Modulen).
- Nutzung des Kurses im Selbststudium und zur Nachbereitung der Lehrveranstaltung

Abbildung 4-2 fasst die Aufgaben der einzelnen Rollen zusammen.

4.2.3 Überblick über den Prozess: Aktivitäten und Kontrollfluss

Bislang wurden die einzelnen Teilprozesse nur grob vorgestellt. Abbildung 4-3 zeigt den Prozess der Lehrmaterialerstellung im Überblick. Für eine Darstellung in UML siehe Anhang C.

Atomdesigner stellen die konkreten Inhalte von Lernobjekten bereit: Texte, Grafiken, Animationen, Videos, etc. Dies sind die elementaren Bausteine, aus denen Lehrmaterialien aufgebaut sind. Im folgenden wird davon ausgegangen, dass diese Bausteine (die Atome) bereits vorliegen und wiederverwendet werden können. Die Rolle des Atomdesigners spielt für die weiteren Betrachtungen also keine große Rolle und wird daher im folgenden nicht weiter beleuchtet. Gegebenenfalls kann man sich die Rolle des Atomdesigners auch zusammenfallend mit der Rolle des Moduldesigners vorstellen.

---

[37] Der Kurstutor steht während seiner Betreuung mit den Lernern wesentlich öfter in Interaktion als der Kurs-Presenter. Interaktion mit den Lernern ist beim Kurstutor tatsächlich der wesentliche Faktor für den Lernerfolg der Lerner.

[38] In meisten Fällen verwendet der Kurstutor direkt den vom Kurs-Presenter vorbereiteten Kurs; er kann allerdings auch analog zum Kurs-Presenter den Kurs selbst adaptieren.

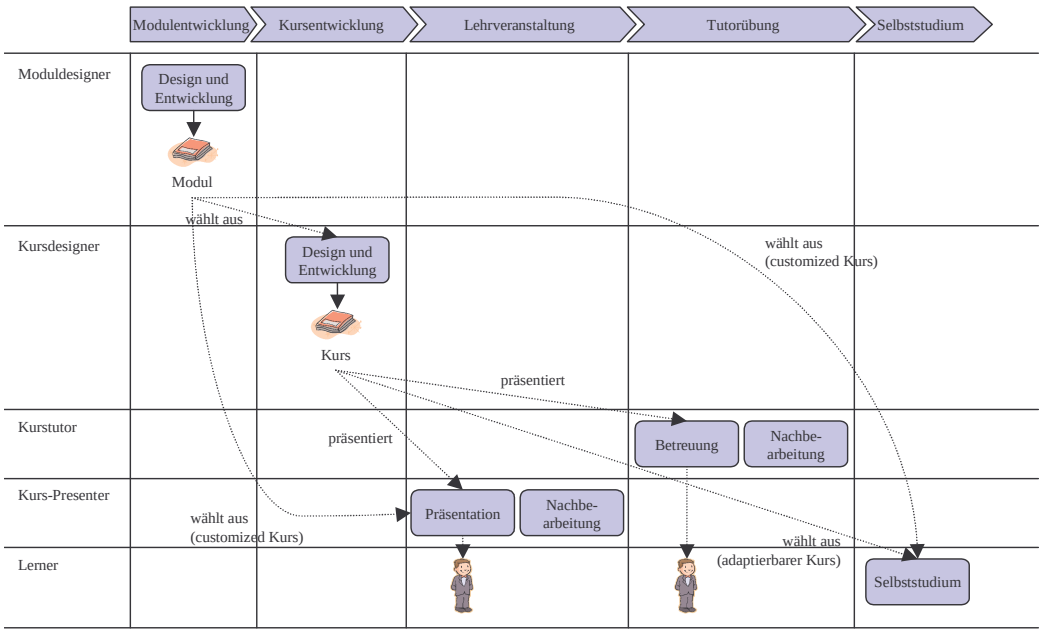


Abbildung 4-3: Überblick über den Prozess zur Lehrmaterialerstellung[39]

4.2.3.1 Prozesssegmentierung: Nutzungsklassen als Strukturierungselemente

In Abschnitt 3.5.2 wurden Nutzungsklassen bereits unter dem Gesichtspunkt der Strukturierung des Datenmodells und der Erhöhung der Wiederverwendbarkeit (bzgl. der einheitlichen Parametrisierung und Spezialisierung) betrachtet. Analog kann der Prozess der Lehrmaterialienerstellung durch Nutzungsklassen strukturiert werden.

Nutzungsklassen beschreiben, wie in 3.5.2 ausgeführt, Arten der Nutzung. Hierzu sind in den Nutzungsklassen die Parameter beschrieben, die für die jeweiligen Nutzungsszenarien maßgeblich sind. Nutzungsszenarien umfassen aber nicht nur Parameter, sondern auch Tätigkeiten. Es ist daher sinnvoll, neben den Nutzungsklassen nicht nur die Eigenschaften von Nutzungsszenarien mittels Parameter und Constraints zu modellieren, sondern auch die Tätigkeiten zu betrachten, die in den jeweiligen Nutzungsszenarien ausgeführt werden. Durch diese Verknüpfung von Nutzungsklassen mit Tätigkeiten können dann Nutzungsklassen verwendet werden, um den Prozess der Lehrmaterialerstellung zu strukturieren. Das Strukturierungskriterium sind dann die Tätigkeiten, die mit einer bestimmten Art der Nutzung – wie in der jeweiligen Nutzungsklasse modelliert – einhergehen.

Jede Rolle nutzt, entwirft und bearbeitet Lehrmaterialien auf eine ganz bestimmte Weise. Moduldesigner beispielsweise definieren die Eigenschaften von Modulen und füllen sie mit Atomen oder wiederum Modulen, Kursdesigner wählen aus bestehenden Modulen aus und bauen mit ihnen Kurse auf, etc. Jede Rolle hat somit eine ganz bestimmte Sicht auf die Lehrmaterialien und definiert auf diese Weise ihre individuelle Nutzungsklasse.

4.2.3.2 Eingangssichten und Ausgangssichten

Jeder Teilprozess benötigt für seine Durchführung bestimmte Daten oder Lernobjekte (Atome, Module und Kurse inklusive aller sie beschreibenden Parameter und Constraints) als Eingabe. Ebenso werden während der Durchführung jedes Teilprozesses Daten oder Lernob-

[39] Da die Rolle Atomdesigner in dieser Arbeit nur am Rande betrachtet wird, wird sie auf dieser Abbildung nicht berücksichtigt.

jekte oder deren Beschreibungen (mittels Parametern oder Constraints) erstellt oder verändert[40]. Die Daten und Lernobjekte, die für die Durchführung eines Teilprozesses benötigt werden, sowie deren Beschreibungen werden *Eingangssicht* genannt; die, die Ergebnis eines Teilprozesses sind oder die während der Durchführung eines Teilprozesses verändert werden, werden als *Ausgangssicht* bezeichnet. Die Daten und Lernobjekte, die während der Durchführung eines Teilprozesses betrachtet und genutzt werden, werden generell *Sicht* auf den Teilprozess genannt; die Sicht auf den Teilprozess umfasst damit die Eingangs- und die Ausgangssicht. Jede Rolle, die einen Teilprozess ausführt, arbeitet so letztlich auf einer Sicht, nämlich auf der Sicht, die durch den jeweiligen Teilprozess definiert ist (vgl. Abbildung 4-4).

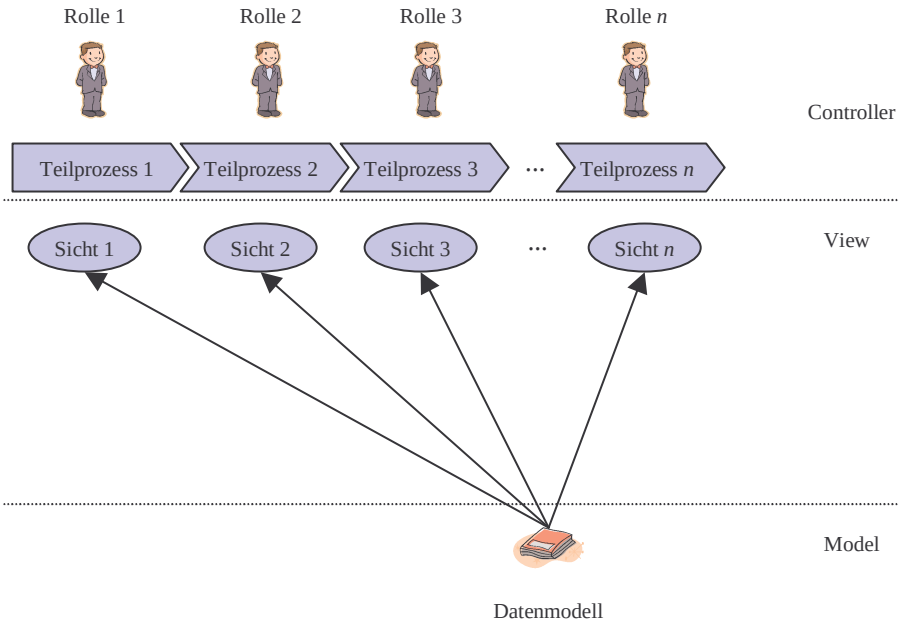


Abbildung 4-4: Teilprozesse, Sichten und Rollen

Eine Sicht auf einen Teilprozess kann als ein spezifisch angepasstes, abstraktes Teilmodell der Lehrmaterialien betrachtet werden, die in dem Teilprozess genutzt, verarbeitet und erzeugt werden. Sichten erlauben es, dass Prozessteilnehmer nur den Ausschnitt der Lehrmaterialien sehen, der für ihren Teilprozess von Bedeutung ist. So wird einerseits die Komplexität für die Prozessteilnehmer deutlich reduziert, indem die Darstellung der Lehrmaterialien mittels einer Filterung durch die Sicht ideal auf den Teilprozess abgestimmt werden kann. Andererseits wird eine Tool-Unterstützung deutlich vereinfacht, da die eingesetzten Tools nicht mehr mit dem gesamten Datenmodell umgehen können müssen, sondern sie auf die Unterstützung eines bestimmten Teilprozesses und damit einer bestimmten Sicht hin entwickelt werden können.

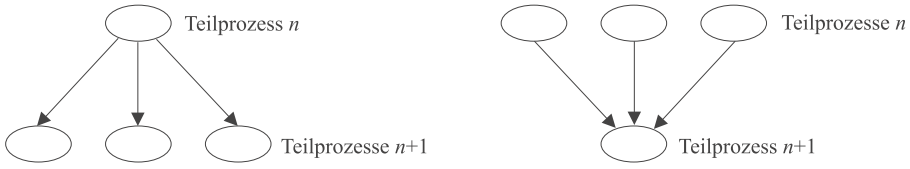
Es können mehrere unterschiedliche Sichten auf die gleichen Lehrmaterialien identifiziert werden: Jede Rolle hat für jeden Teilprozess, den sie durchführt, seine eigene Sicht. Dies realisiert im wesentlichen das Model-View-Controller-Paradigma: Der Datenbestand – das

[40] Daneben werden für die Durchführung des Teilprozesses möglicherweise auch noch temporäre Daten angelegt, verändert und wieder gelöscht. Da diese – wie der Name schon sagt – jedoch nur temporär innerhalb eines Teilprozesses existieren und von anderen Teilprozessen aus nicht sichtbar sind, sollen sie hier vernachlässigt werden.

Model – sind die Lehrmaterialien, die Views entsprechen den Sichten in den einzelnen Teilprozessen auf diese Lehrmaterialien, und der Controller wird durch die Prozessteilnehmer gebildet, die über den Sichten auf den Lehrmaterialien arbeiten und diese so modifizieren.

4.2.3.3 Sichten als Synchronisationspunkte zwischen voneinander abhängigen Teilprozessen

Die Eingangs- und Ausgangssichten definieren Schnittstellen zwischen voneinander abhängigen Teilprozessen. Die Ausgangssicht eines Teilprozesses *n* hat eine nicht-leere Schnittmenge mit den Eingangssichten aller Teilprozesse *n*+1, die unmittelbar auf Teilprozess *n* folgen (vgl. Fall 1 in Abbildung 4-5), andernfalls wären die beiden Teilprozesse nicht voneinander abhängig und könnten parallel ausgeführt werden. Umgekehrt sind auch die Ausgangssichten aller Teilprozesse *n* die Eingangssicht für den Teilprozess *n*+1, in den alle Teilprozesse *n* unmittelbar münden. In den Teilprozessen *n* werden also Daten und Lernobjekte erzeugt oder verändert, die in Teilprozess *n*+1 weiterverarbeitet werden (Fall 2 in Abbildung 4-5).



Fall 1: Ein Teilprozess *n*, auf den mehrere Teilprozesse *n*+1 folgen

Fall 2: Mehrere Teilprozesse *n*, auf die ein Teilprozess *n*+1 folgt

Abbildung 4-5: Ein- und Ausgangssichten von Teilprozessen

Die Schnittstellen zwischen den Teilprozessen *n* und *n*+1 definieren auf diese Weise Synchronisationspunkte zwischen diesen Teilprozessen. An diesen Stellen im Lehrmaterialerstellungs- und –nutzungsprozess ist eine Abstimmung zwischen den Prozesspartnern nötig, damit einerseits die jeweiligen Eingangs- und Ausgangssichten zueinander passen, und damit andererseits die für die Teilprozesse *n*+1 nötigen Daten und Informationen rechtzeitig von den Teilprozessen *n* geliefert werden. Die Prozesspartner treten dann bzgl. der Daten und Informationen der Eingangs- und Ausgangssichten miteinander in Kommunikation, um sich abzustimmen. Dies wird in 5.3.2.2 noch näher betrachtet.

4.3 Analyse der Teilprozesse

4.3.1 Design, Entwicklung und Überarbeitung eines Moduls

4.3.1.1 Überblick

Das Design und die Entwicklung von neuen Modulen sowie die Überarbeitung von Modulen wird vom Moduldesigner durchgeführt. Ziel dieser Teilprozesse ist es, ein neues Modul zu einem bestimmten Thema zu erstellen (umfasst sowohl Entwurf als auch Entwicklung) oder die Anpassung eines bereits existierenden Moduls anhand bestimmter Anforderungen (Anforderungen von Kursdesignern, anderen Moduldesignern, Lernern, etc.) vorzunehmen. Abbildung 4-6 illustriert diese Aufgaben.

Im Teilprozess „Design von Modulen“ werden statt der Entwicklung konkreter Lehrmaterialien lediglich Rahmen sowie Einschränkungen (Constraints) von Modulen mit inhaltlichen und strukturellen Aspekten festgelegt (vgl. 3.2.1.2). Ergebnis des Designs von Lehrmaterialien sind also Rahmenkomponenten.

Dagegen ist das Ergebnis der Entwicklung und Überarbeitung von Lehrmaterialien ein konkretes, inhaltlich und strukturell abgeschlossenes Modul, das eigenständig zu pflegen ist und

zugleich mittels Komposition und Assoziation mit anderen konkreten Modulen zusammen verwendet werden kann. Dabei werden die Konkretisierung und das Suchen bereits bestehender Module bzw. das Entwickeln neuer Lernobjekte bei der Entwicklung bzw. Überarbeitung durchgeführt. Ferner lässt sich die Überarbeitung eines Moduls in Verändern des existierenden Inhalts, der existierenden Struktur bzw. der existierenden Parameterbelegung unterteilen. Ergebnis dieser Teilprozesse sind Instanzkomponenten.

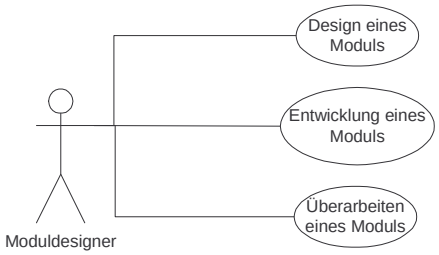


Abbildung 4-6: Aufgaben des Moduldesigners

Die Trennung der beiden Teilprozesse „Design eines Moduls“ und „Entwicklung eines Moduls“ ermöglicht es, mehrere konkrete Module zu ein und derselben Modulbeschreibung anzufertigen. So können zu einem Modul mit demselben Inhalt mehrere Alternativen erstellt werden: Da die Außenschnittstellen (repräsentiert durch die Parameter) der jeweiligen konkreten Module gleich sind, können die konkreten Module einfach gegeneinander ausgetauscht werden.

4.3.1.2 Teilprozess: Design eines Moduls

Der Moduldesigner definiert den Rahmen eines Moduls. Hierzu werden die Parameter und Constraints des Moduls festgelegt (vgl. hierzu auch 3.6.1) und dessen Subelemente (Submodule und Atome) spezifiziert.

Beteiligte Rollen

- Moduldesigner (Autor des aktuell betrachteten Moduls)
- Evtl. weitere Moduldesigner

Eingangsdaten

- Parameter und Constraints, die das Thema des Moduls festlegen.

Diese Parameter und Constraints liegen im allgemeinen nicht in der Form „Attributname – Attributwert“ oder als logischer Ausdruck vor, sondern meistens als natürlichsprachliche Beschreibung von Eigenschaften, die das fertige Modul haben soll. Diese Beschreibungen können beispielsweise von Kursdesignern kommen, die für ihre Kurse Module mit ganz bestimmtem Inhalt und ganz bestimmten Eigenschaften benötigen. Es ist nun die Aufgabe des Moduldesigners, diese Beschreibungen in eine korrekte Menge von Parametern und Constraints umzusetzen.

Die im zu definierenden Modul enthaltenen Atome oder Submodule müssen nicht konkret vorliegen. Es genügt, dass die Platzhalter, an denen später die konkreten Module und Atome eingesetzt werden sollen, korrekt parametrisiert sind, so dass in einem späteren Schritt (vgl. 4.3.1.3) eine Zuordnung „Platzhalter – konkretes Lernobjekt“ möglich ist.

Ergebnis

- Modulrahmen, dessen Platzhalter (Struktur) korrekt parametrisiert sind und der selbst mittels Parametern korrekt beschrieben ist. Die Parameter werden – falls nötig – durch Constraints eingeschränkt .

Detaillierte Beschreibung der Aktivitäten

Der Moduldesigner definiert die Struktur des Moduls. Das Modul besteht dann, wie in 3.4.2 und 3.6.1 beschrieben, aus mit Parametern beschriebenen Platzhaltern, in die konkrete Module bzw. Atome eingesetzt werden können, sowie aus Parametern, die das Modul selbst beschreiben. Die einzelnen Aktivitäten des Moduldesigner sind in Abbildung 4-7 dargestellt[41].

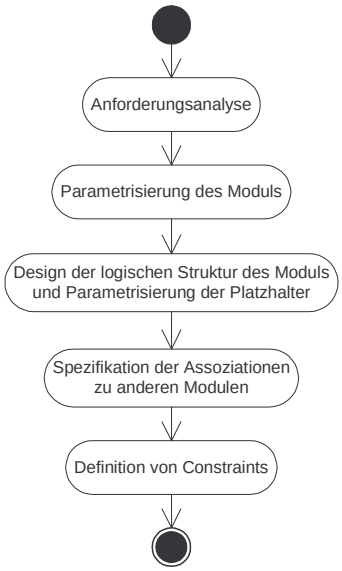


Abbildung 4-7: Design eines Moduls

Aktivität: Anforderungsanalyse (Analyse und Einschränkung des Themas)

Der Moduldesigner arbeitet auf Basis eines vorgegebenen Themas sowie bestimmter Randbedingungen[42] (Detaillierungsgrad des Themas, etc.), die oft erst natürlichsprachlich formuliert und später in der Aktivität „Parametrisierung des Moduls“ dann formal durch Constraints ausgedrückt werden. Bei der Anforderungsanalyse setzt sich der Moduldesigner mit dem Thema auseinander und prüft, welchen Anforderungen unter den gegebenen Randbedingungen ein Modul zu dem geforderten Thema genügen muss.

Das Ergebnis dieser Aktivität ist die thematische und strukturelle Grobvorstellung des aufzubauenden Moduls. Wichtig ist hierbei, dass der Moduldesigner gedanklich die Strukturierung dieses Moduls einer vorliegenden Kategorisierung der Module (Inhaltskategorien, Strukturkategorien, etc., vgl. 3.2.1.3) zuordnet.

[41] Alle diese Schritte werden nicht immer notwendigerweise streng in dieser Reihenfolge ausgeführt. Analog werden auch manchmal nicht alle diese Schritte ausgeführt. Beispielsweise wird der Schritt „Design der logischen Struktur des Moduls und Parametrisierung des Inhalts“ bei einem einfachen Modul (ohne Submodule) nicht durchgeführt.

[42] Randbedingungen kann man sich intuitiv als eine informelle, meist natürlichsprachige Vorstufe zu Constraints, die ja stark formalisiert sind, denken.

Aktivität: Parametrisierung des Moduls

Aus den Ergebnissen der Anforderungsanalyse leitet der Moduldesigner die Parameter, mit denen das Modul beschrieben wird, ab. So legt der Moduldesigner die Außenschnittstelle des Moduls (Parameter und Constraints, die das Modul beschreiben) fest. Grundlage hierbei sind die in 3.4.2.1 beschriebenen (allgemeinen und inhaltlichen) Parameter.

Das Ergebnis dieser Aktivität ist eine formale Beschreibung des Moduls mittels Parametern.

Aktivität: Design der logischen Struktur des Moduls (Definition und Parametrisierung der Platzhalter) und Parametrisierung des Inhalts (nur bei komplexen, zusammengesetzten Modulen)

Beim Design eines zusammengesetzten Moduls definiert der Moduldesigner anhand der Ergebnisse aus der Anforderungsanalyse den strukturellen Aufbau des (zusammengesetzten) Moduls, also die Aufteilung des Themas in Submodule und Atome. Hierzu wird eine Folge von Platzhaltern definiert, in die die passenden Submodule und Atome eingesetzt werden. Diese Aufteilung wird rekursiv für alle Submodule wiederholt, sofern nicht bereits geeignete Submodule (konkrete Module oder Rahmenkomponenten) in einem Repository fertig nutzbar vorliegen. Hierbei werden allerdings keine konkreten Submodule vorgegeben, sondern lediglich deren Eigenschaften mittels Parametern festgelegt, die die an der jeweiligen Stelle einzusetzenden Module oder Atome erfüllen müssen (beispielsweise der Schwierigkeitsgrad eines Moduls). Statt konkreter Module oder Atome werden somit parametrisierte Platzhalter definiert.

Bei der Parametrisierung der Submodule und Atome des Moduls muss zwischen verpflichtend vorgeschriebenen, bedingten und optionalen Subelementen unterschieden werden:

- Für die Parameter von verpflichtend vorgeschriebenen Subelementen sind die Parameter, die in der Außenschnittstelle des Moduls definiert wurden, maßgeblich: Jeder Parameter, der die Außenschnittstelle beschreibt und der bei der Beschreibung der Eigenschaften von Subelementen (Atome und Module) in Platzhaltern wieder auftaucht, wird entsprechend des Werts der Außenschnittstelle belegt (vererbt). Dies betrifft beispielsweise den Parameter „Schwierigkeitsgrad“: Hat der Moduldesigner festgelegt, dass das Modul den Schwierigkeitsgrad „niedrig“ haben soll, so muss dies auch für alle Subelemente des Moduls gelten. Dies ist im Detail in 3.6.2 beschrieben.
- Bei optionalen Subelementen muss dies nicht so streng gesehen werden: Die Parameter, die optionale Subelemente beschreiben, können beliebig belegt oder auch ignoriert werden. Dies ermöglicht es, beliebige Subelemente in ein Modul aufzunehmen, wodurch didaktische Einheiten wie Analogien aus anderen Wissensbereichen oder Ausblicke möglich werden.

Die Schritte der Aktivität „Design der logischen Struktur eines Moduls und Parametrisierung des Inhalts“ werden in der Praxis üblicherweise nicht streng sequenziell, sondern miteinander verschränkt ausgeführt. Dies ist unmittelbar einsichtig, da sich der Moduldesigner bei der Aufteilung komplexer Module in einfachere Submodule zugleich auch Gedanken machen muss, welche Eigenschaften die Submodule haben müssen, damit sie zusammen ein in sich stimmiges Bild ergeben.

Das Ergebnis dieser Aktivität ist eine Hierarchie von Modulen und Atomen, die allerdings keine konkreten Inhalte haben, sondern Platzhalter beinhalten. Die Tiefe der Modulhierarchie hat unmittelbar Einfluss auf den Wiederverwendungsgrad (vgl. 2.2.3, Granularität von Lehrmaterialien): Je kleiner und unspezifischer die Submodule und Atome sind, je flacher also ihre Hierarchie ist, desto weniger speziell sind sie und für umso mehr Lehrveranstaltungen und Lehrmaterialien können sie wiederverwendet werden und umgekehrt.

Aktivität: Spezifikation von Assoziationen zu anderen Modulen

Nach der Spezifikation und Parametrisierung der Teilmodule und Atome des zu entwerfenden Moduls und ihrer strukturellen Beziehungen untereinander, können Assoziationen (Referenzen) zu anderen Modulen und Atomen hergestellt werden (vgl. 3.3.1.4 und 3.3.1.5). Assoziationen betreffen nicht die Struktur eines Moduls, sondern repräsentieren beliebige andere (beispielsweise inhaltliche) Beziehungen zwischen Lernobjekten. Sie müssen vom Moduldesigner explizit definiert werden.

Da noch nicht bekannt ist, mit welchen Modulen oder Atomen die einzelnen Platzhalter gefüllt werden, können Assoziationen nur auf der Grundlage der Parameter, mit denen jeder einzelne Platzhalter versehen ist, definiert werden. Die Assoziationen beziehen sich dann auf den Inhalt des Moduls generell und sind nicht von einem bestimmten Modul oder Atom, das in einen Platzhalter eingesetzt werden kann, abhängig. Diese Art der Assoziation schränkt die Austauschbarkeit von Submodulen oder Atomen eines Moduls und damit dessen Wiederverwendbarkeit nicht ein, da bezüglich der Darstellung der Inhalte in den Submodulen und Atomen keine Annahmen gemacht werden.

Das Ergebnis dieser Aktivität ist eine Menge von Assoziationen zu anderen Modulen, zu denen eine (beispielsweise thematische) Beziehung besteht. Das bislang isoliert existierende zu entwickelnde Modul ist nun thematisch zu anderen Modulen in Zusammenhang gesetzt.

Diese Aktivität kann verschränkt mit der Aktivität „Design der logischen Struktur eines Moduls und Parametrisierung des Inhalts“ ablaufen.

Aktivität: Definition von Constraints

Die in den bisherigen Schritten aufgebaute Modulhierarchie wird in diesem Schritt mittels Constraints präzisiert: Der Moduldesigner schränkt auf Grundlage der Ergebnisse der Anforderungsanalyse gezielt die Wertebereiche von Parametern ein, um die Submodule oder Atome, die in Platzhalter eingesetzt werden können, möglichst präzise zu beschreiben. Des weiteren definiert er die Constraints auf den Parametern von Modulen, Komposition oder Assoziationen.

Ergebnis dieser Aktivität ist eine Hierarchie von Modulen und Atomen, die nun deutlich präziser beschreibt, welche Module bzw. Atome in einen Platzhalter eingesetzt werden dürfen.

Zu beachten ist, dass auch diese Aktivität verschränkt mit den Aktivitäten „Design der logischen Struktur eines Moduls und Parametrisierung des Inhalts“ und „Design von Assoziationen zu anderen Modulen“ ablaufen kann.

Beobachtung

- Bei der Parametrisierung des Inhalts des Moduls werden keine Annahmen über die Existenz bestimmter Module oder Atome gemacht; die Parametrisierung geschieht völlig isoliert von den übrigen Modulen. Dies ermöglicht zwar einerseits eine belie-

bige Austauschbarkeit der Submodule und Atome, da sie lediglich über ihre Eigenschaften spezifiziert werden, aber andererseits ist das Risiko vorhanden, dass es für einen Platzhalter keine passenden Module oder Atome gibt.

- Des weiteren ist ein gemeinsames Verständnis der Semantik der Parameter unbedingt erforderlich, da nur so die Bestückung der Platzhalter mit passenden Submodulen oder Atomen möglich ist. Dies ist aufgrund der Festlegung einer Menge von Parametern und einem gemeinsamen Verständnis über die Semantik der Parameter im Datenmodell (vgl. Kapitel 3) gewährleistet.
- Das Design eines Moduls ist mit einer gewissen „Blindheit“ gegenüber der geplanten Nutzung in der Lehre oder dem Lernen möglich, da zum Zeitpunkt des Designs eines Moduls nicht bekannt ist, in welchen Kursen das Modul eingesetzt werden soll und welchen didaktischen Rahmenbedingungen der Kurs – und damit auch das Modul – genügen muss. Beim Design eines Moduls überwiegen Aspekte der Wiederverwendung.

4.3.1.3 Teilprozess: Entwickeln eines Moduls

Der Moduldesigner füllt das in 4.3.1.2 spezifizierte Modul der Rahmenebene mit Inhalten. Er sorgt für die Verfügbarkeit der Atome und Submodule, die dort als Teile des Moduls angegeben wurden. Dies kann darin bestehen, dass er die benötigten Atome und Module selbst erstellt (und damit kurzzeitig die Rolle des Atomdesigners einnimmt) oder sie von Atomdesignern oder anderen Moduldesignern bereitstellen lässt. Dieser Teilprozess kann erst dann abgeschlossen werden, wenn alle benötigten Subelemente (Atome oder Module) als Instanzkomponenten, also mit konkretem Inhalt, bereitstehen.

Des weiteren werden auch in diesem Schritt Assoziationen zu anderen Modulen parametrisiert.

Beteiligte Rollen

- Moduldesigner
- Evtl. Atomdesigner und weitere Moduldesigner

Eingangsdaten

- Der Modulrahmen, der in 4.3.1.2 erstellt wurde.

Ergebnis

- Konkretes Modul, das Inhalt enthält und korrekt parametrisiert ist. Dieses Modul kann für die Erstellung von Kursen oder weiteren Modulen (wieder-)verwendet werden.

Detaillierte Beschreibung der Aktivitäten

In diesem Teilprozess fügt der Moduldesigner dem Modulrahmen, der im letzten Schritt erstellt wurde, konkrete Inhalte hinzu. Dies bedeutet im wesentlichen, dass der Moduldesigner dafür sorgt, dass die Submodule und Atome zur Verfügung stehen, aus denen das Modul aufgebaut ist. Anschließend werden, falls gewünscht, Querbeziehungen zu anderen Modulen (Assoziationen) eingefügt. Abbildung 4-8 zeigt den Teilprozess im Überblick.

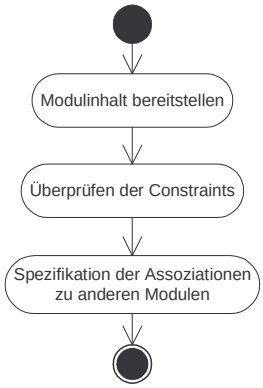


Abbildung 4-8: Entwickeln eines Moduls

Aktivität: Modulinhalt bereitstellen

Der Moduldesigner sorgt dafür, dass alle Submodule und Atome, die im aktuellen Modul enthalten sind, bereitstehen. Sind diese bereits verfügbar, so ist nichts zu tun. Sind hingegen bestimmte Submodule oder Atome nicht verfügbar, so kann der Moduldesigner auf zweierlei Arten für deren Bereitstellung sorgen:

- Der Moduldesigner erzeugt die noch nicht existierenden Subelemente (Module und Atome) selbst. In diesem Fall erstellt der Atom- bzw. Moduldesigner die Atome bzw. durchläuft für jedes neu zu entwickelnde Submodul rekursiv die Teilprozesse „Design eines Moduls“ (vgl. 4.3.1.2) und „Entwickeln eines neuen Moduls“ (vgl. 4.3.1.3). Die Parameter der Platzhalter, an denen ein Modul bzw. Atom eingesetzt werden soll, geben dabei die Außenschnittstelle des Moduls bzw. des Atoms vor.
- Der Moduldesigner stellt eine Anfrage an andere Atom- und Moduldesigner, dass diese die noch fehlenden Subelemente (Module und Atome) erstellen. In diesem Fall muss der Moduldesigner die Eigenschaften der zu erzeugenden Subelemente genau angeben: Im zu erstellenden Modul sind die Eigenschaften über Parameter und Constraints definiert, die die Atome und Submodule jeweils erfüllen müssen. Diese Eigenschaften definieren die geforderte Außenschnittstelle der zu erstellenden Subelemente. Diese Eigenschaften muss der Moduldesigner den anderen Atom- bzw. Moduldesignern (über entsprechende Werkzeuge wie Anforderungsverwaltungssysteme oder direkt in Kommunikation mit den anderen Atom- bzw. Moduldesignern) mitteilen, denn nur Subelemente, die diese Eigenschaften erfüllen, können später in das Modul eingesetzt werden.

Generell ist hierbei zu beachten, dass die Subelemente – Module und Atome – genau zu den Platzhaltern des Moduls passen, d.h. dass die Außenschnittstellen der Module und Atome den Parametern des in 4.3.1.2 erstellten Moduls entsprechen.

Dieser Vorgang wird rekursiv so lange wiederholt, bis alle für das zu erstellende Modul nötigen Subelemente verfügbar sind.

Ergebnis dieser Aktivität ist eine Menge von Modulen und Atomen, die das zu entwickelnde Modul als Subelemente beinhaltet.

Aktivität: Überprüfen der Constraints

Stehen alle Subelemente (Atome und Module) für ein Modul bereit, muss der Moduldesigner überprüfen, ob für die bereitgestellten Subelemente alle Constraints, die auf den Platzhaltern des Moduls definiert sind, erfüllt sind. Es findet hier ein Abgleich der Para-

meter des Moduls mit den Parametern der ins Modul einzusetzenden Submodule und Atome statt. Dieser Schritt ist notwendig, um die Stimmigkeit des Moduls sicherzustellen. Ist beispielsweise für ein Modul gefordert, dass der Schwierigkeitsgrad „niedrig“ sein soll, so darf es kein zwingend vom Lerner zu absolvierendes Submodul oder Atom geben, dass den Schwierigkeitsgrad „hoch“ hat, vgl. auch die Ausführungen in 3.6.2. Dies kann durch geeignete Tools oder Autorensysteme unterstützt werden.

Diese Aktivität hat kein explizit aufführbares Ergebnis, sondern dient der Sicherstellung der Korrektheit.

Diese Aktivität kann mit der Aktivität „Modulinhalt bereitstellen“ parallel ablaufen.

Aktivität: Spezifikation der Assoziationen zu anderen Modulen

Nachdem die Inhalte des Moduls bereitgestellt sind, können Assoziationen zu anderen Modulen hergestellt werden. Dies wurde teilweise schon im Teilprozess „Design eines Moduls“ implizit erledigt, beispielsweise wenn auf Vorwissen referenziert werden soll. Die Assoziationen bezogen sich in diesem Fall auf den Inhalt des Moduls generell.

Es kann jedoch auch sinnvoll sein, Assoziationen zu anderen Modulen herzustellen, wobei die Assoziationen durch die Art, wie der Inhalt in dem jeweiligen Modul dargestellt wird, begründet ist. Diese Assoziationen beziehen sich dann auf ein konkretes Modul. So kann beispielsweise zur Erklärung eines Sachverhalts in einem Modul ganz bestimmtes Vorwissen nötig sein, das dann mittels einer Assoziation referenziert wird.

Das Ergebnis dieser Aktivität ist eine Menge von Assoziationen zu anderen Modulen, zu denen eine Referenz existiert.

Beobachtung

- Um passende Module und Atome in den Modulrahmen aus 4.3.1.2 einsetzen zu können, ist eine Übersicht über die vorhandenen Lernobjekte sowie Zugriff auf potenziell jedes verfügbare Lernobjekt nötig. Um dies zu gewährleisten, sind sowohl zentrale Ansätze (beispielsweise Datenbanken, in denen die Lernobjekte abgelegt werden, oder Shared Workspaces) als auch dezentrale Ansätze (beispielsweise Austausch von Lernobjekten über Peer-To-Peer-Netzwerke) möglich (vgl. [CSCW], [BoSc00] und [MITO]).
- Die Anforderung der Erstellung von fehlenden Lernobjekten ist essenziell, denn ohne passende Lernobjekte kann ein Modul nicht aufgebaut werden. Die Zusammenarbeit zwischen den Modul- und Atomdesignern kann sehr unterschiedlich aussehen: Eng kooperierende Teams (beispielsweise in Unternehmen, die Module herstellen) sind ebenso denkbar wie lose gekoppelte Communities von Atom- und Moduldesignern[43] (beispielsweise interessierte Privatpersonen). Notwendige Voraussetzung hierfür ist allerdings ein Mechanismus zur Verteilung und Zuweisung von Anforderungen an Modul- und Atomdesigner. Dieser Aspekt wird später im Detail behandelt.
- In vielen Fällen wird dieser Teilprozess mit dem Teilprozess „Design eines Moduls“ verschränkt ausgeführt werden, insbesondere dann, wenn die gleichen Personen mit den jeweiligen Teilprozessen betraut sind. Eine getrennte Durchführung dieser Teil-

[43] Ein interessantes Beispiel einer Community mit ähnlicher Zielsetzung sind die Autoren von Wikipedia (http//www.wikipedia.org), einem online verfügbaren Nachschlagewerk. Interessierte Privatpersonen und Spezialisten liefern Erklärungen für Begriffe, die in der Summe zu einer Enzyklopädie gesammelt werden.

prozesse ist eher dann zu erwarten, wenn verschiedene Personen bzw. Personengruppen mit dem Design und der Entwicklung eines Moduls betraut sind.

4.3.1.4 Teilprozess: Überarbeiten eines existierenden Moduls

In 4.3.1.2 und 4.3.1.3 wurde beschrieben, wie ein Modul entworfen und mit Inhalt befüllt wird. Ausgangspunkt war hierbei immer der Wunsch, ein neues Modul zu schaffen. In vielen Fällen ist es hingegen nötig bzw. sinnvoll, ein bereits bestehendes Modul zu überarbeiten bzw. anzupassen (vgl. Umsetzung der Wiederverwendung in 1.4.3). Gründe für eine Modifikation eines Moduls können sein:

- Inhaltliche bzw. redaktionelle Überarbeitung, beispielsweise aufgrund von Ungenauigkeiten oder sachlichen Fehlern.
- Erstellung eines neuen Moduls auf der Basis eines bereits bestehenden Moduls. In diesem Fall wird das bestehende Modul kopiert und die Kopie passend überarbeitet. Dieses Vorgehen kann ressourcenschonender sein als ein Modul komplett neu zu entwickeln.

Eine Modifikation eines Moduls kann dessen Beschreibung (Außenschnittstelle), dessen Inhalt (beispielsweise redaktionelle Überarbeitung eines Textes[44], der als Atom vorliegt) oder dessen Struktur (beispielsweise Einfügen neuer Subelemente) betreffen.

Beteiligte Rollen

- Moduldesigner
- Evtl. Atomdesigner und weitere Moduldesigner

Eingangsdaten

- Änderungsanforderung.

  Die Änderungsanforderung muss nicht notwendigerweise konkrete Handlungsanweisungen („An Position x bitte Modul y einfügen!“) beinhalten. Sie ist meist recht vage formuliert („Absatz x ist zu ungenau. Bitte präzisieren!“). In diesem Fall liegt es am Moduldesigner, gegebenenfalls in Rücksprache mit dem Änderungsanforderer konkrete Handlungsanweisungen zu erarbeiten.
- Konkretes Modul, das Inhalt enthält und korrekt parametrisiert ist.

Ergebnis

- Konkretes, überarbeitetes Modul, das Inhalt enthält und korrekt parametrisiert ist. Dieses Modul kann für die Erstellung von Kursen oder weiteren Modulen (wieder-) verwendet werden.

Detaillierte Beschreibung der Aktivitäten

Ausgangspunkt einer Überarbeitung eines Lernobjekts ist immer eine Änderungsanforderung. Diese muss in einer Anforderungsanalyse wieder analysiert und in präzise Handlungsanweisungen übersetzt werden. Bei den Handlungsanweisungen müssen auch Parameter und Constraints mit berücksichtigt werden.

Es müssen vier verschiedene Arten der Überarbeitung unterschieden werden: Änderungen der Außenschnittstelle eines Lernobjekts, Änderung der Anbindung eines Moduls an andere Lernobjekte, Änderungen der Struktur eines Moduls, Änderungen des Inhalts eines Lernob-

[44] Redaktionelle Änderungen wären beispielsweise die Beseitigung von Rechtschreibfehlern, stilistische Überarbeitungen, etc.

jekts oder beides. Im folgenden werden die Aktivitäten, die jeweils für die entsprechende Überarbeitung durchgeführt werden, vorgestellt.

- Fall 1: Überarbeitung der Außenschnittstelle eines Lernobjekts (Modul oder Atom)

  Die Überarbeitung der Außenschnittstelle (Parametrisierung) eines Lernobjekts umfasst die Änderung der Parameter, die das Lernobjekt beschreiben, oder das Hinzufügen oder Weglassen neuer beschreibender Parameter oder Constraints. Dabei muss immer beachtet werden, dass die Inhalte des Lernobjekts nach wie vor zu den beschreibenden Parametern passen.[45]

  Eine mögliche Anwendung dieser Art der Überarbeitung ist die Präzisierung des Lernobjekts, indem die Werte bereits existierender Parameter korrigiert oder noch fehlende Parameter ergänzt werden. Die Wiederverwendung des Lernobjekts wird durch neu hinzukommende Parameter zwar möglicherweise etwas eingeschränkt[46], aber dies kann toleriert werden, da das Lernobjekt nun besser beschrieben und somit präziser nutzbar wird. Im Ergebnis werden besser aufeinander abgestimmte Lernobjekte verwendet, wodurch man qualitativ höherwertige Lehrmaterialien erhält.

  Jede Änderung der Außenschnittstelle (Parametrisierung) eines Lernobjekts ändert die Möglichkeiten seiner Wiederverwendung, da über die Außenschnittstelle festgelegt wird, in welche Module das Lernobjekt verbaut werden kann. Eine derartige Überarbeitung ist daher mit großer Umsicht durchzuführen.

- Fall 2: Überarbeitung der Anbindung eines Moduls an andere Lernobjekte

  Diese Art der Überarbeitung betrifft die Assoziationen, über die ein Modul an andere Lernobjekte angebunden ist.

  Eine Überarbeitung dieser Art kann nötig werden, wenn man Semantik tragende Beziehungen zu Lernobjekten herstellen will, beispielsweise indem man aus didaktischen Gründen Querbeziehungen zu thematisch verwandten Lernobjekten herstellt.

  Es müssen drei Arten der Überarbeitung unterschieden werden:

  - Hinzufügen einer Assoziation zwischen dem betrachteten Modul und einem anderen Lernobjekt
  - Entfernen einer Assoziation zwischen dem betrachteten Modul und einem anderen Lernobjekt
  - Ändern der Parameter einer Assoziation zwischen dem betrachteten Modul und einem anderen Lernobjekt

- Fall 3: Überarbeitung der Struktur eines Moduls

  Eine Überarbeitung der Struktur eines Moduls kann nötig werden, wenn neue Aspekte betrachtet und somit neue Subelemente hinzugefügt werden müssen, oder

[45] Dies kann beispielsweise problematisch werden, wenn der Parameter „Schwierigkeitsgrad“ eines Moduls von „schwer“ auf „einfach“ geändert wird. Damit die Beschreibung des Moduls mit dessen Inhalt konsistent bleibt, müssen auch die Werte des Parameters „Schwierigkeitsgrad“ der Platzhalter aller Subelemente angepasst werden. Sie müssen nun ebenfalls auf „einfach“ gesetzt werden. In der Folge werden möglicherweise komplett andere Subelemente für die Platzhalter ausgewählt werden, wodurch sich das Modul grundlegend ändern kann.

[46] Durch die neu hinzukommenden Parameter kann es vorkommen, dass ein Lernobjekt (Modul oder Atom) nun bei Platzhaltern nicht mehr verwendet werden kann, in die es vorher noch eingesetzt werden konnte; Parameter, die bei der Auswahl der Module vorher nicht berücksichtigt werden konnten (die Parameter waren im Lernobjekt nicht gesetzt), müssen nun ausgewertet werden.

wenn – beispielsweise aus didaktischen Gründen – die Reihenfolge der Submodule[47] geändert werden muss.

Es müssen folgende Arten der Überarbeitung der Struktur unterschieden werden:

- Einfügen eines neuen Submoduls
- Entfernen eines Submoduls im Modul inklusive aller Assoziationen, die von diesem Submodul ausgehen bzw. zu diesem Submodul führen.
- Änderung der Reihenfolge der Submodule

Abbildung 4-9 illustriert die bei der Überarbeitung durchzuführenden Aktivitäten.

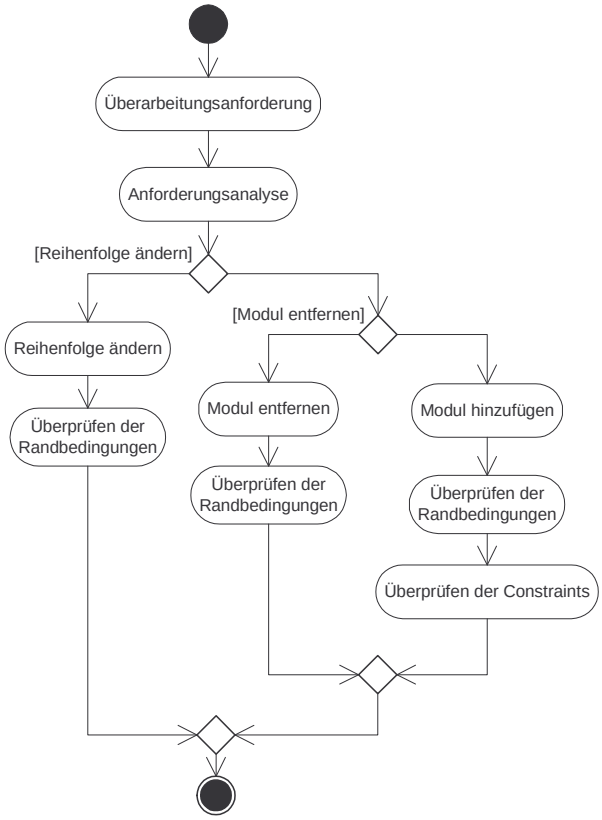


Abbildung 4-9: Überarbeitung der Struktur eines Moduls

Jede Änderung der Struktur eines Lernobjekts hat möglicherweise Einfluss auf die Assoziationen, die von diesem Lernobjekt ausgehen bzw. zu diesem Lernobjekt führen. Eine derartige Überarbeitung ist daher mit großer Umsicht durchzuführen (vgl. Fall 2).

Beim Einfügen eines neuen Lernobjekts muss darüber hinaus sichergestellt sein, dass auch für das neue Lernobjekt alle Constraints gültig sind, die für das zu überarbeitende Modul gelten. Hier gelten die gleichen Regeln, wie sie bereits in 4.3.1.3 genannt wurden.

[47] Zur Erinnerung: Es wird davon ausgegangen, dass Atome immer in Modulen gekapselt sind. Hier werden daher nur Module, nicht aber einzelne Atome betrachtet.

- Fall 4: Überarbeitung des Inhalts eines Lernobjekts (Modul oder Atom)

  Der Inhalt eines Lernobjekts wird – abhängig von der Art des Lernobjekts – folgendermaßen überarbeitet: Atome werden vom Atomdesigner direkt überarbeitet (vgl. Weiterentwicklung eines Lernobjekts in 2.4.3); bei Atomen, die Text enthalten, beispielsweise mittels Textverarbeitungsprogrammen wie Microsoft Word, oder bei Atomen, die Grafiken enthalten, beispielsweise mit Grafikprogrammen wie Photoshop oder Adobe Illustrator. Dabei muss darauf geachtet werden, dass die Parameter, die die Atome beschreiben, falls nötig angepasst werden.

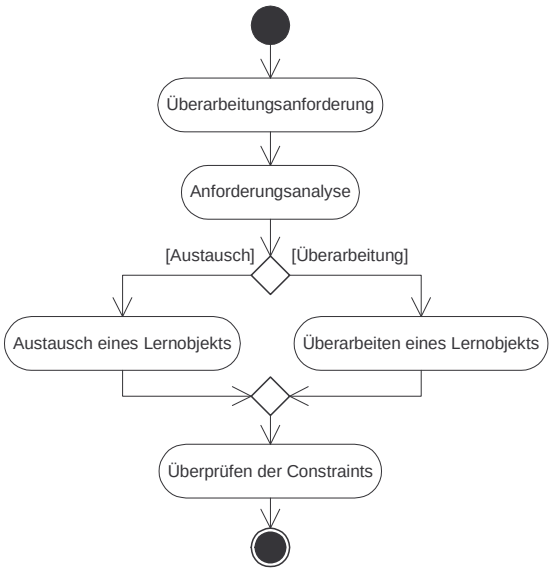


Abbildung 4-10: Überarbeitung des Inhalts eines Lernobjekts

Die inhaltliche Überarbeitung von Modulen geschieht über den Austausch oder die Überarbeitung von ihren Subelementen: Die Atome oder Module, die das zu überarbeitende Modul enthält, werden gegen inhaltlich äquivalente Atome bzw. Module ausgetauscht. So kann beispielsweise auch ein Atom, das eine textuelle Beschreibung eines Sachverhalts enthält, gegen ein Atom ausgetauscht werden, das den gleichen Sachverhalt in einer Animation illustriert. Hierfür bieten sich Alternativen an, vgl. 3.4.3. Abbildung 4-10 illustriert die durchzuführenden Aktivitäten.

Die inhaltliche Gleichwertigkeit der Subelemente wird dadurch sichergestellt, dass die Parameter der Platzhalter, in die die Subelemente eingesetzt werden sollen, unverändert bleiben. In den Platzhalter können somit nur solche Subelemente eingesetzt werden, die untereinander Alternativen sind. Ist dies nicht so, liegt wieder Fall 1 vor.

Analog zu strukturellen Änderungen eines Moduls muss auch hier die Gültigkeit aller Constraints geprüft werden, um die Konsistenz des Moduls gewährleisten zu können.

Alle diese Fälle der Überarbeitungen eines Moduls sind oft voneinander abhängig; sie werden meist nicht separat ausgeführt. Daher ist auf jeden Fall eine Nacharbeitung notwendig, in der Inhalt und Struktur entsprechend aufeinander abgestimmt werden müssen.

Beobachtung

- Grundlage der Überarbeitung eines Moduls ist eine Änderungsanforderung. Diese muss an den Moduldesigner herangetragen werden. Dies kann entweder direkt, von Moduldesigner zu Moduldesigner, oder indirekt über eine verteilende Stelle geschehen.

  Eine direkte Kommunikation zwischen Änderungsanforderer und Moduldesigner ist dann möglich, wenn die Verantwortlichkeiten für ein Modul eindeutig geklärt sind und eine Kontaktmöglichkeit (E-Mail, Telefon, etc.) besteht. In einer großen Gemeinschaft von voneinander unabhängig arbeitenden Moduldesignern, die sich nicht einmal kennen müssen, kann dies problematisch werden. Eine Lösung hierfür könnte darin bestehen, zu jedem Modul Ansprechpartner und Kontaktdaten zu nennen, so dass jeder, der Änderungen an einem Modul beantragen möchte, weiß, an wen er sich wenden kann. Beim Wechsel von Verantwortlichkeiten sind dann allerdings die jeweiligen Daten im Modul aktuell zu halten.

  Eine indirekte Kommunikation zwischen Änderungsanforderer und Moduldesigner ist beispielsweise über einen Verteilungsdienst möglich, der Änderungsanfragen entgegennimmt und automatisch an die zuständigen Moduldesigner weiterleitet. Auch bei diesem Ansatz sind Pflegearbeiten bei einem Wechsel der Zuständigkeiten für ein Modul durchzuführen: Die Zuordnung „Modul – Moduldesigner“ im Verteilungsdienst muss ständig aktuell gehalten werden, damit der Verteilungsdienst bei Änderungsanforderungen jeweils die richtigen Moduldesigner benachrichtigen kann.

- Um Lernobjekte gegen andere Lernobjekte austauschen zu können, ist wieder eine Übersicht über alle verfügbaren Lernobjekte nötig, vgl. auch schon 4.3.1.3.

- Beim Austauschen von Lernobjekten können bedingt durch die Baumstruktur eines Moduls potenziell sehr viele Parameter zu prüfen sein. Wird beispielsweise ein kompletter Teilbaum, der in einem Modul gekapselt ist, gegen einen anderen Teilbaum ausgetauscht, muss für alle Lernobjekte im Teilbaum die Gültigkeit der Constraints sichergestellt werden, die für den Platzhalter festgelegt wurden, in den der Teilbaum eingefügt werden soll.

4.3.2 Design, Entwicklung und Überarbeitung eines Kurses

4.3.2.1 Überblick

Der Kursdesigner arbeitet mit Kursen und Modulen. Er nutzt die Ergebnisse des Moduldesigners, indem er die Module sammelt und diese je nach Kurs entsprechend den Außenschnittstellen der Module zusammensetzt. Es wird davon ausgegangen, dass alle Atome in Modulen gekapselt sind.

Die Aufgaben des Kursdesigners sind in Abbildung 4-11 dargestellt.

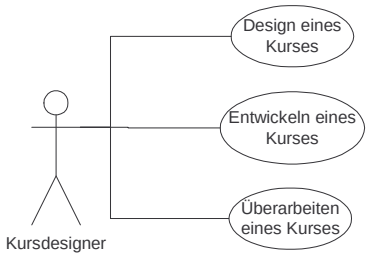


Abbildung 4-11: Aufgaben des Kursdesigners

Der Kursdesigner hat die Aufgabe, einen Kurs für ein bestimmtes Thema zu entwerfen (Teilprozess „Design eines neuen Kurses“), aus bestehenden Modulen und dem Ergebnis des Kursdesigns einen neuen konkreten Kurs aufzubauen (Teilprozess „Entwickeln eines neuen Kurses“) oder auf Basis eines existierenden Kurses einen neuen Kurs zu gestalten bzw. einen bestehenden Kurs zu überarbeiten (Teilprozess „Überarbeiten eines existierenden Kurses“). Den hierbei entstehenden Kursen ist noch kein Präsentationsformat zugeordnet; dies geschieht erst in einem der Teilprozesse „Präsentation eines adaptierbaren Kurses in einer Lehrveranstaltung“ (vgl. 4.3.3.3), „Präsentation eines customized Kurses in einer Lehrveranstaltung“ (vgl. 4.3.3.4), „Lernen durch Nutzung von adaptierbaren Kursen“ (vgl. 4.3.5.2) oder „Lernen durch Aufbau und Nutzung von customized Kursen“ (vgl. 4.3.5.3).

4.3.2.2 Design eines Kurses

Das Design eines neuen Kurses ist sehr ähnlich zum Design eines neuen Moduls; die Aufgaben, die der Kursdesigner hier zu erledigen hat, ähneln sehr denen des Moduldesigners aus 4.3.1.2. Der Unterschied zum Design eines neuen Moduls besteht darin, dass beim Design eines Kurses immer auch die didaktische Anwendung des Kurses (beispielsweise die Nutzung in einer Vorlesung, die angestrebte Zielgruppe, was beispielsweise Auswirkungen auf den Schwierigkeitsgrad hat, etc.) berücksichtigt werden muss (vgl. 3.5.2.1). Dies ist beim Design eines Moduls nicht nötig.

Beim Design eines Kurses definiert der Kursdesigner den Rahmen des Kurses, im folgenden *Kursrahmen* genannt: Ein Kursrahmen ist ein Kurs, bei dem die Parameter festgelegt sind und der mit komplett parametrisierten Platzhaltern versehen ist. Den Platzhaltern sind jedoch noch keine Module zugewiesen.

Beteiligte Rollen

- Kursdesigner

Eingangsdaten

- Parameter und Constraints, die das Thema sowie die vorgesehenen Nutzungen des Kurses beschreiben.
- Eine Menge von Modulen sowie Parameter, die diese Module beschreiben.

Analog zum Design eines Moduls liegen die Parameter und Constraints im allgemeinen als natürlichsprachliche Beschreibung von Eigenschaften vor. Diese Beschreibungen kommen üblicherweise von Kurs-Presentern, die Kurse mit ganz bestimmtem Inhalt und, entsprechend ihrer vorgesehenen Nutzung, ganz bestimmten Eigenschaften benötigen. Es ist die Aufgabe des Kursdesigners, diese Beschreibungen in eine korrekte Menge von Parametern und Constraints umzusetzen.

Ergebnis

- Ein Kursrahmen, also ein Kurs, dessen Platzhalter korrekt parametrisiert sind und der selbst korrekt mittels Parametern beschrieben ist.

Detaillierte Beschreibung der Aktivitäten

Der Kursdesigner definiert den Kursrahmen. Dieser Rahmen besteht aus mit Parametern beschriebenen Platzhaltern, in die konkrete Module eingesetzt werden können, sowie aus Para-

metern, die den Kurs selbst beschreiben. Die einzelnen Aktivitäten des Kursdesigners sind in Abbildung 4-12 dargestellt[48].

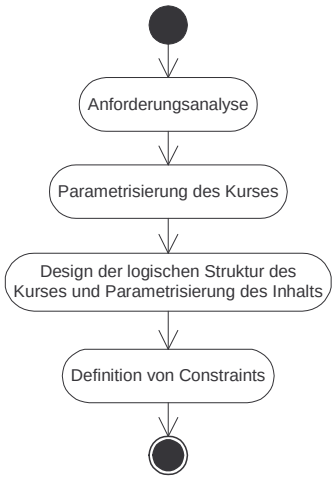


Abbildung 4-12: Design eines Kurses

Aktivität: Anforderungsanalyse (Analyse und Einschränkung des Themas)

Der Kursdesigner arbeitet auf Basis eines vorgegebenen Themas sowie bestimmter Randbedingungen (Detaillierungsgrad, verfügbares Vorwissen, etc.). Bei der Anforderungsanalyse setzt sich der Kursdesigner mit dem Thema auseinander und prüft, welchen Anforderungen unter den gegebenen Randbedingungen ein Kurs zu dem geforderten Thema genügen muss. Neben Aspekten, die Inhalt oder Struktur des Kurses betreffen, muss hier auch die beabsichtigte Nutzung berücksichtigt werden. Dies betrifft sowohl Art als auch Gestaltung der Lehrveranstaltung: Ein Kurs, der beispielsweise für das Fernstudium gestaltet wird, sieht völlig anders aus als ein Kurs, der für eine Vorlesung (klassische Präsenzveranstaltung) entworfen wird. Während der Kurs für das Fernstudium sehr detailliert ausgestaltet und mit vielen Beispielen und Übungen versehen wird, wird der Kurs für eine Vorlesung nur sehr knapp und übersichtlich die wesentlichen Punkte nennen, die der Kurs-Presenter „auf der Tonspur“ ausführlich darstellt.

Das Ergebnis dieser Aktivität ist die inhaltliche, strukturelle Grobvorstellung des aufzubauenden Kurses; auch Aspekte der vorgesehenen Nutzung sind hier bereits angedacht.

Aktivität: Parametrisierung des Kurses

Aus den Ergebnissen der Anforderungsanalyse leitet der Kursdesigner die Parameter und Constraints, mit denen der Kurs beschrieben wird, ab. So legt der Kursdesigner die wesentlichen Eigenschaften des Kurses (Außenschnittstelle) fest, anhand derer der Kurs später von Kurs-Presentern abgerufen werden kann. Grundlage hierbei sind wieder die in 3.3.1.3 und 3.5.2.1 beschriebenen Parameter.

Die Parametrisierung erfolgt nach folgenden Gesichtspunkten:

- Inhalt und Struktur
- Nutzungen

[48] Genau wie beim Design eines Moduls werden die einzelnen Aktivitäten nicht immer notwendigerweise streng in dieser Reihenfolge ausgeführt.

Die Parametrisierung des Inhalts und der Struktur erfolgt analog zum Teilprozess „Design eines Moduls“, vgl. 4.3.1.2. Im Unterschied zur Aktivität „Design eines Moduls“ muss hier zusätzlich die beabsichtigte Nutzung des Kurses berücksichtigt und entsprechend in den Parametern festgelegt werden. Die Parametrisierung der Nutzung erfolgt, wie bereits bei der Beschreibung der Aktivität „Anforderungsanalyse“ erwähnt, entsprechend der vorgesehenen Nutzung. Diesbezüglich sind unterschiedliche Parametrisierungen des Inhalts möglich (wie Kurse mit Detaillierungsgrad „niedrig“ oder „höher“ für unterschiedliche Zielgruppen), die durch Constraints eingeschränkt werden können.

Das Ergebnis dieser Aktivität ist eine formale Beschreibung des Kurses mittels Parametern und Constraints.

Aktivität: Design der logischen Struktur des Kurses (Parametrisierung der Kompositionsbeziehungen) und Parametrisierung des Inhalts

Beim Design eines Kurses definiert der Kursdesigner anhand der Ergebnisse aus der Anforderungsanalyse den strukturellen Aufbau des Kurses. Die Grobstruktur des Kurses wird dabei durch die Kategorien definiert, denen der Kursdesigner den Kurs beispielsweise durch Auswahl eines Themengebiets und entsprechend dem Lehrplan zugewiesen hat. Analog zum Moduldesign werden in diesem Schritt keine konkreten Kurse vorgegeben, sondern lediglich deren Eigenschaften mittels Parametern festgelegt.

Bei der Parametrisierung der Module des Kurses muss wieder zwischen verpflichtend vorgeschriebenen und optionalen Subelementen unterschieden werden. Die Regeln (realisiert durch Constraints), nach denen die Werte der Parameter für die Platzhalter belegt werden, entsprechen denen, die bereits im Zusammenhang mit dem Design von Modulen in 4.3.1.2 beschrieben wurden. Für Details vgl. dort.

Das Ergebnis dieser Aktivität ist eine Folge von Platzhaltern für Lernobjekte, die den logischen Aufbau des Kurses definiert.

Aktivität: Definition von Constraints

Die im letzten Schritt aufgebaute Folge von Modulen und Atomen wird in diesem Schritt mittels Constraints verfeinert: Der Kursdesigner schränkt auf Grundlage der Ergebnisse der Anforderungsanalyse gezielt die Wertebereiche von Parametern[49] ein, um die Module oder Atome, die in die Platzhalter eingesetzt werden können, möglichst präzise zu beschreiben. Dies geschieht mittels Constraints, die der Kursdesigner auf den Parametern definiert.

Auch hier muss der Kursdesigner wieder die beabsichtigte Nutzung des Kurses berücksichtigen und auch solche Constraints definieren, die in der Nutzung begründet sind. Ist beispielsweise aus der Anforderungsanalyse heraus der Parameter „Zielgruppe“ mit „Informatik Grundstudium“ belegt, so muss der Wert des Parameters „Schwierigkeitsgrad“ immer „leicht“ oder höchstens „mittel“ sein.

Ergebnis dieser Aktivität ist eine Menge von Constraints, die nun deutlich präziser beschreibt, welche Module an einem Platzhalter eingesetzt werden dürfen. Ebenso ist der Kurs selbst nun präziser beschrieben.

Zu beachten ist, dass auch diese Aktivität wieder verschränkt mit der Aktivität „Design der logischen Struktur eines Kurses und Parametrisierung des Inhalts“ ablaufen kann.

[49] Parameter der Außenschnittstelle des Kurses ebenso wie die Parameter, die für die Platzhalter von Modulen definiert sind.

Beobachtung

- Das Design eines Moduls ist mit einer gewissen „Blindheit“ gegenüber der geplanten Nutzung in der Lehre möglich; Gedanken bzgl. didaktischer Rahmenbedingungen sind zwar bereits beim Design eines Moduls sinnvoll, jedoch haben sie nicht die Bedeutung wie beim Design eines Kurses. Beim Design eines Moduls überwiegen Aspekte der Wiederverwendung, denn das Modul soll nicht nur in einem, sondern idealerweise in vielen Kursen und auf vielerlei Weise einsetzbar sein.

  Jeder Kurs hingegen wird initial für eine bestimmte Lehrveranstaltung, in der er eingesetzt werden soll, gestaltet; Aspekte der Wiederverwendung in anderen Lehrveranstaltungen stehen hier nicht im Vordergrund. Dies ist ein wesentlicher Unterschied zum Design eines Moduls. Für das Design eines Kurses sind die didaktischen Rahmenbedingungen (Zielgruppe, Art der Lehrveranstaltung wie Tutorübung, Präsenzvorlesung, etc.) der Lehrveranstaltung, in der er eingesetzt werden soll, maßgeblich; Gedanken der Wiederverwendung treten gegenüber einer aus didaktischen Gesichtspunkten möglichst optimalen Gestaltung des Kurses vorerst in den Hintergrund. Erst bei mehrfacher Verwendung des Kurses wird man sich Gedanken darüber machen, wie der Kurs flexibler in Lehrveranstaltungen eingesetzt werden kann.

- Bei der Parametrisierung des Inhalts des Kurses werden – analog zum Design eines Moduls – keine Annahmen über die Existenz bestimmter Module gemacht (analog 4.3.1.2).

- Des weiteren ist auch hier ein gemeinsames Verständnis der Parameter unbedingt erforderlich, da nur so die Bestückung der Platzhalter mit passenden (Sub-)Modulen möglich ist (analog 4.3.1.2).

4.3.2.3 Entwickeln eines Kurses

Ist der Kursrahmen spezifiziert (vgl. 4.3.2.2), füllt ihn der Kursdesigner entsprechend der vorgesehenen Nutzung, wie im Teilprozess „Design eines neuen Kurses“ festgelegt, mit Inhalten. Er sorgt dafür, dass die Module, die im letzten Schritt als Teile des Kurses angegeben wurden, verfügbar sind. Sind geforderte Module nicht verfügbar, stellt der Kursdesigner eine entsprechende Anfrage an die Moduldesigner. Auch wenn einige Aktivitäten dieses Teilprozesses möglich sind, ohne dass alle geforderten Module verfügbar sind, kann dieser Teilprozess erst dann abgeschlossen werden, wenn alle geforderten Module geliefert wurden.

Dieser Teilprozess ist dem Teilprozess „Entwickeln eines neuen Moduls“ (vgl. 4.3.1.3) sehr ähnlich.

Beteiligte Rollen

- Kursdesigner
- Evtl. Moduldesigner

Eingangsdaten

- Der Kursrahmen, der in 4.3.2.2 erstellt wurde.
- Module gemäß der Spezifikation im Kursrahmen. Für deren Erstellung vgl. 4.3.1.2 und 4.3.1.3.

Ergebnis

- Konkreter Kurs, der Inhalt und Struktur enthält und korrekt parametrisiert ist. Dieser Kurs wird vom Kurs-Presenter, vom Kurstutor oder vom Lerner in Lehrveranstaltungen und im Selbststudium genutzt.

Detaillierte Beschreibung der Aktivitäten

In diesem Teilprozess fügt der Kursdesigner dem Kursrahmen, der im letzten Schritt erstellt wurde, konkrete Inhalte hinzu. Dies bedeutet im wesentlichen, dass der Kursdesigner dafür sorgt, dass die Module zur Verfügung stehen, aus denen der Kurs aufgebaut ist. Abbildung 4-13 zeigt den Teilprozess im Überblick.

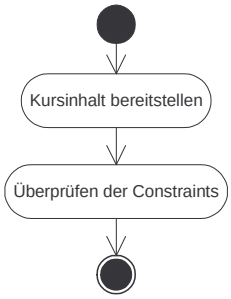


Abbildung 4-13: Entwicklung eines Kurses

Aktivität: Kursinhalt bereitstellen

Der Kursdesigner sorgt dafür, dass alle Module, die im Kurs enthalten sind, bereitstehen. Falls Module noch nicht oder nicht in der gewünschten Form zur Verfügung stehen, stellt der Kursdesigner eine Anfrage an die Moduldesigner, dass diese die noch fehlenden Module erstellen, entwickeln oder geeignet überarbeiten. In diesem Fall muss der Kursdesigner den Moduldesignern die Eigenschaften der zu erzeugenden Module genau angeben, denn nur Module, die diese Eigenschaften erfüllen, können später in den Kursrahmen eingesetzt werden.

Es ist hierbei zu beachten, dass die Außenschnittstellen der Module den Parametern des in 4.3.2.2 erstellten Kursrahmens genügen.

Ergebnis dieser Aktivität ist eine Menge von Modulen, die der zu entwickelnde Kurs als Elemente beinhaltet. Diese Module werden nach der nächsten Aktivität „Überprüfen der Constraints“ in den Kursrahmen integriert, so dass dann ein konkreter Kurs vorliegt.

Aktivität: Überprüfen der Constraints

Stehen alle Module für einen Kurs bereit, muss der Kursdesigner überprüfen, ob für die bereitgestellten Module alle Constraints, die auf den Platzhaltern des Kurses definiert sind, erfüllt werden. Es findet hier ein Abgleich der Parameter des Kurses mit den Parametern der in den Kurs einzusetzenden Module statt. Dieser Schritt ist notwendig, um die Stimmigkeit des Kurses sicherzustellen.

Diese Aktivität hat kein explizit aufführbares Ergebnis, sondern dient der Qualitätssicherung.

Beobachtung

- Um passende Module in den Kursrahmen aus 4.3.2.2 einsetzen zu können, ist eine Übersicht über die vorhandenen Module sowie Zugriff auf potenziell jedes verfügba-

re Modul nötig. Hierzu müssen, wie bereits in 4.3.1.3 erwähnt, entsprechende Technologien bereitgestellt werden.

- Die Beantragung der Erstellung von fehlenden Modulen ist essenziell, denn ohne passende Module kann der Kurs nicht aufgebaut werden.
- In vielen Fällen wird dieser Teilprozess mit dem Teilprozess „Design eines Kurses“ verschränkt ausgeführt werden, insbesondere dann, wenn die gleichen Personen mit den jeweiligen Teilprozessen beschäftigt sind. Dies ist der Regelfall, da bei der Gestaltung (Design und Entwurf) eines Kurses didaktische Überlegungen und Überlegungen bzgl. der beabsichtigten Nutzung eine wichtige Rolle spielen und durch die Verschränkung beider Teilprozesse dies besser berücksichtigt werden kann.

4.3.2.4 Überarbeiten eines existierenden Kurses

Analog zur Überarbeitung von bereits existierenden Modulen können auch Kurse überarbeitet werden. Die Motivation hierfür ist ähnlich wie bei der Überarbeitung von Modulen (vgl. 4.3.1.4):

- Inhaltliche bzw. redaktionelle Überarbeitung aufgrund von Ungenauigkeiten oder sachlichen Fehlern.
- Erstellung eines neuen Kurses (evtl. für eine neue Anwendung) auf der Basis eines bereits bestehenden Kurses. In diesem Fall wird der bestehende Kurs kopiert und die Kopie passend überarbeitet.

Eine Modifikation eines Kurses kann dessen Beschreibung (Außenschnittstelle), dessen Inhalt (beispielsweise redaktionelle Überarbeitung eines Moduls) oder dessen Struktur (beispielsweise Einfügen neuer Module) betreffen.

Beteiligte Rollen

- Kursdesigner
- Evtl. Moduldesigner

Eingangsdaten

- Änderungsanforderung.

  Die Änderungsanforderung muss nicht notwendigerweise konkrete Handlungsanweisungen beinhalten. Es ist die Aufgabe des Kursdesigner, gegebenenfalls in Rücksprache mit dem Änderungsanforderer konkrete Handlungsanweisungen zu erarbeiten.
- Konkreter Kurs, der Inhalt enthält und korrekt parametrisiert ist.

Ergebnis

- Konkreter, überarbeiteter Kurs, der Inhalt enthält und korrekt parametrisiert ist.

Detaillierte Beschreibung der Aktivitäten

Ausgangspunkt einer Überarbeitung eines Kurses ist immer eine Änderungsanforderung. Diese muss in einer Anforderungsanalyse analysiert und in präzise Handlungsanweisungen übersetzt werden.

Analog zur Überarbeitung eines Moduls (vgl. 4.3.1.4) müssen drei verschiedene Arten der Überarbeitung unterschieden werden: Änderungen der Außenschnittstelle eines Kurses, Änderungen der Struktur eines Kurses und Änderungen des Inhalts eines Kurses. Im folgenden

werden die Aktivitäten, die jeweils für die entsprechende Überarbeitung durchgeführt werden, vorgestellt.

- Fall 1: Überarbeitung der Außenschnittstelle / der möglichen Nutzung eines Kurses

  Die Überarbeitung der Außenschnittstelle eines Kurses umfasst die Änderung der Werte der Parameter, die den Kurs beschreiben, das Hinzufügen neuer beschreibender Parameter oder das Weglassen bereits definierter und belegter Parameter. (Letztlich werden so die Einsatzmöglichkeiten des Kurses verändert.) So kann man einen Kurs beispielsweise an eine veränderte Zielgruppe („Studenten im Grundstudium" statt „Studenten im Hauptstudium") anpassen: Der Parameter „Schwierigkeitsgrad" des Kurses wird entsprechend der neuen Zielgruppe gesetzt. Genau wie bei der Überarbeitung eines Moduls muss immer beachtet werden, dass die Inhalte des Kurses nach wie vor zu den beschreibenden Parametern passen.

- Fall 2: Überarbeitung der Struktur eines Kurses

  Eine Überarbeitung der Struktur eines Kurses sollte – eine präzise und umsichtige Planung des Kurses vorausgesetzt – nur selten nötig sein. Dennoch kann es vorkommen, dass neue Inhalte in einen Kurs aufgenommen und damit neue Module zum Kurs hinzugefügt werden müssen. Es kann sich auch zeigen, dass aus didaktischen Gründen die Reihenfolge der Module im Kurs geändert werden muss.

  Es müssen folgende Arten der Überarbeitung der Struktur unterschieden werden:

  - Einfügen eines neuen Moduls
  - Entfernen eines Moduls
  - Änderung der Reihenfolge der Module im Kurs
  - Hinzufügen einer Assoziation zwischen zwei Lernobjekten im Kurs
  - Entfernen einer Assoziation zwischen zwei Lernobjekten im Kurs
  - Ändern der Parameter einer Assoziation zwischen zwei Lernobjekten im Kurs

  Abbildung 4-14 illustriert die bei der Überarbeitung durchzuführenden Aktivitäten.

  Jeder Kurs ist einer oder mehreren Kategorien zugeordnet (vgl. 3.2.1.3), die dem Kurs Randbedingungen[50] bezüglich seiner Struktur auferlegen. Bei allen drei Strukturänderungen muss sichergestellt sein, dass durch die Änderung diese Randbedingungen nicht verletzt werden. Andernfalls muss die Überarbeitung rückgängig gemacht oder der Kurs aus der jeweiligen Kategorie entfernt werden.

  Beim Einfügen eines neuen Moduls in einen Kurs muss darüber hinaus sichergestellt sein, dass auch für das neue Modul alle Constraints gültig sind, die für den zu überarbeitenden Kurs gelten. Auch hier gelten die gleichen Regeln, wie sie bereits in 4.3.1.3 genannt wurden.

[50] Zur Erinnerung: Randbedingungen sind beispielsweise der Detaillierungsgrad eines Themas, bestimmte Vorgaben hinsichtlich des strukturellen Aufbaus, etc. Randbedingungen liegen meist natürlichsprachig und nicht formalisiert vor. Durch die Formalisierung von Randbedingungen erhält man Constraints. Randbedingungen sind damit sozusagen eine Art „Vorstufe" von Constraints.

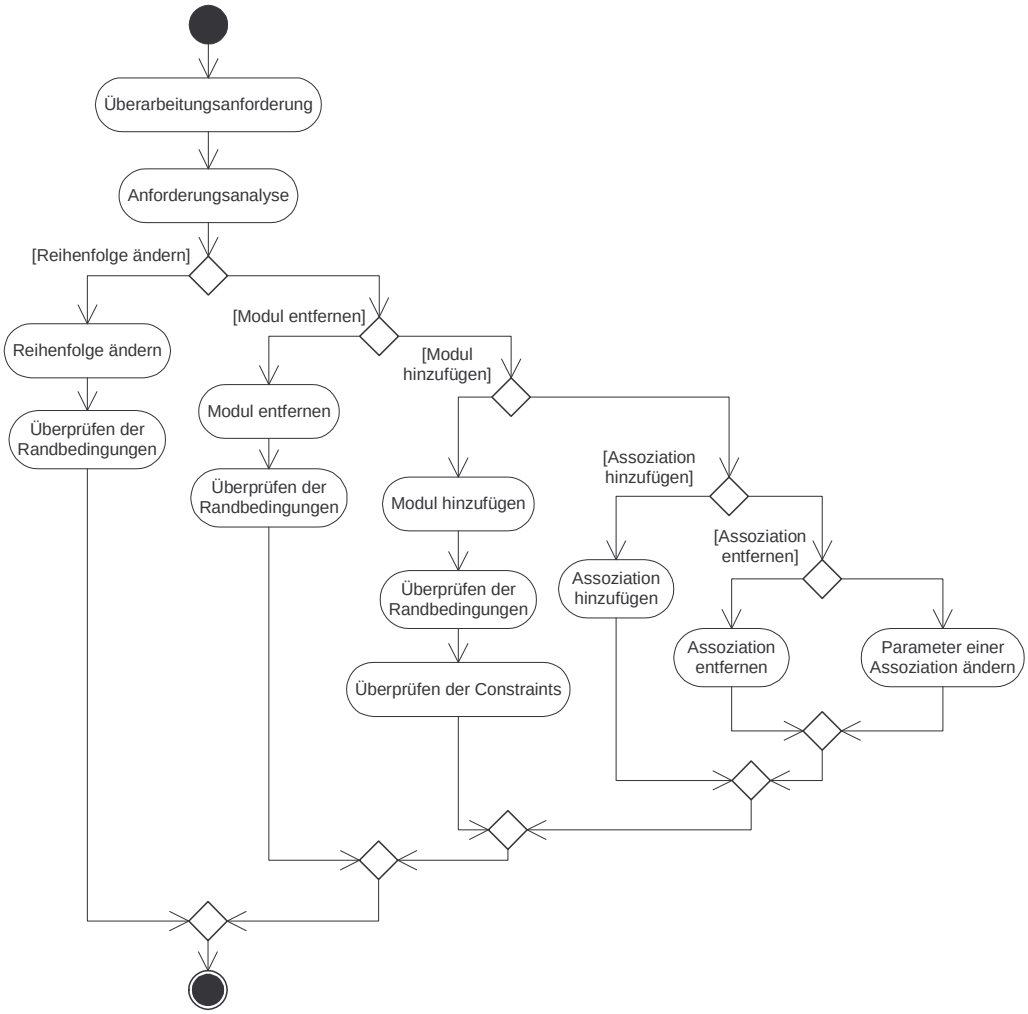


Abbildung 4-14: Überarbeitung der Struktur eines Kurses

- Fall 3: Überarbeitung des Inhalts eines Kurses

  Die inhaltliche Überarbeitung eines Kurses geschieht über den Austausch oder die Überarbeitung seiner Module: Module, die der zu überarbeitende Kurs enthält, werden gegen inhaltlich äquivalente Module ausgetauscht. So kann beispielsweise ein Modul, das eine textuelle Beschreibung eines Sachverhalts enthält, gegen ein Modul ausgetauscht werden, das den gleichen Sachverhalt in einer Animation illustriert.

  Die Überarbeitung eines Moduls wird nicht vom Kursdesigner selbst durchgeführt. Statt dessen fordert er von den Moduldesignern ein Modul mit den geforderten Eigenschaften, modelliert durch Parameter und Constraints, an.

  Die inhaltliche Gleichwertigkeit der Lernobjekte wird dadurch sichergestellt, dass die Parameter der Platzhalter, in die die Module eingesetzt werden sollen, sowie die Parameter des Kurses unverändert bleiben.

  Analog zu strukturellen Änderungen eines Kurses muss auch hier die Gültigkeit aller Constraints geprüft werden, um die Konsistenz des Kurses zu gewährleisten.

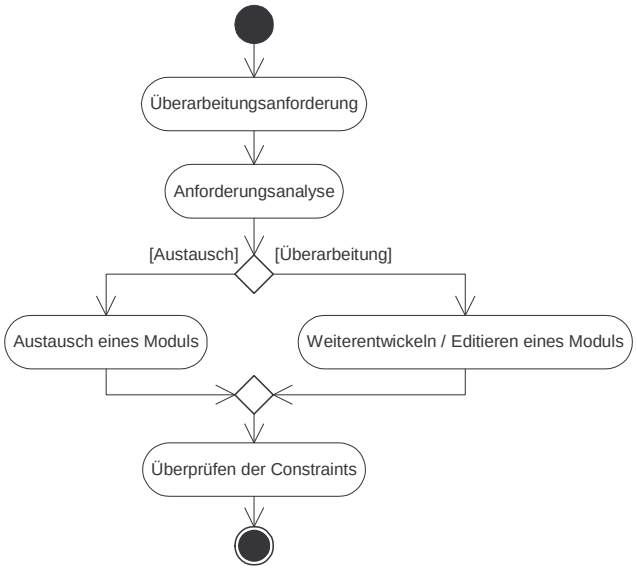


Abbildung 4-15: Überarbeitung des Inhalts eines Kurses

Beobachtung

- Grundlage der Überarbeitung eines Kurses ist eine Änderungsanforderung. Diese muss an den Kursdesigner herangetragen werden. Dies kann entweder direkt oder indirekt, über eine verteilende Stelle, geschehen. Dies ist sehr ähnlich zur in 4.3.1.4 beschriebenen Situation und soll hier nicht noch einmal wiederholt werden.
- Um Module gegen andere Module austauschen zu können, ist wieder eine Übersicht über alle verfügbaren Lernobjekte (hier insbesondere Module) nötig. Dies kann wieder über ein Repository, in dem alle Lernobjekte abgelegt werden, und einen Metadaten-Service, der die Beschreibungen aller Lernobjekte verwaltet, erreicht werden.
- Analog zur Überarbeitung eines Moduls können beim Austauschen von Lernobjekten bedingt durch die Baumstruktur eines Moduls potenziell sehr viele Parameter zu prüfen sein. Hier muss für eine Unterstützung durch geeignete Tools, die dies leisten können, gesorgt sein.

4.3.3 Adaptieren und Präsentieren eines Kurses

Die Aufgaben der Rolle des Kurs-Presenters unterteilen sich allgemein in Kursadaptierung und Kurspräsentation (vgl. Abbildung 4-16).

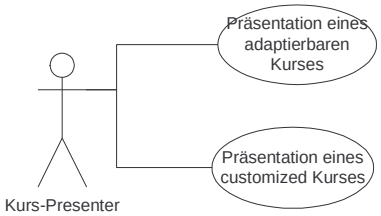


Abbildung 4-16: Aufgaben des Kurs-Presenters

4.3.3.1 Begriffsklärung: Adaptive, adaptierbare und customized Kurse

Ein Kurs, wie er vom Kursdesigner geliefert wird, ist noch nicht notwendigerweise auf eine bestimmte Nutzung oder Art der Präsentation ausgerichtet. Je nachdem, wer die Anpassung

des Kurses entsprechend der Nutzungsklassen „Lehre“, „Lernen“ und „Präsentation“ vornimmt, werden drei Arten von Kursen unterschieden:

- Ein *adaptierter* Kurs ist bereits vom Kursdesigner vollständig nutzungsspezifiziert. Der Kurs ist damit vom Kursdesigner von vorneherein für eine bestimmte Nutzung in der Lehre und vor allem auch für eine spezielle Nutzergruppe (das heißt eine Kursvariante) parametrisiert (vgl. Aktivität „Parametrisierung des Kurses“ des Kursdesigners, insbesondere die Parametrisierung nach der Art der Nutzung). Die Definition der Kursvariante und die Auswahl sowie Entwicklung der Kursvariante werden somit vom Kursdesigner bereits beim Kursdesign nach den Maßgaben der Nutzungsklassen „Lehre“, „Lernen“, „Präsentation“ etc., gesetzt durch den Kursdesigner, erledigt (vgl. 3.5.3).
- Bei einem *adaptierbaren* Kurs wird ein Kursrahmen durch den Kurs-Presenter anhand der vom Kursdesigner vorgesehenen Nutzungen passend parametrisiert. Hierzu setzt der Kurs-Presenter die Parameter des Kursrahmens entsprechend der beabsichtigten Nutzung und seinen persönlichen Vorlieben. Anhand dieser Angaben sind mittels Parameterpropagation an die Platzhalter des Kursrahmens sowie dann an die einzusetzenden Module[51] rekursiv alle Submodule des Kurses festgelegt. Erst dadurch ist der Kurs nicht nur inhaltlich und in seiner Struktur, sondern auch in seiner Nutzung in der Lehre abgeschlossen.
- Ein *customized* Kurs wird von den Nutzern (in Teilen) selbst aus bestehenden Modulen aufgebaut. In diesem Fall übernehmen die Nutzer sowohl Kursdesign als auch Kursentwicklung und entwerfen so Kurse entsprechend ihren Interessen.

  Unterscheidungsmerkmal zu einem adaptierbaren Kurs ist, dass bei einem customized Kurs die Nutzer (Kurs-Presenter, Kurstutor oder Lerner) den Teilprozess „Design eines Kurses“ mit dem Ergebnis eines Kursrahmens durchführen (können), während bei einem adaptierbaren Kurs die Kursrahmen bereits vorgegeben sind.

  Eine Spielart eines customized Kurses umfasst die Vorgabe eines bereits mehr oder weniger parametrisierten und mit Modulen versehenen Kursrahmens, der noch in Teilen parametrisiert und mit Modulen versehen werden kann. Die Nutzer können in diesem Fall nicht mehr völlig frei Module auswählen, sondern nur noch solche, die in ihren Parametern zu dem bereitgestellten Kursrahmen passen. Diese Einschränkung ist vor allem für die im allgemeinen (didaktisch) unerfahrenen Lerner besonders zu empfehlen (Details dazu siehe 4.3.5.3).

Der Kurs-Presenter nutzt in der Regel selbst keine adaptierten Kurse, sondern er konfiguriert bzw. adaptiert einen Kurs, der vom Kursdesigner geliefert wird, entsprechend seiner Vorstellungen, wie der Kurs in seiner Lehrveranstaltung eingesetzt werden soll. Dies geschieht über die Parameter der Nutzungsklasse „Lehre“ und „Präsentation“ (adaptierbarer Kurs). Im Extremfall gestaltet der Kurs-Presenter den Kurs neu (customized Kurs).

4.3.3.2 Vorbereitung eines Kurses für die Nutzung: Präsentation und Navigation

Die letzten beiden Schritte vor der Nutzung eines Kurses in einer Lehrveranstaltung oder zum Selbststudium ist immer die Festlegung eines Präsentationsformats (HTML, PDF, etc.), des Layouts und die Festlegung der verfügbaren Navigationspfade. Beide Schritte sind nicht unabhängig voneinander, denn die Wahl eines Präsentationsformats legt auch häufig die verfügbaren Navigationspfade fest. So erlaubt ein Kurs, der als Hypertext in HTML präsentiert werden soll, eine Vielzahl von Navigationsmöglichkeiten. Guided Tours sind ebenso mög-

[51] Dies ist im Detail in 3.6.2.2 beschrieben. Vgl. dort.

lich wie freies Browsen im Hypertext. Ein Kurs, der als PDF-Datei den Lernern bereitgestellt werden soll, ist hingegen in seiner Navigation stark eingeschränkt: Der Kurs muss als Folge von Modulen und Atomen dargestellt werden; er muss sequenzialisiert werden.

Festlegen der Präsentationsformats und Layouts

Auf der Basis von inhaltlich, strukturell und hinsichtlich ihrer didaktischen Nutzung abgeschlossenen Kursen können die Präsentationsformate des Kurses definiert werden. Hierbei ist die beabsichtigte Nutzung der Lehrmaterialien, festgelegt durch die Parameter der Nutzungsklasse „Lehre“, zu berücksichtigen.

Technisch werden die Layouts mittels Style Sheets realisiert. Je Präsentationsformat existiert mindestens ein Style Sheet[52]. Legen Kurs-Presenter oder Lerner ein Präsentationsformat fest, so geschieht dies implizit über die Auswahl eines Style Sheets, das auf den Kurs angewendet wird.

Festlegen der Navigationspfade

Grundlage für die Navigation sind Kompositionen und Assoziationen zwischen Lernobjekten (Modulen wie Atomen); Beziehungen bilden das Geflecht zwischen den Lernobjekten. Kompositionen realisieren die Struktur von Lernobjekten und Kursen; Assoziationen modellieren inhaltliche Zusammenhänge zwischen Lernobjekten. Während Kompositionen implizit definiert sind, werden Assoziationen von Moduldesignern festgelegt, vgl. 4.3.1.2 und 4.3.1.3.

Kompositionen und Assoziationen können für die Navigation im Kurs genutzt werden. Insbesondere Hypertext bietet hier einige Möglichkeiten. Je nachdem, welche Art von Navigation der Kurs-Presenter für den Kurs vorsieht, werden Kompositionen und Assoziationen unterschiedlich genutzt:

Freie Navigation in einem Hypertext / explorative Navigation:

> Hier werden die Assoziationen als Links umgesetzt, die inhaltliche Beziehungen zwischen Lernobjekten modellieren.

Guided Tour.

> [BoSc00] definiert *Guided Tour* als ein Navigationsmittel, das speziell für Hypertexte entwickelt wurde. Sie bietet dem Nutzer graphische Übersichten oder bereits vorgefertigte Pfade durch den ansonsten möglicherweise sehr unübersichtlichen Hypertext. Eine Guided Tour in diesem Sinne lässt sich als eine Linearisierung von Modulen unter einer bestimmten Nutzung (vgl. Abbildung 4-17) definieren[53].

Navigation entsprechend der hierarchischen Struktur:

> Grundlage dieser Art der Navigation ist die Struktur, in der die Module und Atome angeordnet sind. Vernachlässigt man die bereits definierten Assoziationen, so hat ein Kurs immer eine Baumstruktur: Die inneren Knoten sind die Module, die Blätter werden durch Atome (gekapselt durch Module) gebildet. Bei einer Navigation entsprechend dieser Struktur werden die Kompositionen zwischen den Lernobjekten direkt in Links abgebildet.

---

[52] Mehrere Style Sheets [CSS] je Präsentationsformat sind möglich. Beispielsweise können verschiedene Layouts über mehrere Style Sheets realisiert werden.

[53] Diese Linearisierung der Module kann beispielsweise mittels eines Durchlaufs durch den Kursbaum entlang der Kompositionen erreicht werden. Sinnvollerweise wird hierfür Tiefensuche verwendet.

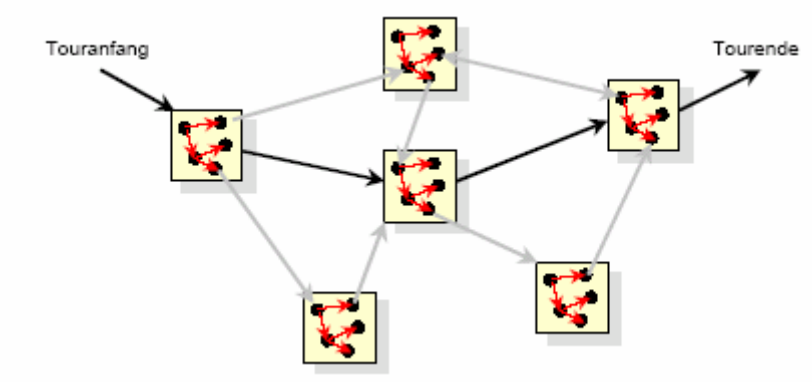


Abbildung 4-17: Beispiel für eine Guided Tour zwischen Lernobjekten [Brü98]

4.3.3.3 Präsentation eines adaptierbaren Kurses in einer Lehrveranstaltung

Der Kurs-Presenter baut die Kurse, die er in seinen Lehrveranstaltungen nutzt, aus Kursrahmen, die Kursdesigner liefern, auf. In einem ersten Schritt muss der Kurs-Presenter einen passenden Kursrahmen auswählen. Dies geschieht anhand des Themas und der Parameter der Nutzungsklasse „Lehre“ des Kursrahmens. Falls keine passenden Kursrahmen zur Verfügung stehen, stellt der Kurs-Presenter eine entsprechende Anforderung an die Kursdesigner. Der ausgewählte Kursrahmen ist dann (falls noch nicht vom Kursdesigner erledigt) entsprechend der geplanten Anwendung – also hinsichtlich Zielgruppe, Präsentationsform und Navigation – zu parametrisieren. Dann erst hat der Kurs seine endgültige Form und kann in der Lehrveranstaltung eingesetzt werden.

Beteiligte Rollen

- Kurs-Presenter
- Falls kein passender Kursrahmen existiert: Kursdesigner

Eingangsdaten

- Kursrahmen, der in Inhalt und Struktur parametrisiert werden kann.

Ergebnis

- Konkreter Kurs, der Inhalt enthält und korrekt bzgl. seiner Nutzung, Präsentation und Navigation in der Lehrveranstaltung parametrisiert ist. Dieser Kurs wird in der Lehrveranstaltung vom Kurs-Presenter eingesetzt.

Detaillierte Beschreibung der Aktivitäten

In diesem Teilprozess werden generische Kursrahmen, die bereits teilweise parametrisiert sind, ausgewählt, an ihre endgültige Nutzung angepasst und anschließend in Lehrveranstaltungen präsentiert. Der Kursrahmen, den der Kurs-Presenter auswählt, ist bereits mit Platzhaltern versehen. Nach der Adaption des Kursrahmens (Aktivität „Adaptieren des Kursrahmens“) können den Platzhaltern konkrete Lernobjekte zugeordnet werden. Abbildung 4-18 zeigt die nötigen Schritte für diesen Teilprozess im Überblick.

Aktivität: Anforderungsanalyse

Wie bei den anderen Teilprozessen auch, ist der erste Schritt eine Anforderungsanalyse. Die Anforderungen, die der Kurs-Presenter an einen fertigen Kurs hat, müssen klar dargestellt werden, so dass der Kurs-Presenter einen passenden Kursrahmen identifizieren kann. Grundlage für die Anforderungsanalyse ist die Lehrform, das Vorwissen der Ziel-

gruppe sowie das Wissen, das die Lerner nach erfolgreicher Teilnahme am Kurs haben sollen (Lehrziel)[54].

Das Ergebnis dieser Aktivität ist die exakte thematische, strukturelle und didaktische Vorstellung des zu präsentierenden Kurses in Form von Parametern und Constraints. Dabei wird die Zielgruppe und die Form der Lehrveranstaltung berücksichtigt.

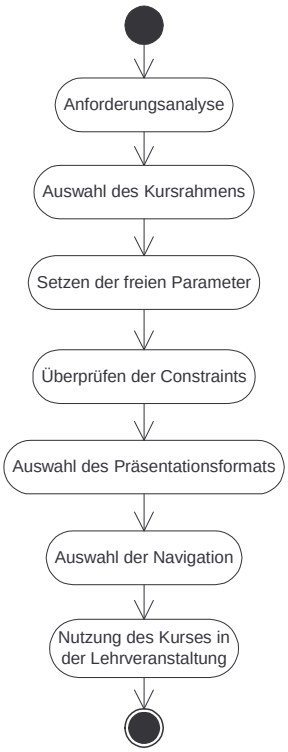


Abbildung 4-18: Präsentation eines adaptierbaren Kurses

Aktivität: Auswahl eines Kursrahmens

Der Kurs-Presenter wählt aus den verfügbaren Kursrahmen zum gegebenen Thema einen Kursrahmen aus, der ihm für seine Lehrveranstaltung auf der Grundlage der Anforderungsanalyse als passend erscheint. Ein Kriterium für die Auswahl ist hierbei die didaktische Ausgestaltung des Kursrahmens, also ob er eher den Charakter einer Vorlesungsmitschrift hat oder ob er mit vielerlei Beispielen, Übungen und Aufgaben mehr Aktivität von den Lernern einfordert. Dies kann entsprechend der Parameter der Nutzungsklasse „Lehre“ entschieden werden.

Ist kein passender Kursrahmen verfügbar, stellt der Kurs-Presenter eine Anforderung an die Kursdesigner, in der er möglichst präzise seinen „Wunschkursrahmen“ anhand von Parametern und Constraints beschreibt. Ab diesem Zeitpunkt steigt man rekursiv in den Prozess des Kursdesigners ein (vgl. die in 4.3.2 beschriebenen Teilprozesse, insbesondere „Design eines Kurses“).

[54] Diese Anforderungen werden entweder durch eine dritte Instanz mittels Lehrplänen oder durch das Lehrziel, das sich der Kurs-Presenter selbst setzt, festgelegt. Der Kurs-Presenter setzt diese Anforderungen dann in Parameter und Constraints um.

Ergebnis dieser Aktivität ist ein Kursrahmen, der für unterschiedliche Nutzungen, repräsentiert durch die Parameterauswahl in Nutzungsklasse „Lehre“, vorbereitet ist und vom Kurs-Presenter noch fertig parametrisiert (adaptiert) werden muss.

Aktivität: Setzen der freien Parameter

Ein adaptierbarer Kurs (der Kursrahmen), wie er vom Kursdesigner geliefert wird, ist bezüglich des Inhalts und der Struktur[55] abgeschlossen. Er wurde allerdings noch nicht entsprechend seines Einsatzes in Lehrveranstaltungen parametrisiert, also ob er beispielsweise in einer Vorlesung, zum Selbststudium oder als Skript verwendet werden soll. Dieser Schritt wird vom Kurs-Presenter durchgeführt, denn er weiß, in welchem didaktischen Umfeld (Vorlesung, Selbststudium, etc.) der Kurs eingesetzt werden soll. Der Kurs-Presenter parametrisiert den Kursrahmen entsprechend der Lehrform (Vorlesung, Übung, Skript für Selbststudium, etc.) durch Setzen der noch nicht vom Kursdesigner gesetzten Parameter der Nutzungsklasse „Lehre“. Erst jetzt ist entschieden, welche Module und Atome konkret angezeigt werden (festgelegt durch Parameter der Nutzungsklasse „Lehre“).

Ergebnis dieser Aktivität ist ein vollständig parametrisierter, an seine Nutzung in der Lehre angepasster Kurs.

Aktivität: Überprüfen der Constraints

Sind alle Parameter des Kursrahmens gesetzt, muss der Kurs-Presenter überprüfen, ob die Parameter alle Constraints erfüllen.

Diese Aktivität hat kein explizit aufführbares Ergebnis, sondern dient der Qualitätssicherung.

Aktivität: Festlegung des Präsentationsformats

In diesem Schritt wird das Layout (Schriftarten, Farben, etc.) des Kurses festgelegt, vgl. auch 4.3.3.2. Der Kurs wird entsprechend der Präsentationsform (Stichpunkte, ausführliche Sätze, Möglichkeit zur Nutzung dynamischer Inhalte wie Animationen, etc.) durch Setzen der Parameter der Nutzungsklasse „Präsentation“ parametrisiert.

Erst nach diesem Schritt hat der Kurs seine endgültige Gestalt: Erst jetzt ist festgelegt, auf welche Weise (Stichpunkte, ausführliche Sätze, Layout, Farbe, Schriftart) die Module und Atome dargestellt werden.

Ergebnis dieser Aktivität ist ein vollständig parametrisierter und an seine Nutzung in der Lehre angepasster Kurs, dessen Darstellung feststeht.

Aktivität: Festlegung der Navigation

In diesem Schritt wird die Navigation (Sequenz von Kapiteln, freie Navigation in einem Hypertext, Guided Tour, etc.) des Kurses festgelegt, vgl. auch 4.3.3.2.

Ergebnis dieser Aktivität ist ein vollständig parametrisierter und an seine Nutzung in der Lehre angepasster Kurs, dessen Navigation feststeht. Dieser Kurs kann nun in einer Lehrveranstaltung genutzt werden.

[55] Jeder Kursrahmen gehört mindestens einer Kategorie an, und diese Kategorien legen die Struktur fest. Somit ist jeder Kursrahmen in seiner Struktur abgeschlossen.

Aktivität: Nutzung des Kurses in der Lehrveranstaltung

Der Kurs wird in der Lehrveranstaltung genutzt. Dies kann – je nach Art der konkreten Gestalt des Kurses – auf vielerlei Art geschehen. Hier eine Auswahl:

- Online-Präsentation mit einem Beamer zur Unterstützung des Vortrags des Kurs-Presenter.
- Ausdruck dieser Online-Präsentation als vorlesungsbegleitendes Skript, das der Kurs-Presenter vor der Vorlesung an die Lerner verteilt.

Beobachtung

- Benötigt der Kurs-Presenter für seine Lehrveranstaltung einen Kursrahmen mit bestimmten Eigenschaften, der bislang noch nicht verfügbar ist, so fordert er einen entsprechenden Kursrahmen an. Die Situation ist hier die gleiche wie wenn ein Kursdesigner noch fehlende Module anfordert: Er muss nicht notwendigerweise seinen Kommunikationspartner kennen. Auf diesen Aspekt wird in 4.5.2.1 und 4.6.2 im Detail eingegangen.
- Um einen geeigneten Kursrahmen auswählen und später adaptieren zu können, benötigt der Kurs-Presenter eine Übersicht über alle verfügbaren Kursrahmen. Die Situation ist wieder sehr ähnlich zum Kursdesigner in den Teilprozessen „Entwickeln eines Kurses“ (vgl. 4.3.2.3) und „Überarbeiten eines existierenden Kurses“ (vgl. 4.3.2.4).

4.3.3.4 Präsentation eines customized Kurses in einer Lehrveranstaltung

Die Präsentation eines customized Kurses in einer Lehrveranstaltung ähnelt der Präsentation eines adaptierbaren Kurses in einer Lehrveranstaltung (vgl. 4.3.3.3) sehr stark. Der Unterschied zwischen beiden Teilprozessen liegt darin, dass nun der Kurs-Presenter nicht einen bereits bestehenden Kursrahmen parametrisiert, sondern einen Kurs völlig neu aufbaut und so Tätigkeiten des Kursdesigners übernimmt. Der Kurs-Presenter kann hierbei beispielsweise einen bereits existierenden Kursrahmen, der vom Kursdesigner zwar noch nicht mit konkreten Inhalten versehen wurde, der aber bzgl. Inhalt, Struktur und didaktischer Funktion bereits teilweise parametrisiert ist, nutzen. (Dieses Szenario ist sehr nahe an der Nutzung eines adaptierbaren Kurses.) Ebenso – und das wird im Fall eines customized Kurses die Regel sein – kann der Kurs-Presenter einen Kurs völlig neu aufbauen (vgl. die Teilprozesse in 4.3.2). Er übernimmt damit sowohl das Design als auch die Entwicklung eines Kurses selbst.

Zuerst wählt der Kurs-Presenter einen Kursrahmen aus bzw. entwirft ihn selbst. Der Kurs-Presenter wählt für diesen Kursrahmen dann die Module aus, die an den Platzhaltern erscheinen sollen. Der so erstellte Kurs wird anschließend in der Lehrveranstaltung genutzt.

Beteiligte Rollen

- Kurs-Presenter
- Optional falls kein passender Kurs(rahmen) existiert: Kursdesigner
- Falls benötigte Module fehlen: Moduldesigner

Eingangsdaten

- Optional: Kursrahmen[56], der keinen Inhalt enthält, aber bzgl. Inhalt, Struktur oder didaktischer Anwendung teilweise parametrisiert ist.

[56] Dieser kann vom Kurs-Presenter auch selbst definiert werden.

Ergebnis

- Konkreter Kurs, der Inhalt enthält und korrekt bzgl. seiner Nutzung in der Lehrveranstaltung parametrisiert ist. Dieser Kurs wird in der Lehrveranstaltung vom Kurs-Presenter eingesetzt.

Detaillierte Beschreibung der Aktivitäten

In diesem Teilprozess werden Kurse neu erstellt. Abbildung 4-19 zeigt die hierfür nötigen Schritte im Überblick.

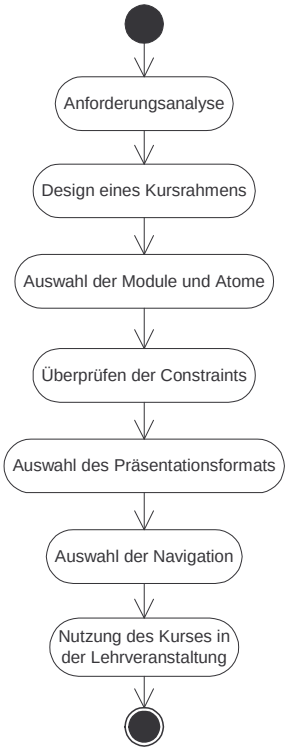


Abbildung 4-19: Präsentation eines customized Kurses

Aktivität: Anforderungsanalyse

Der erste Schritt des Kurs-Presenters ist wieder eine Anforderungsanalyse, in der die Anforderungen, die der Kurs-Presenter an einen Kurs hat, klar dargestellt werden (vgl. 4.3.3.3).

Aktivität: Design eines Kursrahmens

Der Kurs-Presenter entwirft einen Kursrahmen, der seinen Anforderungen aus den Lehrveranstaltungen genügt. Dieser Vorgang läuft analog zum Teilprozess „Design eines neuen Kurses“ (vgl. 4.3.2.2) ab. Dieser Kursrahmen ist dann der Ausgangspunkt für die weiteren Arbeiten (Auswahl passender Module, etc.) des Kurs-Presenters.

Ergebnis dieser Aktivität ist ein Kursrahmen, der vom Kurs-Presenter noch (wo nötig) mit Inhalten versehen und fertig parametrisiert werden muss.

Aktivität: Auswahl der Module und Atome

Der Kurs-Presenter wählt die Module aus, die in den Kursrahmen eingesetzt werden sollen. Maßgeblich sind hierbei sowohl die Constraints, die der Kursrahmen für die in ihn

zu integrierenden Module aufstellt, als auch die Vorstellungen des Kurs-Presenters über die zu vermittelnden Inhalte sowie deren Präsentation.

Ergebnis dieser Aktivität ist ein Kurs, der bzgl. Inhalt, Struktur und didaktischer Funktion abgeschlossen ist und bereits in Lehrveranstaltungen eingesetzt werden könnte.

Aktivität: Überprüfen der Constraints

Stehen alle Inhalte für einen Kurs bereit, muss der Kurs-Presenter überprüfen, ob die ausgewählten Module alle Constraints, die auf den Platzhaltern des Kurses definiert sind, erfüllen. Es findet hier ein Abgleich der Parameter des Kurses mit den Parametern der in den Kurs einzusetzenden Module statt. Dieser Schritt ist notwendig, um die Stimmigkeit des Kurses sicherzustellen.

Diese Aktivität hat kein explizit aufführbares Ergebnis, sondern dient der Qualitätssicherung.

Aktivität: Festlegung der Präsentation

In diesem Schritt wird das Präsentationsformat und das Layout des Kurses festgelegt, vgl. auch 4.3.3.2.

Ergebnis dieser Aktivität ist ein vollständig parametrisierter und an seine geplante Nutzung in der Lehre angepasster Kurs, dessen Präsentationsformat feststeht.

Aktivität: Festlegung der Navigation

In diesem Schritt wird die Navigation des Kurses festgelegt, vgl. auch 4.3.3.2.

Ergebnis dieser Aktivität ist ein vollständig parametrisierter und an seine Zielgruppe angepasster Kurs, dessen Navigation feststeht. Dieser Kurs kann nun in einer Lehrveranstaltung genutzt werden.

Aktivität: Nutzung des Kurses in der Lehrveranstaltung

Der Kurs wird in der Lehrveranstaltung genutzt.

Beobachtung

- Füllt ein Kursdesigner einen Kursrahmen mit Modulen, muss er peinlichst darauf achten, alle Constraints einzuhalten. Andernfalls ist die Gefahr groß, dass der Kurs in sich unstimmig und damit nicht nutzbar wird. Entwirft der Kurs-Presenter hingegen seinen Kurs selbst, wählt er also selbst die Module aus, die er in seiner Lehrveranstaltung nutzen will, so muss die Kontrolle der Constraints nicht so genau durchgeführt werden, da der Kurs-Presenter selbst den Abgleich zwischen der Definition und der Auswertung der Parameter und Constraints macht. Statt ausschließlich eine Kontrolle der Constraints durchzuführen, prüft der Kurs-Presenter vor dem Verbau eines Moduls, ob dieses seinen Anforderungen entspricht, und verbaut es nur dann, wenn es in seinen Kursrahmen passt. Dadurch, dass der Kurs-Presenter eine klare Vorstellung von seiner Lehrveranstaltung und den zu verwendenden Lehrmaterialien hat, ist sichergestellt, dass die so entstehenden Kurse ihre didaktische Funktion erfüllen können.

4.3.4 Unterstützung von Lehrveranstaltungen und Betreuung der Lernenden

Überblick

Aufgabe des Kurstutors ist die Unterstützung des Kurs-Presenters bei dessen Lehrveranstaltung. Während der Kurs-Presenter vor allem in der Lehrveranstaltung sowie deren Vor- und

Nachbereitung tätig ist, ist der Kurstutor in Tutorübungen[57] und nach der Lehrveranstaltung aktiv. In dieser Zeit betreut er die Lerner, beantwortet Fragen zur Lehrveranstaltung und hilft den Lernern bei der Bearbeitung und Lösung von Problemen und Aufgaben[58].

Der Kurstutor wird bei einer Anfrage der Lerner während des (Selbst-)Lernens aktiv. Bei der Erklärung und Lösung der Anfrage hat der Kurstutor viele Freiheitsgrade, beispielsweise die Geschwindigkeit der Interaktion mit den Lernenden, die Reihenfolge, in der bestimmte, zur Problemlösung gehörende Aspekte erläutert werden, der Detaillierungsgrad der Antwort, etc.

Beteiligte Rollen

- Kurstutor
- Lerner

Eingangsdaten

- Konkreter Kurs, wie er vom Kurs-Presenter in einer Lehrveranstaltung[59] eingesetzt bzw. für Tutorübungen vorbereitet wird. Der Kurs enthält Inhalt und ist korrekt parametrisiert.

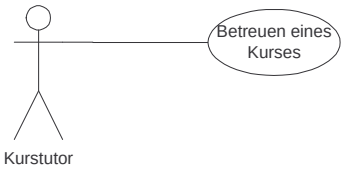


Abbildung 4-20: Aufgaben des Kurstutors

Ergebnis

- Dieser Teilprozess hat kein Ergebnis. Vielmehr dient dieser Teilprozess der Unterstützung der Teilprozesse „Präsentation eines adaptierbaren Kurses in einer Lehrveranstaltung“ und „Präsentation eines customized Kurses in einer Lehrveranstaltung“.

Detaillierte Beschreibung der Aktivitäten

Dieser Teilprozess hat keine klar abgrenzbaren Aktivitäten, sondern er funktioniert meistens ereignisgesteuert: Der Kurstutor nimmt Fragen und Probleme von den Lernern entgegen und hilft ihnen bei der Lösung. Hierbei nutzt er den Kurs, der in der zugehörigen Lehrveranstaltung vom Kurs-Presenter eingesetzt wird.

Beobachtung

- Zwischen den Tätigkeiten des Kurs-Presenters und des Kurstutors gibt es starke Abhängigkeiten. In Tutorübungen füllt ein Dozent sogar wechselseitig beide Rollen aus.

---

[57] Eine Tutorübung ist eine kursbegleitende Zusatzlehrveranstaltung, die als Präsenzveranstaltung stattfindet. Meist werden Übungsaufgaben unter der Anleitung des Kurstutors bearbeitet. In einer Tutorübung erfolgt eine deutlich intensivere Interaktion und Kommunikation zwischen dem Kurstutor und den Lernern.

[58] In Tutorübungen, wie sie für viele Vorlesungen angeboten werden, nimmt der Dozent, der die Tutorübung betreut, üblicherweise zwei Rollen ein: Während der Tutorübung übernimmt er die Rolle des Kurs-Presenters, der einen adaptierten Kurs (die Übungsaufgaben) vorträgt. Bei der Bearbeitung der Aufgaben durch die Lerner während der Tutorübung und in der Zeit zwischen den Tutorübungen übernimmt er die Rolle des Kurstutors, der die Lerner bei den Hausaufgaben oder bei sonstigen Fragen und Problemen unterstützt. Die Rolle des Kurstutors hat hinsichtlich der Wiederverwendbarkeit der Lehrmaterialien keine Bedeutung.

[59] In der Regel verwendet der Kurstutor die selben Kursmaterialien wie der Kurs-Presenter. Der Kurstutor braucht im allgemeinen nicht selbst den Kurs adaptieren oder aufbauen.

- Zwischen dem Kurstutor und dem Kurs-Presenter besteht eine enge Zusammenarbeit, damit der Kurstutor den Lernern die Inhalte des Kurses so vermittelt, wie dies im Sinne des Kurs-Presenters ist. Die hier stattfindende Kommunikation kann auf verschiedenen Wegen erfolgen; meist werden jedoch persönliche Treffen stattfinden, die durch Nachfragen mittels unterschiedlicher Kommunikationswerkzeuge (Telefon, E-Mail) ergänzt werden.
- Im Vergleich zum Kurs-Presenter interagiert der Kurstutor wesentlich mehr mit den Lernern. Dies erfordert einerseits eine tragfähige Kommunikationsinfrastruktur, um den Lernern synchrone als auch asynchrone Unterstützung bieten zu können, und andererseits eine Möglichkeit, auch die Kurse in die Kommunikation mit einbeziehen zu können, beispielsweise um auf relevante Textstellen hinzuweisen oder um bestimmte Abschnitte Schritt für Schritt zu erklären.

4.3.5 Lernen mit Kursen

Als Lerner werden hier diejenigen bezeichnet, die sich Wissen im Selbststudium oder in Lehrveranstaltungen aneignen. Sie arbeiten in der Regel mit Kursen (vgl. Abbildung 4-21).

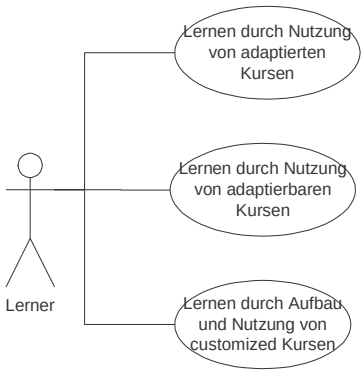


Abbildung 4-21: Aufgaben des Lerners

4.3.5.1 Lernen durch Nutzung von adaptierten Kursen

Die Begleitung und Nachbearbeitung von Lehrveranstaltungen oder das Selbststudium von Lehrinhalten erfolgt mittels vorgefertigter, adaptierter Kurse. Der Lerner erhält vom Kursdesigner oder vom Kurs-Presenter bereits fertige oder stark angepasste Kurse in der Lehrveranstaltung ausgehändigt bzw. einen Hinweis, von wo er diese Kurse beziehen kann, die er nutzt, um sich die Lehrinhalte anzueignen. Diese Kurse müssen nicht notwendigerweise identisch mit denen sein, die in der Lehrveranstaltung genutzt werden: Die Kursvariante, die für die Lehrveranstaltung verwendet wird, basiert auf dem selben Kurs wie die Kursvariante, die für das Selbststudium zur Verfügung gestellt wird, aber während die Kursvariante für die Lehrveranstaltung beispielsweise nur grobe Stichpunkte enthält, ist die Kursvariante für das Selbststudium sehr ausführlich und mit Bildern, Skizzen und Animationen versehen.

Beteiligte Rollen

- Lerner

Eingangsdaten

- Kurs, der konkrete Module enthält und korrekt parametrisiert ist. Dieser Kurs wird den Lernern vom Kursdesigner bzw. Kurs-Presenter zum Selbststudium zur Verfügung gestellt.

Ergebnis

- Dieser Teilprozess hat kein Ergebnis. Vielmehr dient dieser Teilprozess den Lernern der Aneignung von Wissen.

Detaillierte Beschreibung der Aktivitäten

Dieser Teilprozess hat keine klar abgrenzbaren Aktivitäten.

Beobachtung

- Die Verteilung der Lehrmaterialien an die Lerner kann auf sehr unterschiedliche Weise geschehen: In der Lehrveranstaltung durch den Kurs-Presenter oder einen Kurstutor, als Skripten über die Fachschaften oder über andere Lerner, aber auch online über die Web-Seiten der Lehrveranstaltung. Dies unterscheidet diesen Teilprozess von den Teilprozessen der Lehrmaterialerstellung, wo der Datenaustausch nahezu ausschließlich mittels elektronischer Medien (CDs, DVDs, Disketten, Internet) erfolgt.

4.3.5.2 Lernen durch Nutzung von adaptierbaren Kursen

Der Lerner nutzt die Kurse, wie sie die Kursdesigner erstellen, in seinem Selbststudium. In einem ersten Schritt muss der Lerner einen passenden Kursrahmen auswählen. Dies geschieht anhand des Themas und der Parameter der Nutzungsklasse „Lernen“ des Kursrahmens. Der ausgewählte Kursrahmen ist dann (falls noch nicht vom Kursdesigner erledigt) entsprechend der geplanten Anwendung – also hinsichtlich Wissensstand, Lernziel, Präsentationsform und Navigation – zu parametrisieren. Dann erst hat der Kurs seine endgültige Form und kann im Selbststudium eingesetzt werden (vgl. 4.3.3.3).

Beteiligte Rollen

- Lerner
- Falls kein passender Kursrahmen existiert: Kursdesigner

Eingangsdaten

- Kursrahmen, der mit vorgegebenen Modulen versehen und bereits in Inhalt und Struktur korrekt parametrisiert ist.

Ergebnis

- Konkreter Kurs, der Inhalt entsprechend den Vorstellungen des Lerners enthält und korrekt bzgl. seiner Nutzung im Selbststudium parametrisiert ist. Dieser Kurs wird vom Lerner für das Selbststudium genutzt.

Detaillierte Beschreibung der Aktivitäten

In diesem Teilprozess wird ein Kursrahmen entsprechend dem Lernziel und Vorwissen des Lerners parametrisiert und dann für das Selbststudium genutzt. Abbildung 4-22 zeigt die hierfür nötigen Schritte im Überblick.

Aktivität: Anforderungsanalyse

Der erste Schritt des Lerners ist eine Anforderungsanalyse, in der seine Anforderungen an den fertigen Kurs klar dargestellt werden. Die Anforderungen können auch vom Kurs-Presenter oder vom Kurstutor kommen, beispielsweise wenn ein Kurs im Rahmen einer Lehrveranstaltung erstellt werden soll. Des weiteren muss der Lerner den Kursrahmen, den er vom Kursdesigner erhalten hat, soweit verstehen, dass er weiß, welchen Einschränkungen (modelliert durch Constraints) die von ihm zu setzenden Parameter unter-

liegen. Die Anforderungsanalyse bildet die Grundlage für die spätere Parametrisierung des Kurses.

Das Ergebnis dieser Aktivität ist die exakte thematische, strukturelle und didaktische Vorstellung des zu erstellenden Kurses unter Berücksichtigung des Vorwissens des Lerners.

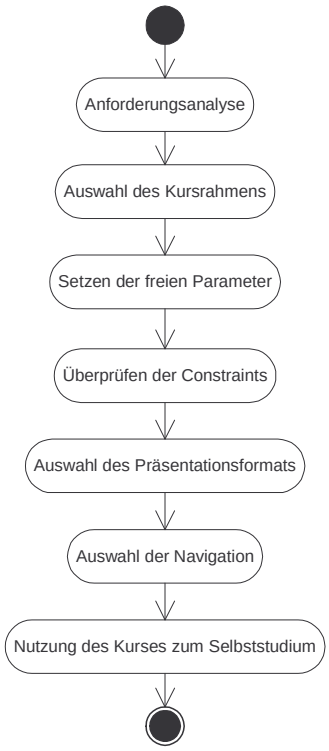


Abbildung 4-22: Aufbau eines adaptierbaren Kurses für das Selbststudium

Aktivität: Auswahl eines Kursrahmens

Der Lerner wählt aus den verfügbaren Kursrahmen zum gegebenen Thema einen Kursrahmen aus. Es ist möglich, dass dem Lerner aus didaktischen Gründen lediglich eine kleine Auswahl von Kursrahmen oder direkt ein konkreter Kursrahmen vom Kurs-Presenter vorgegeben wird.

Ergebnis dieser Aktivität ist ein Kursrahmen, der für unterschiedliche Nutzungen, repräsentiert durch die Parameterauswahl in Nutzungsklasse „Lernen“, vorbereitet ist und vom Lerner noch fertig parametrisiert (adaptiert) werden muss.

Aktivität: Setzen der freien Parameter

Der Lerner setzt die im Kursrahmen noch nicht belegten Parameter entsprechend seinen Anforderungen.

Ergebnis dieser Aktivität ist ein hinsichtlich Inhalt, Struktur und Nutzung fertig parametrisierter Kursrahmen.

Aktivität: Überprüfen der Constraints

Sind alle Parameter des Kursrahmens gesetzt, muss der Lerner überprüfen, ob die Parameter alle Constraints erfüllen. Es muss davon ausgegangen werden, dass die Lerner mit der Syntax und der Semantik komplexer Constraints nicht vertraut sind. Daher sollte die

Prüfung der Constraints automatisch durch die vom Lerner verwendeten Authoring-Tools oder Lernsysteme beim Setzen der Parameter in der Aktivität „Setzen der freien Parameter“ durchgeführt werden.

Diese Aktivität hat kein explizit aufführbares Ergebnis, sondern dient der Qualitätssicherung.

Aktivität: Festlegung des Präsentationsformats

In diesem Schritt wird das Präsentationsformat des Kurses festgelegt, vgl. auch 4.3.3.2.

Ergebnis dieser Aktivität ist ein vollständig parametrisierter und an die Vorstellungen des Lerners hinsichtlich der Präsentation angepasster Kurs.

Aktivität: Festlegung der Navigation

In diesem Schritt wird die Navigation des Kurses festgelegt, vgl. auch 4.3.3.2.

Ergebnis dieser Aktivität ist ein vollständig parametrisierter und an den Lerner angepasster Kurs, dessen Navigation feststeht. Dieser Kurs kann nun in vom Lerner für das Selbststudium genutzt werden.

Aktivität: Nutzung des Kurses im Selbststudium

Der Kurs wird vom Lerner im Selbststudium genutzt.

Beobachtung

- Benötigt der Lerner für sein Selbststudium einen Kurs mit bestimmten Eigenschaften, der bislang noch nicht verfügbar ist, so fordert er einen entsprechenden adaptierten Kurs oder adaptierbaren Kursrahmen an. Die Situation ist hier die gleiche wie wenn ein Kursdesigner noch fehlende Module anfordert: Er muss nicht notwendigerweise seinen Kommunikationspartner kennen. Auf diesen Aspekt wird in 4.5.2.1 und 4.6.2 im Detail eingegangen.
- Um einen geeigneten Kurs auswählen (adaptieren) zu können, benötigt der Lerner eine Übersicht über alle verfügbaren Kurse, Kursrahmen und Kursvarianten (vgl. 4.3.3.3).

4.3.5.3 Lernen durch Aufbau und Nutzung von customized Kursen

Das Lernen durch Aufbau und Nutzung von customized Kursen ist analog zum Präsentieren eines customized Kurses (vgl. 4.3.3.4) zu verstehen: Hier stellt sich der Lerner – eventuell innerhalb von vom Kursdesigner oder vom Kurs-Presenter vorgegebenen Grenzen – selbst seinen Kurs zusammen und nutzt ihn zum Lernen, so dass nun der Lerner nicht einen bereits bestehenden Kurs parametrisiert, sondern einen Kurs (in Teilen) völlig neu aufbaut und so Tätigkeiten des Kursdesigners übernimmt. Der Lerner kann hierbei einen bereits existierenden Kursrahmen, der vom Kursdesigner zwar noch nicht mit konkreten Inhalten versehen wurde, der aber bzgl. Inhalt, Struktur und didaktischer Funktion bereits teilweise parametrisiert ist, nutzen. Der Lerner wählt für diesen Kursrahmen dann die Module aus, die an den Platzhaltern erscheinen sollen. Der so erstellte Kurs wird anschließend im Selbststudium genutzt. Ebenso ist es aber auch möglich, dass der Lerner bereits den Kursrahmen (mit Unterstützung durch Kurs-Presenter oder Kurstutor) selbst erstellt.

Beim Aufbau des Kurses geht der Lerner sehr ähnlich wie der Kursdesigner vor: Er kombiniert Module zu Kursen, wobei er vom Kursdesigner vorgegebene Parameter und Constraints beachtet. Im Unterschied zum Kursdesigner nutzt der Lerner einen bereits vom Kurs-Presenter vorbereiteten Kursrahmen aus Ausgangspunkt für die Erstellung des eigenen Kurses: Der vorgegebene Kursrahmen hat bereits viele Parameter vorgegeben, so dass der Lerner bei der

Gestaltung seines eigenen Kurses anhand der Vorgaben des Kurs-Presenters entsprechend seinen Anforderungen geführt wird. Der Lerner muss sich hierbei aktiv mit dem Lehrstoff und den verfügbaren Kurseinheiten (Modulen) auseinandersetzen und die Kurseinheiten zu einem sinnvollen Ganzen, eben dem „eigenen“ Kurs, kombinieren. Anschließend nutzt er den Kurs zum Selbststudium oder stellt ihn anderen Lernern zur Verfügung.

Beteiligte Rollen

- Lerner
- Optional, falls kein passender Kursrahmen existiert: Kursdesigner
- Kurs-Presenter und Kurstutor
- Falls benötigte Module, fehlen: Moduldesigner[60]

Eingangsdaten

- Thema des Kurses.
- Optional: Kursrahmen, der in Teilen bereits mit vorgegebenen Modulen und korrekten Parametern versehen ist. Dieser Kursrahmen ist die Grundlage, auf der der Lerner seinen eigenen Kurs aufbaut.

Ergebnis

- Konkreter Kurs, der Inhalt enthält und korrekt bzgl. seiner Nutzung für den Lerner parametrisiert ist. Dieser Kurs wird für das Selbststudium genutzt und evtl. an andere Lerner weitergegeben.

Detaillierte Beschreibung der Aktivitäten

Für den Aufbau eines eigenen Kurses führt der Lerner im wesentlichen die gleichen Schritte aus wie der Kurs-Presenter, wenn er einen customized Kurs entwirft (vgl. 4.3.3.4). Abbildung 4-23 stellt die wesentlichen Aktivitäten dar.

Aktivität: Anforderungsanalyse

Analog Teilprozess „Präsentation eines customized Kurses in einer Lehrveranstaltung“ (vgl. 4.3.3.4).

Das Ergebnis dieser Aktivität ist die exakte inhaltliche und strukturelle Vorstellung des zu entwickelnden Kurses, eventuell unter Berücksichtigung der vom Lerner bevorzugten und vorgesehenen Nutzung sowie den Anforderungen des Kurs-Presenters bzw. des Kurstutors.

Aktivität: Verstehen des vorgegebenen Kursrahmens

Der Kursdesigner, Kurs-Presenter bzw. der Kurstutor gibt dem Lerner einen Kursrahmen vor, der bereits zum großen Teil parametrisiert und mit Modulen versehen sein kann. Dieser Kursrahmen bildet den Rahmen, innerhalb dem der Lerner seinen Kurs gestalten darf. Bevor der Lerner sich an die Entwicklung seines Kurses machen kann, muss er diesen Rahmen verstehen.

[60] Der Kurs-Presenter bzw. der Kurstutor sollte darauf achten, dass die Kurse, die er den Lernern als Basis für deren eigene Kurse bereitstellt, keine noch nicht existierenden Module benötigen, um die Moduldesigner nicht zu sehr zu belasten. Eine Möglichkeit hierzu wäre, dass das System den Lernern für die zu belegenden Platzhalter im Kurs eine Auswahl an Modulen bereitstellt und keine völlig freie Wahl erlaubt.

Ergebnis dieser Aktivität ist ein Kursrahmen, der vom Lerner noch (wo nötig) mit Inhalten versehen und fertig parametrisiert werden muss.

Diese Aktivität wird oft verschränkt mit der Aktivität „Anforderungsanalyse“ durchgeführt.

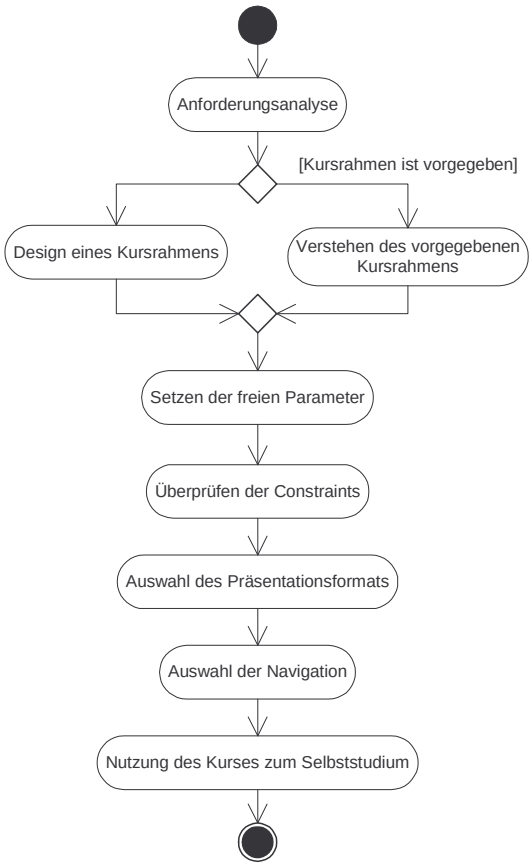


Abbildung 4-23: Lernen mit einem customized Kurs

Aktivität: Design eines Kursrahmens

Diese Aktivität erfolgt analog zum Teilprozess „Design eines neuen Kurses“, vgl. 4.3.2.2.

Ergebnis dieser Aktivität ist ein Kursrahmen, der vom Kurs-Presenter noch (wo nötig) mit Inhalten versehen und fertig parametrisiert werden muss.

Aktivität: Auswahl der Module

Der Lerner wählt die Module aus, die in den Kursrahmen in die noch nicht vom Kurs-Presenter oder Kurstutor belegten Platzhalter eingesetzt werden sollen. Maßgeblich sind hierbei sowohl die Constraints, die der Kursrahmen für die in ihn zu integrierenden Module aufstellt, als auch die Vorstellungen des Lerners über die zu vermittelnden Inhalte sowie deren Präsentation.

Ergebnis dieser Aktivität ist ein Kurs, der bzgl. Inhalt, Struktur und Nutzung beim Lernen abgeschlossen ist.

Aktivität: Überprüfen der Constraints

Analog zu 4.3.3.4 dient dies der Kontrolle, ob der vom Lerner entworfene Kurs auch den Anforderungen des Kursdesigners, des Kurs-Presenters bzw. des Kurstutors entspricht.

Diese Aktivität hat kein explizit aufführbares Ergebnis, sondern dient der Qualitätssicherung.

Aktivität: Auswahl des Präsentationsformats

In diesem Schritt wird die Präsentation des Kurses festgelegt, vgl. auch 4.3.3.2.

Ergebnis dieser Aktivität ist ein vollständig parametrisierter und an seine geplante Nutzung im Selbststudium angepasster Kurs, dessen Präsentation feststeht.

Aktivität: Auswahl der Navigation

In diesem Schritt wird die Navigation des Kurses festgelegt, vgl. auch 4.3.3.2.

Ergebnis dieser Aktivität ist ein vollständig parametrisierter und an die beabsichtigte Nutzung angepasster Kurs, dessen Navigation feststeht.

Aktivität: Nutzung des Kurses zum Selbststudium

Der Kurs wird vom Lerner im Selbststudium genutzt.

Beobachtung

- Lerner sollten sich im Idealfall mit Parametern, insbesondere deren formaler Beschreibung und Syntax gar nicht unmittelbar befassen müssen. Statt dessen sollte die komplizierte Syntax der Parameter und Constraints hinter einfach und intuitiv zu bedienenden Benutzeroberflächen in den Programmen, die die Lerner zum Aufbau ihres Kurses verwenden, verborgen werden. Die Lerner sollten beim Aufbau eines customized Kurses direkt mit Modulen arbeiten und ähnlich einem Baukasten verschiedene Lernobjekte zu einem Kurs kombinieren können. Die Lerner sollen sich auf die Inhalte der Lernobjekte und ihre Kombination konzentrieren können und nicht Details des den Lehrmaterialien zugrundeliegenden Modells beherrschen müssen.
- Es ist möglich, dass ein System ausgehend von dem vom Kurs-Presenter vorbereiteten und parametrisierten Kursrahmen und dem Nutzerprofil des Lerners automatisch einen individuellen, adaptierten Kurs erstellt. Dabei könnte dann auch berücksichtigt werden, welche Kurse der Lerner bereits erfolgreich absolviert hat und welches Wissen darüber hinaus in seinem Nutzerprofil verzeichnet ist. Dies hätte einen Kurs zum Ergebnis, der ideal an den Lerner angepasst ist. Hierbei würde allerdings die Beschäftigung des Lerners mit den zur Verfügung stehenden Modulen entfallen, die nötig ist, um passende Module für den Kurs auszuwählen (Fall 4.3.5.1). Die Auswahl der Module hat eine wichtige didaktische Funktion, und es muss genau geprüft werden, ob der schnellen Erstellung individueller Kurse dem Wissenserwerb durch die Beschäftigung mit den Modulen Vorrang gegeben werden soll.
- Der Kursrahmen muss vom Kursdesigner, Kurs-Presenter oder dem Kurstutor so parametrisiert sein, dass einerseits durch die Parameter und deren Werte eine gute Leitung der Lerner stattfindet, so dass die Lerner beim Aufbau ihres eigenen Kurses nicht Gefahr laufen, sich zu verzetteln, und andererseits die Lerner noch genügend Freiheiten bei der Auswahl der Module haben.

- Dem Lerner muss eine Übersicht über einsetzbare Module gegeben werden. Dies muss nicht notwendigerweise eine Übersicht über alle derzeit verfügbaren Lernobjekte sein; eine sinnvolle Auswahl, die der Kurs-Presenter vorgenommen hat, ist in vielen Fällen aus didaktischen Gründen einer kompletten Übersicht vorzuziehen.
- Analog zum Kurs-Presenter kann der Lerner für sein Selbststudium einen entsprechenden Kursrahmen anfordern. Benötigt der Lerner bestimmte Module, könnte er eine entsprechende Anfrage an die Moduldesigner stellen. Dies ist problematisch, da der direkte, ungefilterte Durchgriff möglicherweise Tausender von Lernern auf die Moduldesigner deren Arbeitslast zu stark erhöhen würde. Es ist zielführender, wenn der Kurs-Presenter oder der Kurstutor den Lernern eine Auswahl von Modulen vorgibt. Auch hier muss wieder eine Kommunikation zwischen Prozessteilnehmern stattfinden, die sich nicht notwendigerweise kennen.
- Um einen geeigneten Kursrahmen definieren zu können, benötigt der Lerner wieder eine Übersicht über alle verfügbaren Kurse bzw. Kursrahmen; für die Befüllung dieses Kursrahmens mit passenden Modulen ist des weiteren eine Übersicht über alle verfügbaren Module nötig (vgl. 4.3.3.4).

4.4 Intergration von Prozess und Datenmodell

Eine wesentliche Forderung ist die Integration von Prozess und Datenmodell. Dies muss näher erläutert werden. Integration meint hier zweierlei: Einerseits Integration und Durchgängigkeit der Erstellung und Nutzung von Lehrmaterialien und andererseits enge Abstimmung von Prozess und Datenmodell.

4.4.1 Durchgängigkeit des Prozesses der Lehrmaterialerstellung und -nutzung

Die Forderung nach Durchgängigkeit von Erstellung und Nutzung von Lehrmaterialien fußt auf der heute vorherrschenden Trennung von Erstellung der Lehrmaterialien auf der einen Seite und deren Nutzung in Lehrveranstaltungen auf der anderen Seite. Beide Tätigkeiten werden bislang nicht als Einheit gesehen, wodurch es schwierig wird, Anforderungen an Lehrmaterialien, die erst im Lauf der Nutzung auftauchen, in den Lehrmaterialien zu berücksichtigen. Der Prozess, wie er in diesem Kapitel beschrieben ist, bietet bereits ein Stück weit Durchgängigkeit in diesem Sinne und verknüpft, gesteuert durch die Prozessphasen, die Nutzung von Lehrmaterialien eng mit deren Erstellung, hat aber noch Defizite. Diese werden in 4.6 noch genauer beleuchtet und im Fortgang der Arbeit gelöst.

4.4.2 Durchgängigkeit des Datenmodells und Abstimmung mit dem Prozess

Grundlage der Durchgängigkeit des Prozesses ist das Datenmodell (vgl. Kapitel 3), das in allen Phasen des Prozesses genutzt wird. Es gibt beim Übergang von Lehrmaterialerstellung zu Lehrmaterialnutzung sowie beim Übergang von einem Teilprozess zum nächsten keinen Bruch bei der Behandlung der Daten. Diese Einheitlichkeit und die damit erreichte Durchgängigkeit des Datenmodells ermöglicht einen in allen Prozessphasen transparenten Informations- und Datenaustausch und so erst die Durchgängigkeit des Prozesses, wie sie im letzten Absatz beschrieben wurde. Zusätzlich wird die Kommunikation zwischen den Prozessbeteiligten erleichtert, da durch das gemeinsame Datenmodell auch ein gemeinsames Verständnis der zu bearbeitenden Elemente (Kurse, Module, Atome) und verwendeten Begriffe (Inhalt, Struktur, Parameter, Constrains, etc.) erreicht wird.

Da in allen Prozessphasen immer auf den selben Daten gearbeitet wird, werden Verbesserungen der Kurse und Lernobjekte unmittelbar in den Datenbestand eingepflegt und stehen so-

fort allen Prozessbeteiligten zur Verfügung. Dies erhöht die Qualität und die Möglichkeit der Wiederverwendung der Lehrmaterialien unmittelbar.

4.5 Prozessbeobachtungen

Der Prozess der Lehrmaterialerstellung ist seiner Natur nach meist ein (räumlich und zeitlich) verteilter, nicht zentral organisierter und hochgradig arbeitsteiliger Prozess. Die einzelnen Prozessteilnehmer arbeiten an unterschiedlichen Orten für unterschiedliche Organisationen und Unternehmen; eine zentrale Koordination existiert in der Regel nicht. Dies hat wesentliche Auswirkungen auf die Prozesssteuerung und -durchführung und die Datenhaltung.

4.5.1 Datenhaltung

Die Prozessteilnehmer arbeiten für verschiedene Unternehmen und Organisationen. Es muss daher davon ausgegangen werden, dass eine gemeinsame Datenhaltung, beispielsweise in einem Shared Workspace oder einer zentralen Datenbank, nicht möglich ist. Im akademischen Umfeld wäre dies noch relativ einfach machbar. Sobald aber Interessen von Unternehmen, die Lehrmaterialien erstellen und bereitstellen, berücksichtigt werden müssen, kommt man ohne ein ausgefeiltes Lizenz- und Rechtesystem für Lehrmaterialien nicht mehr aus. Diese Problematik soll in dieser Arbeit allerdings nicht weiter vertieft werden.

Im folgenden wird davon ausgegangen, dass die Lernobjekte und Kurse an verschiedenen Orten zur Verfügung stehen und alle über das Internet zugänglich sind. Die Lernobjekte selbst sind jedoch immer als Ganzes verfügbar und nicht in verschiedenen Fragmenten an unterschiedlichen Orten gespeichert. Des weiteren wird davon ausgegangen, dass eine komplette Übersicht über alle aktuell verfügbaren Lernobjekte an einer wohlbekannten Stelle (beispielsweise einem Lernobjekt-Service) abrufbar ist. Diese Übersicht könnte beispielsweise realisiert werden, indem jeder Anbieter von Lehrmaterialien die Kurse und Lernobjekte, die er anbietet, bei dieser Stelle registriert, die sie dann – evtl. erst nach einer Qualitätskontrolle – in die Übersicht aufnimmt.

4.5.2 Prozesssteuerung

4.5.2.1 Kommunikation und Kooperation

Die über mehrere Orte verteilte Erstellung und Nutzung von Kursen und Lernobjekten erschwert auch die Kommunikation zwischen den Prozessbeteiligten. Gerade in den Fällen, wenn Lehrmaterialien aus Lernobjekten unterschiedlicher Unternehmen erstellt werden sollen, werden sich die jeweiligen Prozesspartner gar nicht persönlich kennen[61], was eine Koordination der einzelnen Schritte enorm erschwert. Es ist in diesem Fall nicht so einfach möglich, dass sich zwei Prozesspartner mittels eines Telefonats oder E-Mailaustausch absprechen. Ist eine unmittelbare Kommunikation der Prozesspartner nicht möglich, kann eine Koordination nur indirekt über die Kurse und Lernobjekte stattfinden. Dies bedeutet aber auch, dass erst dann ein Prozessschritt ausgeführt werden kann, wenn alle Kurse und Lernobjekte, die für diesen Prozessschritt benötigt werden, verfügbar sind. Eine weitere Konsequenz ist,

[61] Die Lernobjekte werden in diesen Fällen meist nicht an eine bestimmte Person gebundene Kontaktdaten enthalten, sondern generische Kontaktdaten. Dies hat den Hintergrund, dass sich Zuständigkeiten für bestimmte Lernobjekte aus verschiedenen Gründen (Umstrukturierungen im Unternehmen, Änderung von Aufgaben der zuständigen Sachbearbeiter, etc.) leicht ändern können, was jedoch für Kunden des Unternehmens – die Nutzer der Lernobjekte – keine Auswirkungen haben darf. Die Erfahrung zeigt jedoch, dass zwischen unterschiedlichen Zuständigen für dieselben Kontaktdaten oftmals gravierende Unterschiede in Wissensstand und Fähigkeiten bestehen und Übergaben von Aufgaben nicht immer reibungslos funktionieren, so dass Wissen verloren geht. Ist nicht ein einzelner unter bestimmten Kontaktdaten erreichbar, sondern eine Gruppe, kommt noch das Problem hinzu, dass Ansprechpartner unerwartet wechseln können oder dass sich bei einer Kontaktaufnahme niemand angesprochen fühlt.

dass unmittelbare Änderungsanforderungen zu bereits bestehenden Lernobjekten oder Feedback an Prozesspartner nicht unmittelbar möglich sind; dies ist nur indirekt unter Zuhilfenahme der betroffenen Kurse oder Lernobjekte möglich.

4.5.2.2 Iteratives Durchlaufen von Teilprozessen

In einigen Fällen wird es nötig sein, Kurse oder Lernobjekte zu überarbeiten – entweder inhaltlich, strukturell oder bzgl. ihrer Parameter. Teile des Prozesses werden für einen Kurs und eine Lehrveranstaltung nicht nur einmal, sondern mehrmals durchlaufen, solange bis alle Inhalte korrekt sind und alle Parameter soweit gesetzt und abgeglichen sind, dass der Kurs bzw. die Lernobjekte harmonieren. Hierbei ist wieder eine Abstimmung zwischen den Prozesspartnern nötig (vgl. Problemkreis „Kommunikation und Kooperation).

Eine Aussage, wie oft welche Teilprozesse oder Aktivitäten wiederholt werden müssen, ist nicht möglich. Ebenso ist keine Aussage darüber möglich, wann im Gesamtprozess Teilprozesse (oder auch nur einzelne Aktivitäten) wiederholt werden müssen. Dies hängt damit zusammen, dass der Prozess semistrukturiert ist: Jeder einzelne Teilprozess folgt einer bestimmten, mehr oder weniger starr vorgegebenen Struktur (vgl. hierzu die Beschreibungen der einzelnen Teilprozesse in 4.3), aber der Zeitpunkt, wann ein Teilprozess (beispielsweise „Design eines neuen Moduls“) erneut ausgeführt werden muss, steht nicht von vorneherein fest, sondern hängt von der konkreten Anwendung und von den Vorstellungen und Arbeitsweisen der jeweiligen Prozessteilnehmer ab.

4.5.2.3 Abgleich zwischen Lehrmaterialerstellung und –nutzung

Dieser Problembereich umfasst die Unterschiede zwischen tatsächlicher und beabsichtigter Nutzung von Lehrmaterialien. Der Abgleich muss in beide Richtungen erfolgen.

Abgleich von der Erstellung von Lehrmaterialien mit deren Nutzung

Der Kursdesigner legt mittels Parametern und Constraints bestimmte Freiheitsgrade für die Kurse fest. So kann er beispielsweise festlegen, dass die Lerner mehr und kompliziertere Module als im Kurs vorgesehen lernen dürfen, nicht aber weniger oder leichtere Module. Dies betrifft ebenso die Erstellung von Modulen durch Moduldesigner und deren Nutzung durch Kursdesigner. Bei all diesen Szenarien muss der Designer eines Lernobjekts (Atom, Modul) oder eines Kurses immer prüfen, ob die von ihm vorgesehene Nutzung zu der tatsächlichen Nutzung passt (vgl. 2.4.2).

Abgleich von der Nutzung von Lehrmaterialien zu deren Erstellung

Die Ansprüche der Lerner definieren ein „Pflichtenheft“ für den Erstellungsprozess, nach dem Modul-, Atom- und Kursdesigner Kurse und Lernobjekte entwickeln sollen. Die jeweiligen Designer sind im allgemeinen erfahren in ihren Fachdisziplinen und können daher ihre (möglicherweise isolierten) Aufgabenbereiche gut erfüllen. Neben Inhalt und Struktur müssen sie aber auch die möglichen Variationen für unterschiedliche Nutzungen eines Lernobjekts oder eines Kurses vordefinieren. Aber kann der Designer alle Nutzungen vorhersehen?

Abgesehen davon gibt es noch dynamische Verhaltensweisen der Nutzer, beispielsweise Lernen mittels der Erstellung von adaptierbaren oder gar customized Kursen, welche unter Umständen den Kontext und die Varianten stark beeinflussen. In der Realität ist eine Einschätzung aller möglichen Nutzungen sicher schwierig; eine Evaluation ist somit erforderlich, zwar beispielsweise durch nutzerseitige Kontrolle und nutzerseitiges Feedback. Dies ist für die Designer eine notwendige Information zur Verbesserung und Korrektur der vordefinierten Varianten. Als Designer muss man daher die Nutzung seiner Lernobjekte oder Kurse vor Augen haben und bei Bedarf die Lernobjekte oder Kurse sowie deren Beschreibungen im nachhinein ändern und anpassen (mehrmalige Spezifizierung und Verfeinerung). Dies führt

zu mehrmaligem Design und Entwicklung, und zwar für jede vertikale Abstraktion (Inhalt, Struktur, Nutzung, Präsentation und Navigation, vgl. 3.2.2).

4.6 Kritik: Schwächen des Prozesses

Der Prozess, der in diesem Kapitel vorgestellt wurde, bietet bereits einige Vorteile. Insbesondere wird ein durchgängig verwendetes Datenmodell für Lehrmaterialien eingesetzt und die Nutzung sowie die Erstellung der Lehrmaterialien wird durchgängig abgebildet. Dennoch gibt es einige Schwächen im Prozess, die im folgenden betrachtet werden sollen.

4.6.1 Komplexität beim Umgang mit Constraints

Constraints regeln die Definition und Belegung von Parametern von Kursen und Lernobjekten, um beispielsweise bestimmte Formen der Nutzung zu modellieren[62]. Constraints können hierbei sehr komplex sein und auch von den Werten mehrerer Parameter abhängen.

Der iterative Charakter der meisten Teilprozesse, die in 4.3 beschrieben wurden, kann hier zum Problem werden: Angenommen ein Modul wird mehrmals verbaut und eine Änderung eines Constraints auf einem Parameter dieses Moduls wird durchgeführt. Ist durch die Constraint lediglich ein Parameter betroffen, sind die Folgen absehbar: Alle Submodule, die in dem fraglichen Modul verbaut sind, werden rekursiv überprüft, ob sie den neuen Constraints noch genügen, und umgekehrt wird in allen Lernobjekten, die das fragliche Modul verbauen, überprüft, ob das Modul aufgrund der veränderten Constraints noch verwendet werden kann oder ob es durch ein anderes Modul ausgetauscht werden muss. Sind durch die Constraint allerdings mehrere Parameter betroffen, so können sich Situationen ergeben, in denen Constraints im Widerspruch zueinander stehen: Wenn eine Constraint befolgt wird, wird eine andere Constraint verletzt und umgekehrt. Dieser Konflikt kann nicht mehr programmatisch aufgelöst werden und erfordert den Eingriff durch die entsprechende Rolle (Atomdesigner, Moduldesigner, Kursdesigner, etc.). Also sind in vielen komplexen Fällen sowohl der Definition und Korrektur als auch der Auswertung von Constraints menschliche Interaktion notwendig, und diese führt in komplexen Fällen möglicherweise zum Problem der Übersichtlichkeit und Fehleranfälligkeit.

4.6.2 Kommunikation und Koordination

Der verteilte Charakter des Prozesses schafft enorme Probleme bei der Kommunikation der Prozessbeteiligten:

- Ansprechpartner für Atome, Module und Kurse müssen nicht bekannt sein. Dieses Problem tritt insbesondere dann auf, wenn geschlossene Organisationen, beispielsweise Unternehmen, Atome, Module und Kurse anbieten und vielleicht nur eine E-Mailadresse, die nicht direkt einer Person zugeordnet ist, als Kontakt anbieten (vgl. auch 4.5.2.1).
- Ansprechpartner für Atome, Module und Kurse (falls bekannt) ändern sich; die Änderung wird nicht allen Prozesspartnern mitgeteilt bzw. nicht an den Lernobjekt-Service propagiert.
- Das gemeinsame Datenmodell schafft zwar Klarheit hinsichtlich der Eigenschaften und des Aussehens von Lernobjekten und der in diesem Kapitel vorgestellte Prozess strukturiert die für die Lehrmaterialerstellung nötigen Schritte. Hinsichtlich der Abstimmung, welcher Prozessteilnehmer welche Rolle einnimmt und wie er seine Auf-

[62] Für Details zu Constraints siehe 3.4.2.3 und 3.5.2.3.

gabe mit den anderen Prozessbeteiligten koordiniert, muss hingegen kein gemeinsames Verständnis vorliegen.

- Zwischen den Prozessbeteiligten muss kein gemeinsames Verständnis bzgl. der Parameter und ihrer Werte existieren. Dieses Problem wurde bereits in 3.2.1.3 adressiert und zur Lösung wurden Ontologien auf Parametern oder Kategorien vorgeschlagen.

Aufgrund dieser Probleme wird eine Kommunikation zwischen den Prozessbeteiligten sehr erschwert, worunter die Abstimmung zwischen den einzelnen Teilprozessen leidet. So entstehen enorme Reibungsverluste bei der Erstellung von Lehrmaterialien, was einerseits zu längeren Prozessdurchlaufzeiten führt und andererseits der Qualität der erstellten Lehrmaterialien abträglich ist.

Ein weiteres Problem ist, dass im Prozess Feedback bislang nur in Form von Änderungsanforderungen vorgesehen ist (vgl. Überarbeiten eines Lernobjekts in 4.3.1.4 bzw. eines Kurses in 4.3.2.4). Feedback von Nutzern wird nicht berücksichtigt. (Hier wird davon ausgegangen, dass die Kurstutoren und Lerner Feedback unmittelbar an den Kurs-Presenter herantragen, der dann das Feedback prüft und gegebenenfalls eine Änderungsanforderung erstellt. Die Filterung des Feedbacks durch den Kurs-Presenter hat ihre Berechtigung, um ungerechtfertigte Kritik abblocken und sinnlose Arbeiten verhindern zu können.) Dennoch sollte zwischen Feedback und konkreten Änderungsanforderungen unterschieden werden, um bereits beschlossene Änderungen, die durchgeführt werden müssen und die in Form von Änderungsanforderungen vorliegen, von Ideen, die erst noch diskutiert werden müssen, unterscheiden zu können. Dies wird im folgenden Kapitel näher betrachtet.

4.7 Zusammenfassung

In diesem Kapitel wurden die Aktivitäten vorgestellt, die nötig sind, um Lehrmaterialien aus wiederverwendbaren Lernobjekten zu erstellen und zu nutzen. Hierzu wurde ein generischer Prozess definiert, in dem sich die real stattfindenden Prozesse wiederfinden; dieser generische Prozess bildet die Grundlage für alle weiteren Betrachtungen hinsichtlich der Lehrmaterialerstellung und –nutzung. Die Wiederverwendung von Lernobjekten geschieht zum einen zwischen Design und Entwicklung innerhalb jedes Teilprozesses und zum anderen zwischen den einzelnen Phasen in Form von Teilergebnissen, die entlang der gesamten Prozesskette verwendet werden.

Zuerst wurden die an dem Prozess beteiligten Rollen vorgestellt: Atom-, Modul- und Kursdesigner erstellen die Kurse und Lernobjekte, die später von Kurs-Presentern, Kurstutoren und natürlich den Lernern in der Lehre und im Selbststudium genutzt werden. Anschließend wurde der Prozess im Überblick vorgestellt: Als grobe Schritte wurden Modulentwurf, Kursentwurf, Durchführung einer Lehrveranstaltung, Tutorübung und das Selbststudium identifiziert, wobei zwischen diesen Schritten nicht notwendigerweise eine strenge Reihenfolge herrschen muss. Zwischen den einzelnen Prozessschritten (die jeweils eigenständige Teilprozesse kapseln) werden Sichten definiert, um den Datenfluss zwischen den Prozessschritten zu strukturieren. Sichten sind hierbei eine Zusammenfassung aller Informationen und Daten (beispielsweise Lernobjekte), die entweder als Eingabe in einen Prozessschritt oder als Ergebnis eines Prozessschrittes anfallen.

Nachdem diese Grundlagen definiert wurden, wurden die einzelnen Teilprozesse des Lehrmaterialerstellungs- und des Lehrmaterialnutzungsprozesses sowie die in ihnen enthaltenen Aktivitäten im Detail vorgestellt: Die Schnittstellen zwischen den Teilprozessen (Eingangs- sowie Ausgangsdaten) wurden genannt und die einzelnen Aktivitäten wurden beschrieben.

Anschließend wurde jeder Teilprozess einer kritischen Betrachtung unterzogen und Besonderheiten eines jeden Teilprozesses wurden herausgestellt.

Anschließend wurde das Zusammenspiel zwischen den einzelnen Teilprozessen und Aktivitäten mit dem zugrundeliegenden Datenmodell beschrieben. Das Datenmodell unterstützt alle Prozessschritte vom Entwurf der Module bis hin zur Nutzung der Lehrmaterialien ohne Medienbruch. Des weiteren erleichtert es durch die Bereitstellung von Sichten die Strukturierung des Lehrmaterialerstellungs- und –nutzungsprozesses.

Die Beobachtungen aus der Beschreibung der Teilprozesse sowie der Integration des Datenmodells in den gesamten Lehrmaterialerstellungs- und –nutzungsprozess wurden anschließend zusammengefasst. Dabei wurden insbesondere Probleme der Datenhaltung, beispielsweise die Probleme bei der Nutzung von Shared Workspaces, als auch der Prozesssteuerung angesprochen. Besonderes Augenmerk wurde auch auf den iterativen Charakter vieler Teilprozesse und auf die Möglichkeit, Änderungsvorschläge und Feedback geben zu können, gelegt. Beide Aspekte benötigen als Grundlage eine funktionierende Kommunikation als Basis einer effektiven Koordination und Kooperation zwischen den Prozesspartnern.

Dies ist auch einer der wesentlichen Kritikpunkte bei dem vorgestellten Lehrmaterialerstellungs- und –nutzungsprozess: Die Sicherstellung einer effizienten Kommunikation zwischen nicht notwendigerweise namentlich oder persönlich bekannten Prozesspartnern in unterschiedlichen Organisationen und Unternehmen. Insbesondere die internen Strukturen von kommerziellen Lehrmaterialerstellern können eine effiziente Kommunikation sehr erschweren, und ein Ziel dieser Arbeit ist es, diese Hürden zu überwinden. Ein weiterer Kritikpunkt war die Komplexität von Constraints, mit der die verschiedenen Prozessteilnehmer umgehen müssen. Dies betrifft insbesondere die „nutzergerechte Verpackung“ der Syntax und Semantik der Constraints und ihrer Beschreibung, um ihre Nutzung zu erleichtern.

Im folgenden wird mit Annotationen ein Ansatz vorgestellt, der die gerade genannten Kritikpunkte löst: Annotationen erlauben es, auch komplexe Constraints so zu beschreiben, dass die Prozessteilnehmer sie verstehen und gewinnbringend nutzen können. Essenziell ist hierbei die Freiheit, in Annotationen auch natürliche Sprache verwenden zu können. So kann die hohe Abstraktion und Komplexität von Constraints besser verständlich und greifbarer gemacht werden. Des weiteren können Annotationen genutzt werden, um Feedback zu Lernobjekten und Kursen zu übermitteln und die Kommunikation bezüglich bestimmter Lernobjekte oder Kurse auch bei völlig unbekannten Kommunikationspartnern zu ermöglichen.

5 Annotationen zur Verbesserung der Wiederverwendung von Lehrmaterialien

5.1 Überblick

In den letzten beiden Kapiteln wurden ein Datenmodell und ein generischer Prozess vorgestellt, die beide als Grundlage für die Erstellung und Nutzung von Lehrmaterialien dienen. Der Prozess beschreibt die Schritte, die zur Erstellung von Lehrmaterialien nötig sind, und welche Rollen diese Schritte ausführen. Wesentliche Eigenschaften des Prozesses sind sein verteilter Charakter und die Abwesenheit von Schritten zur Rückmeldung von Feedback an Prozessteilnehmer, die vorgelagerte Prozessschritte ausführen. Das Datenmodell ist die Grundlage, auf der der Prozess arbeitet. Es ist so gestaltet, dass bereits bestehende Lernobjekte leicht integriert und neue Lernobjekte ebenso leicht erstellt werden können. Die Beschreibung eines Lernobjekts erfolgt mittels Parametern. Das Datenmodell wirkt wie eine einheitliche Hülle um die Inhalte und deren Beschreibungen und stellt einen einheitlichen Zugriff sicher.

Sowohl beim Datenmodell als auch beim Prozess gibt es Probleme (vgl. 3.7 und 4.6), die einer effizienten Erstellung möglichst gut wiederverwendbarer Lehrmaterialien entgegenstehen. Als Lösung für diese Probleme wurden Annotationen vorgeschlagen.

In diesem Kapitel wird nun im Detail beschrieben, wie Annotationen die verschiedenen Probleme adressieren. Zu Beginn werden in 5.2 Annotationen näher vorgestellt und untersucht, um ein besseres Gespür für einen sinnvollen Einsatz von Annotationen zu erhalten. Anschließend werden die einzelnen Problemfelder untersucht: In 5.3.1 wird beschrieben, wie die Probleme im Datenmodell genau gelagert sind und wie hier Annotationen helfen können, diese zu lösen. In 5.3.2 wird analog für die Probleme des Lehrmaterialerstellungs- und. –nutzungsprozesses vorgegangen. Um die Lösungsvorschläge aus 5.3.1 und 5.3.2 zu realisieren, müssen das Datenmodell und der Lehrmaterialerstellungs- und -nutzungsprozess geeignet erweitert werden. Was hierzu nötig ist, wird schließlich in 5.4 beschrieben.

5.2 Das Wesen von Annotationen

Bei „Annotationen“ denkt man intuitiv zuerst an Freitext, beispielsweise Kommentare in Büchern, beschriftete PostIts, die an einen Gegenstand geheftet sind, etc. In dieser Arbeit – und insbesondere in diesem Kapitel – soll das Verständnis von Annotationen etwas erweitert werden: Annotationen sollen als ein Mechanismus zur Kommunikation zwischen Menschen

und zur Präzisierung von Sachverhalten verstanden werden. Diese Sichtweisen ermöglichen es, dass Annotationen auf umfassende Weise zur Verbesserung der Wiederverwendbarkeit von Lehrmaterialien eingesetzt werden können. Wie dies bewerkstelligt wird, wird in der zweiten Hälfte dieses Kapitels beschrieben. Grundlage hierfür bildet ein Überblick über verschiedene Aspekte von Annotationen, der in der ersten Hälfte dieses Kapitels geliefert wird.

5.2.1 Begriffsklärung und Einordnung

5.2.1.1 Begriffsklärung

Annotationen können sehr verschieden realisiert sein, beispielsweise als Fußnoten, als Pop-up-Fenster bei der Verwendung von HTML, etc. Allen Realisierungen gemeinsam ist, dass Annotationen einen *Inhalt* und einen *Bezug zu einer Entität* (hier: ein Lernobjekt oder ein Kurs) haben.

Inhalt

Der Inhalt einer Annotation kann ein beliebiger Medientyp sein. Am einfachsten ist Freitext, aber auch Bilder oder sonstige multimediale Inhalte sind möglich. Ebenso können Annotationen beliebig einfach oder beliebig komplex strukturiert sein.

Annotationen können selbst als Entität betrachtet und somit selbst wieder annotiert werden. Es ist auch möglich, bereits bestehende Entitäten als Annotationen zu anderen Entitäten zu behandeln.

Bezug zu einer Entität

Eine Annotation hat immer einen Bezug zu einer Entität, die sie annotiert. Die Entität, auf die sich eine Annotation bezieht, wird auch *Basiselement* genannt.

Die Beziehung zwischen einer Annotation und ihrem Basiselement kann auf verschiedene Weise realisiert werden. Annotationen können in das Basiselement eingebettet sein. In diesem Fall spricht man von einer *Inline-Annotation*. Annotationen können aber auch als eigenständige Entitäten vorliegen. In diesem Fall wird die Beziehung zwischen dem Basiselement und ihrer Annotation mittels einer Referenz realisiert; die Annotation wird dann *referenzierte Annotation* genannt (vgl. Abbildung 5-1).

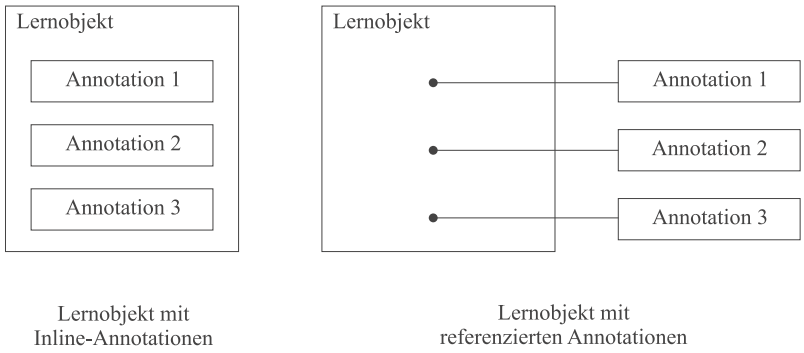


Abbildung 5-1: Inline-Annotationen und referenzierte Annotationen

Referenzierte Annotationen haben gegenüber Inline-Annotationen folgende Vorteile:

- Inline-Annotationen sind in ihr Basiselement eingebettet; damit ist beim Einfügen und Bearbeiten einer Inline-Annotation eine Modifikation des Basiselements nötig. Bei referenzierten Annotationen ist dies nicht so. Hier ist es möglich, die Beziehung zwischen einem Basiselement und einer Annotation so zu realisieren, dass keine Änderungen am Basiselement und der Annotation nötig sind.

- Referenzierte Annotationen können als eigenständige Entitäten vorliegen. Inline-Annotationen sind immer Teil ihres Basiselements.
- Referenzierte Annotationen können auf einfache Weise selbst wieder annotiert werden. Annotation von Inline-Annotationen führt zu mehrfach verschachtelten Inline-Annotationen, die deutlich komplizierter zu realisieren und zu pflegen sind.

Die Realisierung der Referenz von einem Basiselement zu einer Annotation muss je nach Medientyp des Basiselements unterschiedlich realisiert werden. Liegt das Basiselement als strukturiertes Dokument, beispielsweise in XML, PDF oder Microsoft Word, vor, so kann jedes Element des Basiselements annotiert werden (mehr hierzu in 7.2.1.3 und [KLTWO04]). Bei multimedialen Elementen ist dies komplizierter. Hier bietet sich als Ankerpunkt für die Referenz eine Adressierung eines Ausschnitts beispielsweise nach dem HyTime-Standard [HYTIME] an.

5.2.1.2 Zusammenhang Metadaten – Annotationen

Metadaten (vgl. 2.5.1) und Annotationen werden häufig als zwei getrennte Konzepte gehandelt. Dies liegt unter anderem daran, dass Metadaten und Annotationen meist für verschiedene Zwecke eingesetzt werden und dass Annotationen gedanklich üblicherweise mit Freitext verbunden werden.

Metadaten und Annotationen sind jedoch zwei eng verwandte Konzepte; Metadaten liefern ebenso wie Annotationen Informationen über ein Objekt. Metadaten können als eine bestimmte Art von Annotation angesehen werden, die einer streng festgelegten Struktur mit einem genau definierten Vokabular folgt. Metadaten wären somit eine strukturierte Annotation (der Begriff wird in 5.2.3.2 eingeführt) mit definiertem Vokabular. Umgekehrt können Annotationen als Metadaten implementiert werden. Metadaten werden häufig als Attribut-Wertpaare realisiert. Eine (Freitext-)Annotation könnte beispielsweise mittels eines Attributs „Annotation“ mit frei belegbarem Wert realisiert werden. Der Text der Annotation würde dann im Wert des Attributs abgelegt werden.

5.2.1.3 Verwandte Themenbereiche

Zu Annotationen gibt es einige verwandte Themen aus anderen Bereichen der Informatik. Annotationen können selbst annotiert werden, wodurch sich ein Diskussionsfaden entspinnen kann. Dies führt beispielsweise in den Bereich der Diskussionssysteme, der weiter unten kurz angesprochen wird. Ebenso können Annotationen eine koordinierende Funktion haben, wodurch sie insbesondere für Multi-Autoren-Systeme interessant werden. Im folgenden werden einige Themenbereiche, die eine enge Verbindung mit Annotationen haben, kurz vorgestellt.

Diskussionssysteme: Usenet News und Internet-basierte Diskussionsforen

Usenet News [McL97] und Internet-basierte Diskussionsforen bieten die Möglichkeit, Artikel zu schreiben, auf die andere Nutzer antworten können. Beispielsysteme sind HyperNews [LaL95] oder CoNote [Dav94]. Auf einen Artikel wird geantwortet, die Antwort wird selbst wieder kommentiert, und so fort. Im Vordergrund steht hierbei die Kommunikation zwischen den Nutzern.

Die News-Threads, die entstehen, wenn auf einen Artikel eine Menge von Antwortartikeln folgt, die wiederum beantwortet werden, sind sehr ähnlich zu den Strukturen, wenn ein Objekt annotiert wird und diese Annotation selbst wieder annotiert wird.

Multi-Autoren-Systeme

In Multi-Autoren-Systemen werden Dokumente, Objekte, etc. von mehreren Autoren gemeinsam erstellt. Wesentlich für das Gelingen ist eine enge Zusammenarbeit der Autoren. Insbesondere sind Markierungen nötig, die Änderungen am gemeinsamen Dokument bzw. Objekt markieren. Diese Markierungen kann man als eine Art Annotation verstehen. Ein Beispiel für ein Multi-Autoren-System ist der Dokumenteneditor IRIS. Details sind in [KoKo97] zu finden.

Shared Workspaces

Shared Workspaces stellen mehreren Nutzern eine gemeinsame Datenablage zur Verfügung. Sie werden häufig als Repository für alle Dokumente und Objekte genutzt, die im Rahmen einer Projektarbeit entstehen. Damit die Nutzer erkennen, an welchen Dokumenten und Objekten Änderungen vorgenommen wurden oder welche Dokumente und Objekte neu hinzugekommen sind, werden Markierungen an den Dokumenten und Objekten angebracht, die entsprechende Informationen – Datum der letzten Änderung, Autor, etc. – beinhalten. Beispiele hierfür sind BSCW [BSCW] oder HyperWave [Hyperwave].

Allgemeine Annotationssysteme

Im Gegensatz zu den bisher genannten Themenbereichen steht bei den allgemeinen Annotationssystemen die Annotation von beliebigen Objekten im Internet im Vordergrund. Die Annotationen beinhalten Zusammenfassungen, Kommentare, private Annotationen, Landmarks etc. Für ein Beispiel vgl. das System ComMentor [RMW97].

5.2.1.4 Gängige Anwendungen von Annotationen

Annotationen in Semantic Web

Ein großer Anwendungsbereich für Annotationen ist das Semantic Web. Das Semantic Web ist eine Erweiterung des heutigen World Wide Web (kurz: WWW), in der Informationen eine wohldefinierte Bedeutung bekommen, damit Rechner und Menschen besser zusammenarbeiten können, vgl. [SW a] , [SW b], [SSS01] und [BHL01].

Das Semantic Web versucht insbesondere die heute bestehenden Schwächen bei der Definition, Integration und Auffinden von Informationen zu lösen. Insbesondere gibt es folgende Probleme:

- Eine Definition und Integration von Wissensmanagement-Elementen in Informationen ist nötig, um den Nutzern allgemein verständliche und dennoch domänen- bzw. kontextspezifische Informationen bereitstellen zu können.
- Beim Auffinden von Informationen sollen nicht nur Verweise auf Dokumente geliefert werden, die (möglicherweise) die gesuchten Informationen beinhalten. Statt dessen soll eine direkte Verknüpfung der Informationen aus verschiedenen, heterogenen Informationsquellen möglich sein. Dazu gehört auch, entsprechende Recherchemöglichkeiten zur Verfügung zu stellen, beispielsweise ein Glossar zum Suchen.

Die Lösung dieser Probleme erfordert es, die Semantik von Dokumenten im WWW zu beschreiben. Hierzu werden heute üblicherweise Ontologien verwendet. Nach Aßmann [Aß02] hat eine Ontologie folgende Eigenschaften:

- Ontologien verwenden eine standardisiere Menge von Begriffen und Kontextbedingungen.
- Das Vokabular ist durch Mehrfachvererbung partiell geordnet.

- Kontextbedingungen sind durch Inferenzregeln abgeleitet und allgemein verständlich.

Annotationen werden nun verwendet, um Dokumenten Semantik zuzuweisen. In diesen Annotationen werden Dokumente auf der Basis von Ontologien spezifiziert, wobei ein standardisiertes Vokabular verwendet wird. Die Dokumente werden so mittels Annotationen mit Metadaten angereichert. Davon können beispielsweise wissenschaftliche Datenbanken oder Web-Portale profitieren. Wie dies in verschiedenen Anwendungen funktioniert, wird im folgenden genauer beschrieben.

Annotationen bei der Wissenserzeugung
Dokumente, Teile von Dokumenten und Beziehungen zwischen Dokumenten können mittels Ontologien kategorisiert werden. Die Kategorisierung geschieht auf der Grundlage von Metadaten, die mittels Annotationen den Dokumenten, deren Teilen bzw. den Beziehungen zugeordnet werden. Die Metadaten müssen konform zum Vokabular der Ontologie sein, so dass mittels Analysen und der Anwendung von Inferenzregeln Beziehungen zu weiteren Dokumenten hergestellt und so Wissen generiert werden kann.

Annotationen zum Zugriff auf Wissen
Wie im letzten Abschnitt beschrieben, werden Dokumente mit Annotationen versehen, die Metadaten eines standardisierten Vokabulars enthalten. Dies ermöglicht eine effiziente Suche, denn statt identischen Zeichen kann nun nach Begriffen mit einer bestimmten Semantik gesucht werden, wobei auch „benachbarte“, ähnliche Konzepte und Begriffe in die Suche mit einbezogen werden können.

Vokabular für Metadaten
XML- oder HTML-Dokumente im WWW werden heute mit Metadaten in RDF [RDF] oder der darauf aufbauenden Web Ontology Language OWL [OWL] annotiert. Beide Sprachen dienen dazu, Aussagen über Dokumente in maschinenlesbarer Form abzulegen und aufgrund der besseren Kategorisierungsmöglichkeiten für beispielsweise die Dokumentensuche zu nutzen. An dieser Stelle wird auf die Projekte Simple HTML Ontology Extensions (SHOE) [SHOE], Creating RElation Annotation-based Metadata (CREAM) [HaSt02] und W3C Annotea verwiesen [Ann01].

Annotationen im E-Learning

Annotationen im Zusammenhang mit E-Learning werden heute üblicherweise so verstanden, dass Lerner schnell Informationen und Meinungen zu Lehrveranstaltungen und Lehrmaterialien austauschen können. Annotationen werden hier oft im Rahmen von Lerngruppen oder Communities von Lernern benutzt, so dass sie auch als ein Werkzeug der Kommunikation dienen.

Folgende Aufstellung bietet einen Überblick, wie Annotationen im Bereich E-Learning verwendet werden können:

- Annotationen als Links:

  Die Autoren von Kursen können Annotationen nutzen, um den Lernern einen bestimmten Lern- bzw. Navigationspfad zu empfehlen. Die Annotationen dienen hierbei als Link auf die Lehrmaterialien, die die Lerner als nächstes lesen bzw. bearbeiten sollten.

- Annotationen als Feedback-Fragebogen:

  Um das Lernergebnis zu prüfen und so gezielt die Qualität des Lernens verbessern zu können, füllen die Lerner Feedback-Fragebögen in Form von Annotationen aus.

Im Projekt Automatic Questionnaire and Analysis (AQuA)[63] beispielsweise kann so der Dozent überprüfen, ob die Lerner den Lehrstoff verstanden haben (Details siehe [GaMü03] und [AQuA]).

- Annotationen für Bookmarks:

  Durch eine Analyse von Annotationen von Lernern werden automatisch kategorisierte Bookmark-Listen oder Empfehlungen erzeugt. Dabei unterstützt das System AKI[64] die Bewertung von URLs in Bookmark-Listen nach festgelegten Kriterien und die Suchmechanismen. Details sind in [BoSc97] beschrieben.

- Annotationen als Kommentar:

  Lerner verfassen Annotationen als Kommentar zu (Teilen von) Dokumenten. Diese Annotationen können von anderen Lernern gelesen und wiederum annotiert werden. Mit dieser Art des Informationsaustausches sollen die klassischen elektronischen Medien wie Homepages, Newsgroups und E-Mail[65] kombiniert und so verbessert werden. Dabei wurde darauf Wert gelegt, dass die Zusammenhänge der Annotationen erhalten bleiben und relevante Informationen besser gefunden werden können [Don01].

- Annotationen als Lehrinhalt bei Präsenzveranstaltungen:

  Im Rahmen von Präsenzveranstaltungen sind mehrere Arten der Nutzung von Annotationen denkbar. Einmal können Video- und / oder Audio-Aufzeichnungen der Lehrveranstaltungen an das Skript der Lehrveranstaltung angehängt werden. Ebenso können Annotationen verwendet werden, um Videoaufzeichnungen der Lehrveranstaltungen gezielt mit weiteren Informationen zu versehen, beispielsweise um Links zu weiteren Informationen zu dem gerade besprochenen Thema zu präsentieren (Details siehe Projekt Call A Lecture (CAL)[66] [CAL])

5.2.2 Annotationen: Dualismus von Nutzung und Inhalt

Annotationen, die an Lernobjekte oder Kurse angehängt werden, wirken auf mehrere Arten: Werden in der Annotation Schwächen des Inhalts, der Struktur, der Parametrisierung oder der Darstellungsweise von Lernobjekten beschrieben, so werden in erster Linie Schwächen der Elemente bzw. des Kurses aufgezeigt, die behoben werden müssen. Insofern hat die Annotation eine *inhaltliche Dimension*, die – wenn die Vorschläge dieser Arbeit umgesetzt werden – eine Verbesserung der Qualität und damit auch der Wiederverwendung der Lernobjekte bzw. des Kurses zur Folge haben.

Die Beschreibung von Schwächen der Lernobjekte bzw. der Kurse sind jedoch auch eine Aufforderung an den oder die Autoren der Lernobjekte bzw. der Kurse, diese Schwächen zu korrigieren. Hier kommen Koordinationsaspekte ins Spiel, wird doch durch die Annotation zugleich eine Abstimmung zwischen den Prozessteilnehmern bezüglich der zu erledigenden Aufgaben realisiert. Insofern haben Annotationen eine *pragmatische Dimension*.

---

[63] AQuA ist ein System für die einfache Erstellung und Auswertung von Online-Feedback-Fragebögen für die computerunterstützte Weiterbildung, entwickelt vom Fraunhofer Institut für Integrierte Publikations- und Informationssysteme in Darmstadt.

[64] AKI wurde von Uni Oldenburg in Kooperation mit anderen Instituten und Unternehmen entwickelt. Es handelt sich um ein Annotationssystem zur kooperativen Nutzung gefundener Informationen im WWW.

[65] Annotationen unterscheiden sich von den klassischen elektronischen Kommunikationsmedien dadurch, dass die Kommunikation immer passiv über ein gemeinsames Objekt stattfindet.

[66] CAL ist ein Forschungsprojekt aus dem Gebiet E-Learning des Lehrstuhls für Angewandte Informatik und Kooperative Systeme, Institut für Informatik, TU München.

Anforderungen in Annotationen werden üblicherweise nicht sofort umgesetzt, sondern bedürfen meist noch einer Klärung oder – bei verschiedenen Ansichten – einer Diskussion unter den Prozessteilnehmern (vgl. 4.6.2). Annotationen sind somit auch Ausgangspunkt einer Kommunikation unter den verschiedenen Prozessteilnehmern, die von den in der Annotation genannten Maßnahmen betroffen sind. Diese Diskussionen können unmittelbar zwischen den Betroffenen erfolgen, beispielsweise in Besprechungen oder Telefonkonferenzen, aber auch indirekt über Annotationen, die wiederum an Annotationen geheftet werden. Annotationen können somit eine Kommunikation zwischen Prozessteilnehmern in Gang setzen und zugleich auch Kommunikationswerkzeug sein. Annotationen haben insofern auch eine *soziale Dimension*. Abbildung 5-2 setzt diese Aspekte in Zusammenhang.

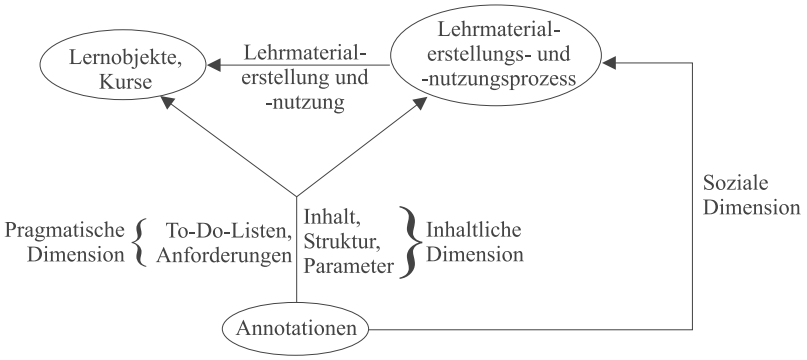


Abbildung 5-2: Dualismus inhaltliche Dimension – pragmatische Dimension – soziale Dimension

Interessant ist hierbei das Zusammenwirken der inhaltlichen und der pragmatischen Dimension von Annotationen. Beide Dimensionen sind eng verbunden; genau genommen können sie bei der oben beschriebenen Nutzung von Annotationen gar nicht getrennt werden. Je nachdem, aus welchem Blickwinkel ein Prozessteilnehmer eine Annotation betrachtet, steht für ihn der inhaltliche Aspekt oder die Aufforderung, etwas zu tun, im Vordergrund. Annotationen wohnt somit ein Dualismus zwischen der Beschreibung einer Schwäche eines Lernobjekts oder eines Kurses und der Aufforderung, bestimmte Maßnahmen durchzuführen, inne.

5.2.3 Kategorisierung von Annotationen

Annotationen umfassen mehrere Aspekte, anhand derer sie kategorisiert werden können. Hier ist beispielsweise der Grad der Formalisierung von Annotationen zu nennen, oder auch, wer auf eine Annotation zugreifen kann. Abbildung 5-3 zeigt die in dieser Arbeit betrachteten Aspekte auf.

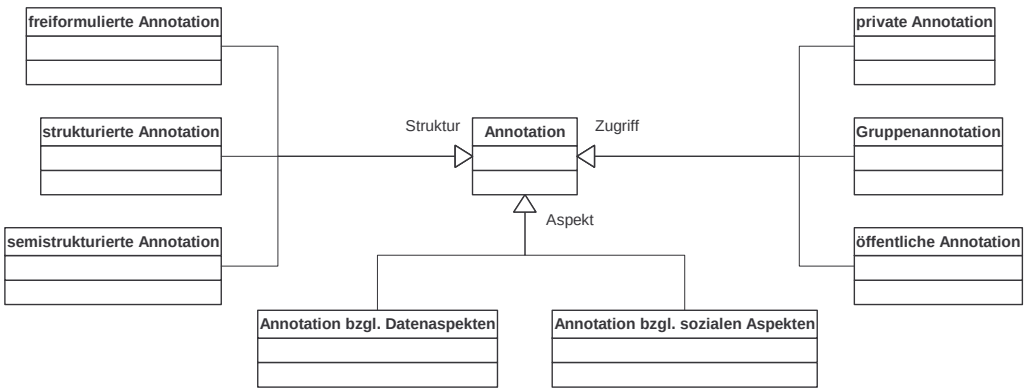


Abbildung 5-3: Kategorisierungsmöglichkeiten für Annotationen

5.2.3.1 Datenaspekte und soziale Aspekte

Annotationen können unterschieden werden je nachdem, ob eine Annotation Aspekte der Lernobjekte oder Kurse (beispielsweise Inhalt, Struktur, Parametrisierung, etc.) zum Thema hat oder ob sie der Koordination und Steuerung des Lehrmaterialerstellungsprozesses dient.

Annotationen der ersten Art haben zum Ziel, die Qualität der Lehrmaterialien zu erhöhen, indem Fehler korrigiert werden, die Struktur zum Zwecke der Klarheit und Einfachheit überarbeitet wird oder die Parametrisierung der Lernobjekte oder Kurse korrigiert oder präzisiert wird, um die Lernobjekte oder Kurse treffender zu beschreiben. In der Summe wird durch solche qualitätssichernden Maßnahmen die Wiederverwendbarkeit erhöht.

Annotationen der zweiten Art dienen der besseren Abstimmung im Prozess mit allen Facetten der Verbesserung der Prozessqualität. Dies umfasst beispielsweise die Prozessdurchlaufzeit, also die Zeit, die zur Erstellung qualitativ hochwertiger Lehrmaterialien benötigt wird, aber auch die Optimierung des Prozesses an den Schnittstellen der Teilprozesse hinsichtlich Abstimmung der Prozessteilnehmer und Datenübergabe.

In der Realität kann keine strenge Einteilung von Annotationen in die eine Kategorie (Inhalt: Datenaspekte) oder die andere Kategorie (Prozessunterstützung: soziale Aspekte) vorgenommen werden. Wie bereits in 5.2.2 beschrieben, beinhalten Annotationen meist beide Aspekte – allerdings in unterschiedlicher Gewichtung.

5.2.3.2 Struktur und Formalisierung

Annotationen können entsprechend des Grades ihrer Strukturierbarkeit kategorisiert werden. Dabei wird zwischen *freiformulierten* (unstrukturierten), *semistrukturierten* und *strukturierten* Annotationen unterschieden. Annotationen der drei Kategorien unterscheiden sich vor allem in ihrer maschinellen Verarbeitbarkeit: Strukturierte Annotationen sind ähnlich Metadaten maschinell verarbeitbar, haben aber auch die Schwächen von Metadaten hinsichtlich ihrer Ausdrucksfähigkeit[67]. Semistrukturierte Annotationen können maschinell vorverarbeitet werden, beispielsweise mittels einer Vorfilterung für die Adressaten. Die unstrukturierten Anteile können zwar ebenfalls maschinell verarbeitet werden, was jedoch sehr aufwändig ist und leicht zu Fehlern und Mehrdeutigkeiten führen kann. Freiformulierte Annotationen sind gänzlich unstrukturiert und dienen meistens zum Verständnis für Menschen. Üblicherweise sind sie für maschinelle Verarbeitung nicht geeignet[68].

Freiformulierte (unstrukturierte) Annotationen

Freiformulierte Annotationen geben keine Struktur vor, der die Annotationen folgen müssen. Die Autoren der Annotationen können ohne Einschränkungen Freitext schreiben; freiformulierte Annotationen haben so eine gewisse Ähnlichkeit mit einer virtuellen Notiz, einen Post-It, das an ein Lernobjekt oder einen Kurs angeheftet wird. Eine maschinelle (Vor-)Verarbeitung ist so leicht nicht möglich.

Die Einsatzgebiete für freiformulierte Annotationen sind vielfältig: Die Annotationen können sich auf Lernobjekte und Kurse beziehen, müssen sich aber hierbei nicht auf den Inhalt oder die Struktur der Lernobjekte beziehen, sondern können reinen Kommunikations- oder Koordinationscharakter haben (vgl. 5.2.2). In einigen Fällen ist freiformulierter Text die einzige Möglichkeit, Schwächen oder Probleme auszudrücken, beispielsweise wenn Nutzungsanweisungen für Lernobjekte oder Kurse festgeschrieben werden sollen, die sich nur mittels extrem komplexen Ausdrücken in Parametern realisieren ließen. Auch komplexere Zusammenhänge und Abhängigkeiten von anderen Lernobjekten lassen sich mit freiformulierten

[67] Dies wird im Detail in 5.3.1.2 betrachtet.
[68] Für mehr Informationen zum Thema „maschinelle Verarbeitung von Sprache“ sei beispielsweise auf [SAB94] oder [DDLFH90] verwiesen.

Annotationen oftmals eleganter und für Menschen leichter verständlich beschreiben als mit logischen Ausdrücken in Parametern oder Constraints.

Strukturierte Annotationen

Strukturierte Annotationen sind Metadaten sehr ähnlich. Während Metadaten – in dieser Arbeit die Parameter – die Aufgabe haben, Lernobjekte und Kurse sowie deren Nutzung möglichst präzise zu beschreiben und einer maschinellen Verarbeitung zugänglich zu machen, haben Annotationen neben dem Datenbezug auch soziale Aspekte. Dies gilt auch für strukturierte Annotationen.

Strukturierte Annotationen eignen sich aufgrund ihres streng festgelegten Aufbaus weniger für die Kommunikation zwischen den Prozessteilnehmern oder für Erläuterungen und Anleitungen, wie bestimmte Lernobjekte am besten genutzt werden. Dafür bietet ihre fixe Struktur zu wenig Freiheiten. Statt dessen können sie gut für die Steuerung automatisierbarer Anteile des Lehrmaterialerstellungsprozesses genutzt werden, beispielsweise für Benachrichtigungen der Prozesspartner bei Änderungen an Lernobjekten (automatisierte History-Funktion). Folgende DTD beschreibt beispielhaft eine solche Annotation, die über Änderungen an einem Lernobjekt informiert[69]:

```
<!ELEMENT annotation (id, referer, domain, event)>,
```

wobei `id` der Identifikator der Annotation ist, `referer` der Identifikator des Lernobjekts oder des Kurses ist, auf den sich diese Annotation bezieht, `domain` den Aspekt (Inhalt, Struktur, bestimmter Parameter) des Lernobjekts oder des Kurses beschreibt, den die Änderung betrifft, und `event` die Art der Änderung angibt:

```
<!ELEMENT id #CDATA>
<!ELEMENT referrer #CDATA>
<!ELEMENT domain („content“|„structure“|„parameter“)>
<!ELEMENT event („created“|„modified“|„deleted“)>
```

Semistrukturierte Annotationen

Semistrukturierte Annotationen kombinieren die Eigenschaften von freiformulierten und strukturierten Annotationen: Sie erlauben einerseits in gewissen Grenzen eine maschinelle Verarbeitung und bieten andererseits den Autoren der Annotationen alle Freiheiten, die sie auch bei freiformulierten Annotationen haben.

Dies wird durch die Struktur der Annotationen ermöglicht: Semistrukturierte Annotationen bestehen aus einem strukturierten Anteil, der einen klar definierten Aufbau aus Elementen mit bestimmten, in ihrer Semantik eindeutigen Werten hat, und einem unstrukturierten Anteil, der beliebige Daten (Texte, Bilder, etc.) beinhalten kann. Der strukturierte Anteil erlaubt hierbei die maschinelle Verarbeitung, während der unstrukturierte Anteil zur Informationsübertragung und Kommunikation mit den anderen Prozessteilnehmern genutzt werden kann. So wird es beispielsweise möglich, Informationen oder Nachrichten für andere Prozessteilnehmer abhängig von der Existenz oder den Werten bestimmter Parameter eines Lernobjekts anzuzeigen oder auszublenden: Der unstrukturierte Anteil der Annotation enthält die Information, die präsentiert werden soll, und der strukturierte Anteil der Annotation enthält die maschinell auswertbaren Bedingungen, wann die Information gezeigt werden darf und wann nicht. Nur wenn die Parameter des Lernobjekts diesen Bedingungen in den Annotationen entsprechen, wird die Information präsentiert, ansonsten wird sie ausgeblendet.

---

[69] Die hier genannte Definition ist lediglich ein Beispiel und umfasst beispielsweise keine Statusinformationen wie Autor der Annotation, Erstellungsdatum, etc. In 5.4.1 werden Annotationen ins Datenmodell integriert; die Modellierung von Annotationen wird dann beschrieben.

5.2.3.3 Zugriff

Neben der Kategorisierung von Annotationen nach ihren Aspekten (Datenaspekt sowie sozialer Aspekt) und ihrer Formalisierung können Annotationen auch danach kategorisiert werden, für wen sie bestimmt sind. Je nachdem, ob nur der Autor einer Annotation selbst, eine Gruppe von Prozessteilnehmern oder jeder Prozessteilnehmer die Annotation einsehen kann, ist der Anwendungsbereich der Annotation ein anderer. In folgenden werden die verschiedenen Arten des Zugriffs auf Annotationen diskutiert. Die Einsatzgebiete der verschiedenen Arten von Annotationen werden in Tabelle 5-1 zusammengefasst.

Annotationstyp	Sichtbarkeit	Üblicher Einsatz
Private Annotationen	Verfasser der Annotation	Gedankenstütze für den Verfasser
Gruppenannotationen	Mitglieder der Gruppe	Kommunikation und Koordination innerhalb der Gruppe
Öffentliche Annotationen	jeder Prozessteilnehmer	Hinweise (bzgl. des Inhalts ebenso wie bzgl. des Vorgehens im Prozess), die für alle Prozessteilnehmer von Bedeutung sind

Tabelle 5-1: Verwendung der Annotationstypen entsprechend ihres Zugriffs

Private Annotationen

Private Annotationen sind nur für den Autor der Annotation selbst sichtbar, nicht aber für andere Prozessteilnehmer. Sie dienen als Gedankenstütze, als Merker für den Autor eines Lernobjekts oder eines Kurses und können dabei Aspekte der jeweiligen Lernobjekte (Inhalt, Struktur, Parametrisierung) sowie auch Aspekte des Prozesses (Prozesskoordination, beispielsweise Kontaktdaten für Ansprechpartner, Datenübergabe an Schnittstellen, Termine, etc.) umfassen.

Gruppenannotationen

Gruppenannotationen sind für alle Mitglieder einer bestimmten Gruppe[70] sichtbar. Diese Annotationen behandeln sowohl Aspekte der jeweils annotierten Lernobjekte (inhaltliche Aspekte), als auch Aspekte der Zusammenarbeit innerhalb der Gruppe (soziale Aspekte).

Mittels Gruppenannotationen kann eine Kommunikation in der Gruppe ermöglicht werden, falls eine direkte Kommunikation nicht möglich ist. Lernobjekte werden annotiert, diese Annotationen werden wiederum annotiert, und so fort. Die so entstehenden Strukturen haben große Ähnlichkeit mit News Threads, wie sie aus Usenet News ([Hub] und [USENET]) bekannt sind, und in der Tat können Gruppenannotationen eine ähnliche Funktion wie Usenet News oder ähnliche Diskussionssysteme einnehmen.

Neben der reinen Kommunikation ist mittels Gruppenannotationen auch eine Koordination der Arbeitsschritte, die in der Gruppe erledigt werden müssen, möglich. Dies ist insbesondere dann interessant, wenn die Gruppe Mitglieder mit Aufgaben über den gesamten Prozess hinweg hat: Moduldesigner stimmen ihr Vorgehen mit Kursdesignern auf der einen und

[70] Die Gruppe muss definiert sein. Mitgliedschaft kann sowohl innerhalb einer Menge von Prozessteilnehmern mit gleichen Aufgaben bestehen, beispielsweise eine Gruppe, deren Mitglieder alle Moduldesigner sind, oder die Mitgliedschaft kann sich über den gesamten Prozess erstrecken, beispielsweise alle Prozessteilnehmer (Atomdesigner, Moduldesigner und Kursdesigner), die an der Entwicklung eines bestimmten Kurses beteiligt sind.

Atomdesignern auf der anderen Seite ab, und Kursdesigner geben ihre Anforderungen an die Module direkt an die Moduldesigner in ihrer Gruppe weiter.

Öffentliche Annotationen

Öffentliche Annotationen sind für jeden einsehbar. Ebenso wie private und Gruppenannotationen umfassen sie sowohl Aspekte der Lernobjekte und Kurse, als auch Prozessaspekte. Da sich öffentliche Annotationen an alle Prozessteilnehmer wenden, sind sie weniger für Koordinationsaufgaben geeignet; der Adressatenkreis wäre zu wenig spezifisch, als dass sich jemand angesprochen fühlte. Vielmehr bieten sie sich für Mitteilungen und Hinweise an, die alle Prozessteilnehmer betreffen.

5.3 Annotationen und Lehrmaterialien

Annotationen können genutzt werden, um Schwächen der Beschreibung von Lehrmaterialien mittels Parametern und Constraints auszugleichen und damit deren Nutzungsmöglichkeiten und deren Wiederverwendung zu verbessern. Auch zur Unterstützung der Kommunikation und Koordination im Lehrmaterialerstellungs- und –nutzungsprozess können sie verwendet werden. Insbesondere die Abstimmung von a priori unbekannten Schnittstellenpartnern im Lehrmaterialerstellungsprozess kann mit Annotationen deutlich verbessert werden. Diese beiden Aspekte werden in den beiden folgenden Abschnitten vertieft und es wird gezeigt, wie Annotationen zur Lösung der verschiedenen Problemkreise eingesetzt werden können.

5.3.1 Verbesserung der Wiederverwendbarkeit und Qualität der Lehrmaterialien

5.3.1.1 Diskussion: Qualität im Kontext von Lehrmaterialien

Lehrmaterialien sind in erster Linie Daten. Wenn der Begriff „Qualität von Lehrmaterialien“ präzisiert werden soll, kann man sich ihm über die Qualität von Daten nähern. Das Problem, die Qualität von Daten definieren, einschätzen und bewerten zu können, wird in der Fachwelt als *Datenqualitätsproblem* bezeichnet. In der Literatur ([Bal98], [WaSt96], [DIN95] und [Wall90]) gibt es verschiedene Ansätze, das Datenqualitätsproblem zu erfassen und mittels Kategorisierungen besser handhabbar zu machen. Hier sind insbesondere die Forschungsgebiete Software-Engineering und Data Warehousing sehr aktiv. Von den Ergebnissen dieser Disziplinen sollen nun Anleihen genommen werden.

Eine Beschreibung, was die Qualität von Daten ausmacht, kann in [WaSt96] und [Hel02] nachgelesen werden. Dort werden vier Kategorien von Qualitätsmerkmalen eingeführt, mit deren Hilfe die Qualität von vorliegendem Datenmaterial eingeschätzt werden kann. In Tabelle 5-2 sind die Kategorien mit Beispielen aufgeführt.

Kategorie	Qualitätsmerkmale (Beispiele)
Innere Datenqualität	Genauigkeit, Objektivität, Glaubwürdigkeit, Vertrauenswürdigkeit
Zugriffsqualität	Zugriffsmöglichkeiten, Zugriffssicherheit
Kontextuelle Datenqualität	Relevanz, Mehrwert für den Nutzer, Aktualität, Vollständigkeit, Informationsgehalt
Darstellungsqualität	Interpretierbarkeit, Verständlichkeit, Prägnanz, konsistente Darstellung

Tabelle 5-2: Qualitätsmerkmale von Daten

[RaDo00] geben eine Möglichkeit an, Ursachen von Problemen mit der Datenqualität zu beschreiben. Sie führen hierzu zwei Dimensionen ein: Zum einen unterscheiden sie Einzelquellenprobleme und Mehrquellenprobleme, zum anderen unterscheiden sie zwischen Problemen in der Instanzebene und Problemen in der Schemaebene der Daten. Tabelle 5-3 liefert vereinfachte Beispiele hierfür.

	Einzelquellenprobleme	**Mehrquellenprobleme**
Instanzebene	Fehlende oder falsche Werte	Inkonsistenz
Schemaebene	Mangelhaftes Schema-Design	Homonyme und synonyme Konzept- bzw. Attributbezeichnungen

Tabelle 5-3: Ursachen für mangelnde Datenqualität

Einzelquellenprobleme und Mehrquellenprobleme können auch bei Lehrmaterialien beobachtet werden: Einzelquellenprobleme treten beispielsweise aufgrund isolierter Betrachtung lediglich einer Nutzung oder Nutzungsart auf: Andere Arten der Nutzung als die betrachteten werden nicht berücksichtigt und daher auch nicht bei der Parametrisierung der Kurse, Module und Atome vorgesehen. Ein Beispiel dafür ist eine unzureichende Zuordnung von Lernobjekten zur Nutzungsklasse „Lehre“. Mehrquellenprobleme können insbesondere bei der Integration von Lernobjekten auftreten, beispielsweise wenn mehrere Module von verschiedenen Moduldesignern zu einem komplexeren Modul kombiniert werden sollen. In diesem Fall kann es leicht zu Fehlern kommen, wenn die Moduldesigner unterschiedliche Vorstellungen von der Semantik von Parametern haben.

In Anlehnung an das gerade Gesagte werden folgende Qualitätsdimensionen für Lehrmaterialien definiert:

- Konsistenz von Inhalt und Beschreibung der Lehrmaterialien:

  Lehrmaterialien müssen in Inhalt und Struktur konsistent (also widerspruchsfrei) sein und korrekt und präzise parametrisiert werden (Inhalt und Struktur müssen stimmig zu ihrer Beschreibung durch Parameter sein).

- Inhalt der Lehrmaterialien:

  Lehrmaterialien müssen inhaltlich vollständig sein und die geforderten Lehrinhalte ausreichend gut wiedergeben, also den Inhalt in der geforderten Weise abdecken.

- Nutzung der Lehrmaterialien:

  Die Lehrmaterialien sind in der Lehre und für das Lernen anwendbar und nutzbar. Nutzbarkeit bedeutet für Modul- und Kursdesigner, dass die Lehrmaterialien gut für den Aufbau neuer Module und Kurse verwendet werden können. Für den Kurs-Presenter und den Kurstutor bezieht sich Nutzbarkeit auf die Anwendung in der Lehre. Für die Lerner schließlich heißt Nutzbarkeit, dass die Lehrmaterialien zur Wissensvermittlung – vor allem im Selbststudium – tauglich sind.

- Soziale Aspekte im Zusammenhang mit Lehrmaterialien:

  Wie bereits in 4.5 beschrieben, kann der Prozess zur Lehrmaterialerstellung räumlich und zeitlich verteilt sein, wobei sich die Prozesspartner nicht einmal kennen müssen. In diesem Umfeld ist es wichtig zu wissen, in welchem Bearbeitungszustand ein Lernobjekt ist, von welcher Rolle im Prozess es genutzt wird und was man mit ihm tun kann. Die Prozessbeteiligten müssen ein gemeinsames Verständnis bezüglich des Bearbeitungszustandes und den noch zu erledigenden Arbeiten haben.

Dies erfordert eine Koordination unter den Prozesspartnern. Qualität von Lehrmaterialien hinsichtlich dieser sozialen Aspekte bedeutet damit eine einheitliche, abgestimmte Meinung unter den Prozessbeteiligten hinsichtlich dem Bearbeitungsstand der einzelnen Lehrmaterialien.

Diese Qualitätsdimensionen betreffen einerseits den Inhalt sowie Eigenschaften des Inhalts (beschrieben durch Parameter) der Kurse und Lernobjekte, und andererseits, wie diese Kurse und Lernobjekte genutzt werden. Neben einer reinen Bewertung des Inhalts erlauben diese Qualitätsdimensionen damit auch eine Bewertung der Nutzbarkeit. Ziel des Lehrmaterialerstellungsprozesses ist es, Lehrmaterialien möglichst hochwertig hinsichtlich dieser Kategorien (Tabelle 5-2) und Dimensionen (Tabelle 5-3) zu gestalten.

5.3.1.2 Ausdrucksstärke von Metadaten

Metadaten werden verwendet, um Kurse und Lernobjekte zu beschreiben. Im Datenmodell, das in Kapitel 3 vorgestellt wurde, wird dies durch Parameter und Constraints, die auf den Parametern arbeiten, geleistet; Parameter sind Metadaten zur Beschreibung von Kursen und Lernobjekten, und Constraints beschreiben Einschränkungen und Abhängigkeiten, die für diese Parameter gelten. Die Qualität der Lehrmaterialien und der Grad ihrer Wiederverwendbarkeit hängt zu großen Teilen von einer konkreten Parametrisierung (Beschreibung) der Lehrmaterialien ab, so dass Metadaten hier besonders betrachtet werden müssen. Im Zusammenhang mit Metadaten müssen einige Problemkreise betrachtet werden. Hierbei ist interessant, auf welche der oben genannten Qualitätsdimensionen sie Einfluss haben und wie die Qualität der Kurse und Lernobjekte gezielt verbessert werden kann.

Problemkreis „Subjektivität“

Wie bereits in 3.7 festgestellt, sind Parametrisierung und Definition von Constraints auch immer ein Stück weit vom jeweiligen Bearbeiter (Atom-, Modul- oder Kursdesigner) abhängig. Ein eindeutiger Bezug, ein qualitativer Kategorierahmen ist erforderlich, wie in [Schu-03] zu lesen ist, aber selbst wenn diese Voraussetzungen gegeben sind, bleibt immer noch ausreichend Raum für unterschiedliche Auffassungen und Bewertungen.

Hier sind natürlichsprachliche Erklärungen hilfreich, die die Designer den von ihnen entwickelten Lernobjekten und Kursen als Annotationen mitgeben. In diesen Erklärungen können subjektive Bewertungen und Einschätzungen als solche markiert und klar verständlich erläutert werden. Die Unschärfe beispielsweise einer Bewertung, die einem Parameterwert zugrunde liegt, kann so explizit gemacht werden. Würden keine derartigen Annotationen vorliegen, so wüsste man bei einem Parameter mit potenziell subjektivem Wert nicht, welche persönlichen Einschätzungen und Meinungen des jeweiligen Designers der Bewertung zugrunde lagen und wie diese wiederum zu bewerten sind. Der Parameterwert erhielte den Anschein der objektiven Gültigkeit, obwohl diese Gültigkeit gar nicht vorliegen muss (vgl. auch 5.2.3.2).

Problemkreis „Präzision von Parameterwerten“

Metadaten haben im allgemeinen diskrete Wertemengen. Diese können zwar sehr mächtig sein, bleiben jedoch auf diskrete Stufen beschränkt, von denen man nur eine auswählen kann. Hier liegt ein ähnliches Problem wie beim Punkt „Problemkreis Subjektivität“ vor: Unschärfen, die in den Parameterwerten implizit vorhanden sind – beispielsweise weil beim Setzen der Parameter nicht alle Informationen zur Verfügung standen, weil der passende Parameterwert nicht in der diskreten Wertemenge liegt oder einfach nur weil das Wissen der jeweiligen Designer nicht unendlich ist und er daher den Parameterwert nur unter Berücksichtigung von nicht allen Randbedingungen setzen kann – bleiben unerkannt. Lässt man beispielsweise für den Parameter „Schwierigkeitsgrad“ die Werte „leicht“, „mittel“ und

„schwer“ zu, kann man keine Aussagen wie „leicht, aber in manchen Teilen auch ein wenig schwer“[71] machen. Manche bewusst vagen Aussagen lassen sich mit einigem Aufwand und unter Annahme jeweils zu bestimmender Rahmenbedingen mittels komplexer Constraints ausdrücken, aber das prinzipielle Problem bleibt bestehen.

Auch hier können natürlichsprachliche Erklärungen, die den entsprechenden Lernobjekten in Form von Annotationen beigelegt werden, für Klarheit sorgen: Einerseits können sie verwendet werden, um die Bedingungen und das Umfeld deutlich herauszustellen, unter denen der jeweilige Parameterwert gültig ist, und andererseits können Parameterwerte präzisiert werden, die nicht den aus Sicht der Designer passenden Werten entsprechen. So erlauben auch hier Annotationen wieder, einen möglichen Unsicherheitsgrad von Parameterwerten sowie die Randbedingungen, unter denen die Parameterwerte gesetzt wurden, explizit zu machen, was letztlich eine präzisere Nutzung der Lernobjekte und Kurse erlaubt.

Problemkreis „Komplexität von Constraints“

Constraints werden eingesetzt, um Randbedingungen für Parameterwerte zu beschreiben. Diese Randbedingungen orientieren sich an verschiedenen Aspekten der geplanten späteren Nutzung der Lernobjekte und Kurse, vgl. auch 2.4.2 und 3.5.2.

Constraints können jedoch schnell sehr komplex werden, wenn etwas kompliziertere Sachverhalte modelliert werden sollen; es werden Bedingungen, Fallunterscheidungen oder Verweise auf Werte anderer Parameter nötig. Dann werden diese Constraints für Menschen, die diese eventuell manuell definieren und auswerten, schnell unübersichtlich, fehleranfällig und schwer zu warten[72]. In diesem Fall können natürlichsprachliche Annotationen helfen, den Überblick über die Funktionsweise der Constraints zu bewahren. Die Annotationen werden dann wie ausführliche Kommentare der Constraints genutzt.

Bezüglich der Ausdrucksstärke von Metadaten sind Annotationen insbesondere hilfreich für Daten,

- die sich nicht passend mittels formaler Mittel beschreiben lassen,
- die komplexe Kontrollstrukturen (Fallunterscheidungen, etc.) beinhalten und für Menschen so nur mehr schwer zu verfassen und zu verstehen sind,
- die subjektive Elemente beinhalten,
- die Mehrdeutigkeiten aufweisen, beispielsweise weil sie von Personen stammen, die ein unterschiedliches Verständnis bezüglich einiger Parameter haben.

5.3.1.3 Kontextabhängigkeit von Metadaten

Abgesehen von den konzeptuellen Einschränkungen von Metadaten, die im vorherigen Abschnitt beschrieben wurden, gibt es auch Probleme, die in der räumlichen und zeitlichen Verteiltheit des Prozesses, wie er in Kapitel 4 beschrieben wurde, begründet sind: Je nachdem, wer gerade mit einem Lernobjekt arbeitet, sieht nur einen Ausschnitt der Parameter (vgl.

[71] Man beachte, dass in dieser Aussage gleich drei Unklarheiten verborgen sind: Es ist nicht klar, welche Teile „leicht“ sind, und es ist auch nicht klar, wie kompliziert „ein wenig schwer“ eigentlich genau ist. Des weiteren ist nicht geklärt, für welche Zielgruppe diese Schwierigkeitsangaben gelten, aber dieser Punkt ließe sich durch weitere Parameter genauer spezifizieren.

[72] Hier können geeignete Tools helfen, die ein Stück weit die Komplexität von Constraints vor den Nutzern (Designer wie Endnutzer) verbergen. Bei komplexeren Abhängigkeiten der Parameter voneinander sind jedoch natürlichsprachige Erläuterungen auf jeden Fall hilfreich – unabhängig von der Mächtigkeit der eingesetzten Tools.

Sichten in 4.2.3.2). So arbeitet beispielsweise ein Moduldesigner nur mit den Parametern, die er für seine Arbeit am Modul benötigt[73].

Des weiteren kann ein Prozessteilnehmer spätere Nutzungen der Lernobjekte und Kurse nicht immer kennen – auch dies hängt wieder mit der zeitlichen und räumlichen Verteiltheit des Prozesses zusammen. Er kann die Parameter nur so setzen, wie er annimmt, dass es im weiteren Prozessverlauf sinnvoll ist (Aspekt *Korrektheit*). Beispielsweise hängt die Belegung des Parameters „Schwierigkeitsgrad" eines Lernobjekts von der Zielgruppe ab, welche das fragliche Lernobjekt in einer Lehrveranstaltung oder im Selbststudium nutzt. Der Prozessteilnehmer kann außerdem auch nicht wissen, welche Parameter für den weiteren Prozessablauf von Bedeutung sind und ob auch wirklich alle zukünftig benötigten Parameter gesetzt sind (Aspekt *Vollständigkeit*).

Bei all diesen Unwägbarkeiten ist eine rein maschinelle Auswertung der Parameter kaum sinnvoll; eine Auswertung durch Prozessteilnehmer hingegen ist sehr wohl möglich. Hierzu sind jedoch Erklärungen zu Lernobjekten und Parametern, wie sie mit Hilfe von Annotationen erbracht werden können, sehr hilfreich.

5.3.1.4 Inhaltliche Ergänzungen von Lehrmaterialien

In den letzten Abschnitten wurden Annotationen als ein Mittel beschrieben, um die Definitionen und Belegungen von Parametern zu erklären oder um Unsicherheiten in Parameterwerten explizit zu machen. Annotationen hatten hier immer einen Bezug zu Metadaten, die Lehrmaterialien beschreiben.

Annotationen können jedoch auch genutzt werden, um Lehrmaterialien selbst – und damit nicht mehr Metadaten, sondern konkrete Inhalte – zu ergänzen. Solche Annotationen werden im folgenden *inhaltsergänzende Annotationen* genannt. Ziel von inhaltsergänzenden Annotationen ist es, den Inhalt der betreffenden Lehrmaterialien mit weiteren Informationen anzureichern, ohne jedoch den Lesefluss der Lerner zu unterbrechen. Es ist sinnvoll, von dieser Möglichkeit nur bei optionalen Informationen, die neben dem eigentlichen Lehrinhalt genutzt werden können, gebrauch zu machen.

Aufgabe	Modul-designer	Kurs-designer	Kurs-Presenter	Kurstutor	Lerner
Design	-	-			
Entwicklung	Annotiert	Annotiert			
Adaptieren/ Customization			Annotiert	Annotiert	Annotiert
Präsentieren			Annotiert	Annotiert	
Lernen					Annotiert

Tabelle 5-4: Einfügen von inhaltsergänzenden Annotationen[74]

Inhaltsergänzende Annotationen können natürlichsprachlicher Freitext (gekapselt in einem Atom) sein, aber ebenso auch Atome mit nicht-textuellem Inhalt oder Module. Dienen bereits existierende Lernobjekte (Atome oder Module) als inhaltsergänzende Annotationen für

[73] Dies ist zwar einerseits gewollt, um die Komplexität des Lehrmaterialerstellungsprozesses unter Kontrolle zu halten, und aus genau diesem Grund wurden in 4.2.3.2 auch die Sichten auf den Prozess eingeführt. Aber andererseits darf man nicht vergessen, dass man so ein Stück weit die Kontrolle abgibt: Man sieht nicht mehr alle Parameter und kann deren Werte so nicht mehr prüfen.

[74] Aufgaben, die eine Rolle nicht ausführt, sind ausgegraut.

andere Lernobjekte, lassen sich die Beziehungen zwischen diesen Lernobjekten analog zu anderen Arten von Annotationen wieder mittels Assoziationen realisieren, wie später beschrieben. Dass es sich bei diesen Lernobjekten um Annotationen für andere Lernobjekte handelt, wird dann in der Assoziation entsprechend über Parameter festgelegt.

Inhaltsergänzende Annotationen werden meist nicht während der Designphase eines Moduls oder Kurses festgelegt, sondern erst dann, wenn das Modul oder der Kurs mit konkreten Inhalten gefüllt oder genutzt wird. Tabelle 5-4 zeigt, wann inhaltsergänzende Annotationen eingefügt werden.

5.3.2 Prozessoptimierung

5.3.2.1 Diskussion: Optimierung eines Prozesses

Optimierung und damit Qualitätsverbesserung eines Prozesses umfasst mehrere Problemkreise. Einmal müssen Durchlaufzeiten minimiert werden, um möglichst schnell Lehrmaterialien bereitstellen zu können. Des weiteren gilt es, Medienbrüche aus Effizienzgründen möglichst zu vermeiden, und schließlich muss die Zusammenarbeit der einzelnen Prozessteilnehmer so effizient wie möglich laufen. Diese Problemkreise werden nun besprochen.

Problemkreis „Effizienter und schneller Prozessdurchlauf“

Ein effizienter und schneller Prozessdurchlauf stellt sicher, dass aktuelle Lehrmaterialien den Nutzern rechtzeitig zur Verfügung stehen. Die einzelnen Prozessschritte müssen hierzu eng aufeinander abgestimmt sein und miteinander harmonieren. Schnittstellenkontrakte zwischen den Prozesspartnern müssen unbedingt eingehalten werden, um Verwirrung zu vermeiden.

Je schneller ein Prozess durchlaufen wird, desto größer ist allerdings die Gefahr, dass einzelne Prozessschritte nicht mehr mit der nötigen Sorgfalt durchgeführt werden. Die Qualität des Ergebnisses – hier: der Lehrmaterialien – würde leiden, wodurch der Lernerfolg und die Wiederverwendbarkeit der Lehrmaterialien gefährdet sein kann. Die Geschwindigkeit des Prozessdurchlaufs darf nicht auf Kosten der Qualität der Lehrmaterialien gehen. (Für Kriterien der Qualität von Lehrmaterialien vgl. 5.3.1.1.) Es ist hier sorgfältig zwischen beiden Anforderungen – Geschwindigkeit und Qualität – abzuwägen.

Problemkreis „Medienbrüche“

Durch Medienbrüche kann es zu folgenden Problemen im Prozessablauf kommen: Informationen gehen verloren oder werden verfälscht, und durch die zwischen den jeweiligen Prozessschritten nötigen Konvertierungen zwischen Medien geht Zeit und Geld verloren. Um den Prozess möglichst ressourcenschonend (Zeit, Kosten) durchführen zu können, müssen daher Medienbrüche so weit wie möglich vermieden werden.

Medienbrüche entstehen oft an Stellen im Prozess, wo die Abstimmung zwischen den Projektpartnern nicht ideal läuft bzw. wo sich die technischen Rahmenbedingungen (verwendete Software, verschiedene Datenformate und Kodierungen wie ASCII, UTF-8, EBCDIC, etc.) ändern. Für einen idealen Prozess müssen damit folgende Bedingungen gelten:

- Über alle Prozessschritte hinweg müssen die gleichen technischen Rahmenbedingungen (vgl. oben) gelten. Dies betrifft insbesondere das Datenmodell, das dem Prozess zugrunde liegt.
- Die Übergänge von einem Prozessschritt zu den nächsten Prozessschritten müssen reibungslos erfolgen; Schnittstellenvereinbarungen (beispielsweise wann welche Lernobjekte an wen zu übergeben sind), Synchronisationspunkte (welche Lernobjekte vorliegen müssen, damit der nächste Prozessschritt durchgeführt werden kann) und Kommunikationswege müssen eingehalten werden.

Problemkreis „Koordination der Prozessteilnehmer“

Eine reibungslose und effiziente Koordination der Prozessteilnehmer untereinander ist die wesentliche Voraussetzung für eine wirkungsvolle Lösung der beiden oben genannten Problemkreise. Ohne eine gut funktionierende Koordination der einzelnen Prozesspartner ist ein effizienter Prozessdurchlauf nicht denkbar. Analog können Medienbrüche oft vermieden oder zumindest besser beherrschbar werden, wenn die jeweils betroffenen Prozesspartner ihre Schritte sowie die Modalitäten des Datenaustausches absprechen.

Um eine effiziente Koordination der Prozessteilnehmer untereinander gewährleisten zu können, muss eine entsprechende Kommunikationsinfrastruktur installiert sein. Dies betrifft einerseits technische Aspekte wie beispielsweise die Verfügbarkeit von Telefon, E-Mail oder Instant Messaging, aber auch soziale und organisatorische Aspekte wie beispielsweise eine Regelung, wer bei welchen Ereignissen wen benachrichtigt (vgl. 5.3.2.2).

5.3.2.2 Kommunikation und Koordination

Eine funktionierende Kommunikation ist die Grundlage für rasche, effiziente Übergänge zwischen den Prozessschritten. Selbst wenn – aus welchen Gründen auch immer – Medienbrüche zwischen den Prozessschritten auftreten sollten, können deren Auswirkungen bei entsprechender Kooperation der Prozesspartner überschaubar gehalten werden.

Aufgrund des verteilten Charakters des Lehrmaterialerstellungsprozesses hinsichtlich Zeit, Raum und meist auch Organisation ist jedoch ein direkter persönlicher Kontakt zwischen den unmittelbaren Schnittstellenpartnern im Prozess meist nicht oder nur unter Schwierigkeiten möglich; vielfach kennen sich die Prozesspartner nicht einmal oder haben mangels Kontaktdaten keine Möglichkeit zur unmittelbaren Kommunikation (vgl. 4.5.2).

Eine Möglichkeit, trotzdem einen Informationsfluss zwischen den Prozesspartnern zu etablieren, bieten Annotationen, die den Lernobjekten und Kursen angeheftet werden, auf die sie sich beziehen. Ein Beispiel soll die Vorgehensweise verdeutlichen:

> Der übliche Weg, eine Überarbeitung an einem Dokument zu erreichen, ist es, die Überarbeitungsvorschläge in einer E-Mail zu beschreiben, das fragliche Dokument der E-Mail als Attachment beizufügen und die E-Mail samt dem Dokument an den Autor des Dokuments zu schicken. Der Nachricht *an einen wohlbekannten Adressaten* ist das zu überarbeitende Dokument angeheftet; das Übertragungsmedium für die Information ist die E-Mail.
>
> Nutzt man Annotationen zur Übermittlung eines Überarbeitungsvorschlags, ist dies genau umgekehrt. Die Überarbeitungsvorschläge werden direkt an das zu überarbeitende Dokument angehängt, und wenn das Dokument in Zukunft betrachtet wird, kann die Annotation gelesen werden. Zentral ist hierbei, dass nur eine Instanz des Dokuments existiert, die von allen Autoren und Nutzern gemeinsam genutzt wird und an die alle Annotationen angehängt werden. Das Übertragungsmedium für die Information ist hier das *gemeinsam genutzte* Dokument.

Annotationen ermöglichen so eine indirekte Kommunikation, indem sie mit gemeinsam genutzten Objekten verbunden werden. Annotationen können auf diese Weise zur Verbesserung der Qualität von Lernobjekten und Kursen sowie zur Prozesssteuerung und –optimierung genutzt werden. Werden Annotationen zur Kommunikation und zur Koordination zwischen Prozessteilnehmern genutzt, müssen zwei Probleme geklärt werden: Wie weiß der Adressat, an den sich eine Annotation richtet, dass er eine Nachricht bekommen hat? Und welche Arten von Informationsübermittlung sind mit Annotationen überhaupt möglich? Auf diese Fragen wird in den folgenden Absätzen eingegangen.

Awareness

Wenn eine Annotation an ein Lernobjekt oder einen Kurs angehängt wird, müssen die Prozessteilnehmer auf die eine oder andere Weise informiert werden. Hierzu gibt es mehrere Möglichkeiten:

- Aktive Benachrichtigung:

  Die Interessenten (beispielsweise Moduldesigner oder Kursdesigner) werden aktiv benachrichtigt. Beispielsweise könnte bei Erstellen, Ändern oder Löschen einer Annotation eine E-Mail verschickt werden.

  Diese Art der Benachrichtigung kann bei häufiger Nutzung von Annotationen zur Belastung für die Interessenten werden. Insbesondere bei öffentlichen Annotationen, bei denen ein vergleichsweise hohes Maß an Änderungen angenommen wird, sollte den Interessenten die Möglichkeit eingeräumt werden, aktive Benachrichtigungen zu deaktivieren. Alternativ könnte den Interessenten eine Möglichkeit geboten werden, Benachrichtigungen im System zu sammeln und nur beispielsweise einmal am Tag eine Zusammenfassung über alle Benachrichtigungen zu erhalten.

- Passive Benachrichtigung: Markierung des Lernobjekts bzw. des Kurses:

  Das Lernobjekt bzw. der Kurs, der annotiert wurde, wird markiert; der annotierte Inhalt des Lernobjekts bzw. des Kurses wird farblich, mittels grafischer Symbole oder mittels besonderem Layout hervorgehoben. Sobald die Interessenten das Lernobjekt bzw. den Kurs nutzen oder bearbeiten, werden sie auf die Markierung aufmerksam und können die Annotation einsehen.

  Diese Art der Benachrichtigung ist weniger aufdringlich als die Formen der aktiven Benachrichtigung, geht aber mit dem Risiko einher, dass Annotationen erst spät oder gar nicht wahrgenommen werden.

- Keine Benachrichtigung:

  Interessenten wollen über bestimmte Änderungen an Lernobjekten oder Annotationen nicht benachrichtigt werden.

  Um den Designern und Nutzern nicht jede mögliche Information, die sie vielleicht gar nicht benötigen, aufzuzwingen, muss es die Möglichkeit geben, von der Benachrichtigung ausgenommen zu werden.

Um festzulegen, wer beim Erstellen, Ändern oder Löschen einer Annotation benachrichtigt wird bzw. wer die zur Annotation gehörende Markierung sehen kann, wird wieder auf die Kategorisierung von Annotationen in 5.2.3 zurückgegriffen: Private Annotationen sieht nur ihr eigener Autor, Gruppenannotationen sehen nur die Mitglieder der Gruppe und öffentliche Annotationen können von jedem eingesehen werden. Entsprechend der Sichtbarkeit findet auch die Benachrichtigung – falls eine solche von den Interessenten gewünscht ist – statt.

Für mehr Informationen zum Thema „Awareness“ sei der interessierte Leser an dieser Stelle auf [BoSc00] und [Bür99] verwiesen.

Informationsspeicherung und -übermittlung

Annotationen können für verschiedene Arten von Informationsspeicherung und –übermittlung genutzt werden (vgl. auch Abb. 5.2). *Informationsspeicherung* betrifft inhaltliche oder strukturelle Aspekte des Lernobjekts oder des Kurses, an den die Annotation angeheftet ist, und dient im wesentlichen dem Autor der Annotation als Merker oder Gedankenstütze; in

diesem Fall steht die Verbesserung der Qualität des Lernobjekts im Vordergrund, was bereits in 5.3.1 behandelt wurde und daher hier nicht weiter betrachtet werden soll.

Aspekte der *Informationsübermittlung und Kommunikation* können bei Gruppenannotationen und öffentlichen Annotationen beobachtet werden. Die Kommunikation dreht sich dabei üblicherweise um das Lernobjekt bzw. den Kurs, an den die Annotation geheftet ist. Die Themen können hierbei das Lernobjekt bzw. den Kurs selbst betreffen, beispielsweise Korrekturen an Inhalt oder Struktur, die an andere Prozessteilnehmer übermittelt werden sollen oder das weitere Vorgehen mit diesem Lernobjekt betreffen, beispielsweise eine To-Do-Liste, in der aufgelistet ist, welcher Prozessteilnehmer noch welche Aufgaben erledigen muss, bis das Lernobjekt wirklich nutzbar ist. Auf diese Weise kann auch ohne unmittelbare Kenntnis der Kommunikationspartner ein gemeinsames Vorgehen zur Erstellung und Qualitätssicherung von Lernobjekten und Kursen realisiert werden.

5.3.2.3 Iteration von Teilprozessen

In 4.2 wurde der Prozess der Lehrmaterialerstellung als eine lineare Abfolge einzelner Prozessschritte dargestellt. Diese Darstellung ist ausreichend, um das grundlegende Verständnis der Lehrmaterialerstellung herzustellen. In der Realität ist die Linearität der Prozessschritte so jedoch meist nicht gegeben. Statt dessen werden die einzelnen Prozessschritte iterativ durchlaufen und die Lehrmaterialien von Iteration zu Iteration (falls nötig) korrigiert und präzisiert. Beispielsweise beginnt man beim Design eines neuen Moduls mit dem initialen Entwurf und der Parametrisierung. Durch die Auswertung der Parameter und dem Einsetzen von entsprechenden Subelementen kommt man zu einem konkreten Modul, das von dessen weiteren Nutzern – Kursdesignern, Kurs-Presenter, etc. – korrekturgelesen und kritisch hinterfragt wird. Die Kritik wird dann dem Moduldesigner wieder mitgeteilt, was zu einer Korrektur des Moduls führt, und so fort, vgl. Abbildung 5-4.

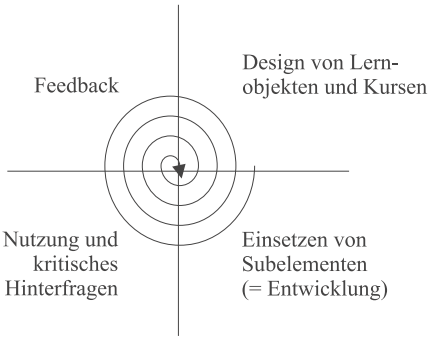


Abbildung 5-4: Iteratives Durchlaufen von Prozessschritten

Das Ausgangsmodul wird so Schritt für Schritt verfeinert und verbessert. Diese Korrekturschritte sind aber nur dann möglich, wenn es eine Kommunikation entgegen der Prozessrichtung gibt, so dass nachfolgende Prozesspartner Feedback geben können. Das Feedback ist der Ausgangspunkt jeglicher Überarbeitung von Lernobjekten und Kursen (vgl. 4.5.2.2 und 4.6.2).

Annotationen sind nun eine Möglichkeit, Feedback zu geben; die Anmerkungen eines Prozesspartners werden – ähnlich einem Zettel – einem Lernobjekt oder einem Kurs angeheftet, so dass jeder, der das Lernobjekt oder den Kurs betrachtet und die entsprechenden Rechte hat, Zugriff auf das Feedback erhält. Für Feedback mittels Annotationen ist auch keine direkte Kommunikation zwischen den beteiligten Prozesspartnern nötig, was insbesondere dann Vorteile bringt, wenn eine direkte Kommunikation nur schwer oder gar nicht möglich ist.

5.4 Erweiterungen

Damit Annotationen genutzt werden können, um die in 5.3 genannten Probleme zu lösen, müssen das Datenmodell sowie der Lehrmaterialerstellungs- und -nutzungsprozess geeignet erweitert werden. Eine Erweiterung des Datenmodells ist nötig, da Annotationen an die verschiedenen Prozessteilnehmer verteilt und für spätere Nutzung und Bearbeitung persistent abgelegt sein müssen. Der Prozess muss erweitert werden, da der Informationsfluss, wie er durch Annotationen realisiert wird, bislang im Prozess nicht vorgesehen ist. Insbesondere wenn Informationen entgegen der Prozessrichtung fließen (vgl. Feedback-Annotationen in 5.4.2.3), muss dies im Prozess berücksichtigt werden.

5.4.1 Erweiterung des Datenmodells

5.4.1.1 Anforderungen

Grundsätzlich stellt sich die Frage, ob Annotationen mit den bereits im Datenmodell vorhandenen Elementen modelliert werden oder als eigenständige Elemente neben den bereits definierten existieren sollen; Annotationen sind schließlich keine Lehrmaterialien. Dies ist nur auf den ersten Blick richtig, denn Annotationen von Atomdesignern, Moduldesignern und Kursdesignern müssen getrennt von Annotationen von Nutzern – Kurs-Presentern, Kurstutoren und Lernern – betrachtet werden. Im folgenden werden daher Annotationen von Nutzern unterschieden von Annotationen, die rein prozesssteuernde Aufgaben haben. Für beide Gruppen von Annotationen wird geprüft, ob eine Modellierung mit den bereits bestehenden Mitteln des Datenmodells sinnvoll ist.

Annotationen von Nutzern entstehen erst in der intensiven Beschäftigung mit den Lehrmaterialien. Die Nutzer formulieren ihre Gedanken zum Lehrstoff in Annotationen und hängen diese quasi als Ergänzung an die jeweiligen Lehrmaterialien. Bei Lernern insbesondere bewirkt das Aufschreiben der Gedanken eine Reflexion des Lehrstoffs und damit dessen Vertiefung. Annotationen haben somit eine wichtige Funktion im Lernen und Verstehen des Lehrstoffs und können quasi als Lehrmaterialien betrachtet werden. Für die Teilnehmer am Lehrmaterialerstellungsprozess, also Atomdesigner, Moduldesigner und Kursdesigner, haben diese Annotationen den angenehmen Nebeneffekt, dass sie Aufschlüsse darauf geben, wo ihre Atome, Module und Kurse noch zu verbessern, zu korrigieren und zu präzisieren sind. Die Integration von Annotationen von Nutzern in das Datenmodell ist unter diesem Gesichtspunkt gesehen sinnvoll.

Für Annotationen hingegen, die rein prozesssteuernde Aufgaben ohne inhaltlichen Bezug zu Lernobjekten haben (beispielsweise Annotationen, in denen andere Designer um ihre Meinung gebeten werden, oder die zur Absprache des weiteren Vorgehens bei der Erstellung eines Kurses dienen), ist eine Integration in das Datenmodell hingegen nicht unbedingt notwendig. Die Identifikation jedoch, welche Annotation rein prozesssteuernde Aufgaben hat und ob daneben noch weitere Aspekte (beispielsweise Korrekturvorschläge bzgl. des Inhalts von Lernobjekten oder Kursen) adressiert werden, ist nicht immer einfach.

Um Schwierigkeiten bei der Einschätzung der Bedeutung von Annotationen zu umgehen, werden generell alle Annotationen – egal ob sie von (End-)Nutzern oder Designern verfasst werden – in das Datenmodell integriert. Hierbei werden die vorhandenen Elemente des Datenmodells genutzt.

Die Integration von Annotationen in das bestehende Datenmodell erlaubt es, die Vorteile des Datenmodells auch für Annotationen zu nutzen. Insbesondere der Aspekt der Durchgängigkeit über den gesamten Prozess ist hier von Bedeutung, denn Annotationen, die nicht über

Teilprozessgrenzen hinweg genutzt werden können, sind wertlos. Eine Integration von Annotationen ins Datenmodell muss folgenden Bedingungen genügen:

- Durch Annotationen darf die Parametrisierung der Kurse, der Lernobjekte bzw. der Beziehungen zwischen den Lernobjekten nicht verändert werden. Andernfalls würden die Nutzungsmöglichkeiten der Lernobjekte bzw. Kurse verändert.
- Annotationen müssen in gewisser Weise eigenständig und von den Lernobjekten, mit denen sie verbunden sind, unabhängig sein:
    - Wird das Lernobjekt verändert, bleibt die Annotation unverändert.
    - Erst wenn das Lernobjekt gelöscht wird, werden auch die Annotationen gelöscht, da sie ohne Bezugspunkt, also „ihr“ Lernobjekt, nicht existieren können (vgl. 5.2.1.1).
- Allerdings stehen Annotationen, gerade wenn sie von den Nutzern im Lehrmaterialerstellungs- und –nutzungsprozess oder den Kursdesignern erstellt werden, in engem Zusammenhang mit dem gerade genutzten Kurs. Verfasst ein Nutzer (Kurs-Presenter, Kurstutor oder Lerner) oder ein Kursdesigner eine Annotation, so muss vermerkt werden, in Zusammenhang mit welchem Kurs die Annotation verfasst wurde.

5.4.1.2 Modellierung von Annotationen

Im folgenden wird beschrieben, wie die im letzten Abschnitt genannten Anforderungen mit Mitteln des in Kapitel 3 vorgestellten Datenmodells umgesetzt werden können. Hierzu werden Annotationen direkt auf Elemente des Datenmodells abgebildet; es ist nicht nötig, ein neues Objekt „annotation“ einzuführen. Je nachdem, wie stark die Annotationen strukturiert sind, werden sie durch unterschiedliche Elemente des Datenmodells modelliert.

- Strukturierte Annotationen:

  Strukturierte Annotationen haben große Ähnlichkeit mit Metadaten. Es liegt nahe, strukturierte Annotationen als ein leeres Modul (ohne Submodule oder Atome) zu modellieren, das lediglich die Metadaten aufnimmt: Die Metadaten, die in der strukturierten Annotation enthalten sind, werden unmittelbar auf die Parameter des Moduls abgebildet. Attributnamen der Metadaten werden in Parameternamen des Moduls übernommen, Attributwerte in Parameterwerte.

- Semistrukturierte Annotationen und freiformulierte Annotationen:

  Semistrukturierte Annotationen und freiformulierte Annotationen werden auf ein Modul *M* abgebildet, das Subelemente (Module oder Atome) enthält. Die eingebetteten Subelemente enthalten dabei den Inhalt der Annotationen:

    - Bereits existierende Subelemente (Module und Atome) können als Annotationen verwendet werden, indem sie mit Modul *M* mittels einer Assoziation verbunden werden.
    - Freiformulierte Texte werden in einem Atom gekapselt, das mittels einer Assoziation[75] in Modul *M* integriert wird.

[75] Es ist möglich, hier statt einer Assoziation auch eine Komposition zu verwenden. Um jedoch die Erstellung von Annotationen im Datenmodell so einheitlich wie möglich zu gestalten, sollen hier, ebenso wie in anderen Fällen, Assoziationen verwendet werden.

Im Fall von semistrukturierten Annotationen werden noch die Beschreibungen (Metadaten) der Annotation wieder direkt auf die Parameter des Moduls abgebildet (vgl. Tabelle 5-5). Dies geschieht analog zu den strukturierten Annotationen.

Annotationen sind damit Sonderformen von Modulen. Sie gliedern sich entsprechend Abbildung 5-5 in das Datenmodell ein.

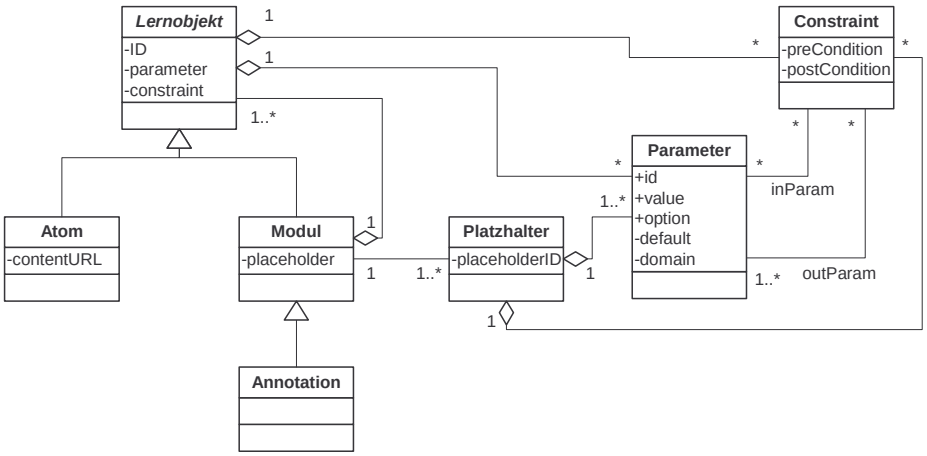


Abbildung 5-5: Eingliederung von Annotationen in das Datenmodell

Eine Annotation hat dabei Parameter gemäß Tabelle 5-5. Weitere Parameter werden im Lauf dieser Arbeit vorgestellt, wenn sie benötigt werden.

Parametername	Bedeutung	Beispiel
created	Zeitpunkt der Erzeugung der Annotation	14.06.2004
created-by	ID des Nutzers, der die Annotation erstellt hat	Weilun
last-modified	Zeitpunkt der letzten Änderung der Annotation	20.06.2004
last-modified-by	ID des Nutzers, der die Annotation zuletzt geändert hat (beispielsweise Änderungen an Parametern)	Weilun
format	Struktur der Annotation, vgl. 5.2.3.2	Freiformulierte Annotation
access	Angabe, wer die Annotation sehen darf	Privat / Gruppe / Öffentlich
ownerID	Name des Autors dieser Annotation	Weilun
contact	Kontaktdaten des Autors, beispielsweise E-Mailadresse, etc.	zhuang@cs.tum.edu
courseID	Identifikator des Kurses, bei dessen Bearbeitung bzw. Nutzung die Annotation verfasst wurde. Kann undefiniert sein.	1234

Tabelle 5-5: Parameter von Annotationen

5.4.1.3 Verbindung von Annotationen mit Lernobjekten

Annotationen haben einen Inhalt und einen Bezug zu einem Lernobjekt oder einem Kurs. Im letzten Absatz wurde die Datenstruktur von Annotationen definiert: Der Inhalt von Annotationen wird als Module und Atome realisiert. Eine Zuordnung einer Annotation zu einem Lernobjekt fehlt bislang.

Diese Zuordnung wird mittels Assoziationen realisiert. Assoziationen zwischen Lernobjekten sind bereits im Datenmodell verankert (vgl. 3.3.5) und können auf einfache Weise für die Nutzung mit Annotationen erweitert werden.

Jede Assoziation hat einen Parameter „kind“, der angibt, welche semantische Beziehung zwischen den beiden durch die Assoziation verbundenen Lernobjekten besteht (vgl. 3.4.1.1). Zur Menge der bereits definierten Typen wird nun ein neuer Typ „annotiert“ hinzugenommen. Mit diesem Typ werden Beziehungen zwischen einem Lernobjekt und der ihm angehefteten Annotation gekennzeichnet. Die Assoziation zwischen einem Lernobjekt und einer Annotation hätte dann folgenden Parameter „kind“:

```
<parameter>
   <id>kind</id>
   <domain>Type</domain>
   <value>annotiert</value>
   <default>none</default>
   <option>mandatory</option>
</parameter>
```

Neben dem allgemeinen Typ „annotiert“ können noch weitere, speziellere Inhaltstypen von Annotationen modelliert werden, die mittels des Parameters „kind“ gesetzt werden können. Der Typ „annotiert“ wird so verfeinert. Dadurch kann die inhaltliche Aufgabe bzw. die Nutzung der Annotation näher spezifiziert werden. Tabelle 5-6 liefert Werte für Inhaltstypen.

Typ	Bedeutung
diskussion	Diskutiert Fragen des annotierten Lernobjekts. Ähnlich den Threads in Usenet News.
verweis	Verweist auf andere Lernobjekte, ähnlich einem Literaturverweis.
erklaerung	Erklärt ein Lernobjekt genauer. Dies kann den Inhalt des Lernobjekts, aber auch dessen Beschreibung mittels Parametern und Constraints betreffen.
korrektur	Regt eine Korrektur eines Lernobjekts an.
notiz	(Private) Gedankenstütze.

Tabelle 5-6: Inhaltstypen von Annotationen

Neben der Angabe des Typs der Annotation muss auch noch vermerkt werden, ob die Annotation bei der Nutzung oder Bearbeitung eines bestimmten Kurses erstellt wurde, wie dies bei allen Annotationen von Kursdesignern, Kurs-Presentern, Kurstutoren und Lernern der Fall ist. Dann kann es nämlich sein, dass die Annotation nur in Zusammenhang mit einem bestimmten Kurs von Bedeutung ist. Hierzu wird den Assoziationen neben dem Parameter „kind“ noch ein Parameter „courseID“ zugewiesen. Der Parameter „courseID“ enthält den Identifikator des Kurses, in dessen Zusammenhang die Annotation gemacht wurde: Bei Kursdesignern, Kurs-Presentern, Kurstutoren und Lernern ist dieser Parameter entsprechend gesetzt, bei Atom- und Moduldesignern, deren Annotationen meist keinen Bezug auf einen

bestimmten Kurs haben, bleibt dieser Parameter ungesetzt. Der Parameter „courseID“ sieht folgendermaßen aus:

```
<parameter>
   <id>courseID</id>
   <domain>Identifier</domain>
   <value>MeineID</value>
   <default>none</default>
   <option>optional</option>
</parameter>
```

Auf diese Weise kann jede Annotation durch ein Modul, das evtl. Inhalt enthält, dargestellt werden. Die Integration ins Datenmodell geschieht auf natürliche Weise, indem bereits vorhandene Elemente des Datenmodells zur Modellierung der Annotationen genutzt werden. Da das Datenmodell über alle Prozessschritte durchgängig ist, ist so auch für die Durchgängigkeit der Modellierung der Annotationen gesorgt.

5.4.2 Erweiterung des Prozesses

5.4.2.1 Anforderungen

Da ein wesentliches Problem des bestehenden Lehrmaterialerstellungs- bzw. –nutzungsprozesses die Kommunikation und Koordination der Prozessteilnehmer untereinander ist, muss der Prozess um Maßnahmen erweitert werden, die eine Kommunikation und Koordination ermöglichen und so helfen, die Qualität der Lehrmaterialien zu erhöhen und ihre Wiederverwendbarkeit zu steigern. In 5.3.2 wurden Annotationen als Mechanismus hierfür vorgeschlagen.

Kommunikation und Koordination werden in verschiedenen Phasen des Lehrmaterialerstellungsprozesses benötigt. Eine enge *Koordination* zwischen den Prozessteilnehmern ist insbesondere zwischen Atomdesignern, Moduldesignern, Kursdesignern und evtl. Kurs-Presentern von Bedeutung. Ohne eine enge Zusammenarbeit zwischen Vertretern dieser Rollen ist die Erstellung qualitativ hochwertiger, vielfach wiederverwendbarer Lehrmaterialien nicht möglich. Die Notwendigkeit einer engen Kooperation und Abstimmung sehen daher am ehesten die Vertreter dieser Rollen.

Kommunikation hingegen ist im gesamten Prozess von großer Bedeutung. Auf der Seite der Lehrmaterialersteller ist Kommunikation nötig, da Kommunikation die Grundlage jeglicher Koordination ist; ohne Kommunikation (direkt oder indirekt) findet keine Koordination statt. Auf der Seite der Lehrmaterialnutzer hat Kommunikation im wesentlichen eine didaktische Funktion: Durch die Kommunikation mit anderen Lernern oder mit Kurstutoren oder Kurs-Presentern wird das Erlernte reflektiert, wiederholt und vertieft. Die Kommunikation zwischen Lehrmaterialnutzern und Lehrmaterialerstellern wiederum dient der Qualitätssicherung der Lehrmaterialien, beispielsweise wenn Fehler korrigiert werden müssen oder Anregungen zur Gestaltung von Kursen diskutiert werden.

Damit eine Kommunikation zwischen den verschiedenen Prozessteilnehmern möglich wird, müssen Annotationen nicht nur zu gelesen, sondern auch selbst wieder annotiert werden können. So entstehen Strukturen ganz ähnlich den Threads in Usenet News.

Mit Hinblick auf Kommunikation und Koordination muss der erweiterte Prozess folgende Eigenschaften erfüllen:

- Jede Rolle im Prozess muss Annotationen erstellen können.
- Jede Rolle im Prozess muss Annotationen aller anderen Rollen lesen und – je nach Rolle – auch annotieren können, sofern sie die hierfür nötigen Rechte hat.

Dies gilt nicht für Lerner. Diese können nur die Annotationen lesen, die von Lernern selbst verfasst wurden. Sie können auch nur solche „Annotations-Threads“ lesen, die ursprünglich von einem Lerner initiiert wurden. Andere Annotationen dürfen sie nur dann sehen, wenn diese explizit öffentlich gemacht wurden.

- Annotationen, die in einem Prozessschritt *n* angelegt wurden, müssen auch in Prozessschritten *m* lesbar und annotierbar sein, wobei Prozessschritt *m* im Prozess vor Prozessschritt *n* liegt.

  Dies bedeutet nicht, dass Prozessschritt *m* zeitlich vor Prozessschritt *n* liegt. Aufgrund des iterativen Charakters des Prozesses (vgl. 5.3.2.3) kommt es jedoch vor, dass Prozessschritte mehrmals nacheinander durchlaufen werden (siehe auch Abbildung 5-4).

Neben der generellen Eignung von Annotationen für die Kommunikation muss insbesondere auf ihre Funktion eingegangen werden, denn je nach Prozessschritt werden Annotationen aus unterschiedlichen Gründen und mit unterschiedlicher Absicht erstellt. Um die Funktion von Annotationen genauer beschreiben zu können, werden in den folgenden Abschnitten didaktische Annotationen, Feedback-Annotationen und soziale Annotationen eingeführt.

5.4.2.2 Didaktische Annotationen zur Unterstützung des Lernvorgangs

Didaktische Annotationen werden von Lernern, Kurs-Presentern und Kurstutoren verfasst, wenn sie sich Kommentare zu Lehrmaterialien machen. Die Kommentare werden als Annotationen dann direkt an die Lehrmaterialien angehängt.

Didaktische Annotationen haben nicht mit der Wiederverwendung von Lehrmaterialien zu tun, sondern unterstützen den Lernprozess: In ihnen beschreiben die Lerner eigene Beispiele, erklären mit eigenen, für sie klareren Worten einen Sachverhalt, etc. Didaktische Annotationen haben damit eine rein didaktische Funktion. Je nach Art der Annotation – private Annotation, Gruppenannotation oder öffentliche Annotation – haben auch andere Lerner Zugriff und können die Annotationen für ihr eigenes Studium nutzen. Gruppenannotationen und öffentliche Annotationen können auch für Diskussionen mit anderen Lernern, aber auch mit Kurs-Presentern und Kurstutoren genutzt werden, wodurch die Lerninhalte reflektiert und vertieft werden. Da didaktische Annotationen für die Verbesserung der Qualität der Lehrmaterialien oder des Lehrmaterialerstellungsprozesses keine Bedeutung haben, sondern „nur“ einen Beitrag zum Lernerfolg liefern, sollen sie in dieser Arbeit nicht weiter betrachtet werden.

5.4.2.3 Feedback-Annotationen zur Erhöhung der Lehrmaterialqualität

Generelle Eigenschaften von Feedback-Annotationen

Feedback-Annotationen werden eingesetzt, um Korrekturen an Lernobjekten und Kursen vorzuschlagen und zu diskutieren. Der Informationsfluss läuft im Fall von Feedback-Annotationen immer entgegen der Prozessrichtung; er geht immer von den Nutzern eines Lernobjekts bzw. eines Kurses[76] zu den Prozesspartnern, die für das annotierte Lernobjekt bzw. den annotierten Kurs verantwortlich sind. Je nachdem, ob Endnutzer (Kurs-Presenter, Kurstutoren und Lerner) oder Designer (Atomdesigner, Moduldesigner und Kursdesigner) ein Lernobjekt annotieren, ist die Zielrichtung eine andere:

[76] Dies sind alle Prozessteilnehmer, die mit einem Lernobjekt bzw. einem Kurs arbeiten, also nicht nur (End-) Nutzer, sondern auch Modul- oder Kursdesigner.

- Endnutzer annotieren Lernobjekte und Kurse:

  Ziel ist die Erhöhung der Qualität der Lernobjekte und Kurse, also ihre Verbesserung in Hinblick auf ihre Nutzung, beispielsweise ihre Anwendung in der Lehre (didaktische Funktion, fachliche Korrektheit, etc.).

- Designer annotieren Lernobjekte und Kurse:

  Ziel ist die Verbesserung der Lernobjekte und Kurse in Hinblick auf ihre Qualität (fachliche Korrektheit, sinnvoller Aufbau, Güte der Parametrisierung, etc.) und ihre Wiederverwendung im Lehrmaterialerstellungsprozess.

Tabelle 5-7 zeigt im Überblick, welche Rollen Feedback-Annotationen verfassen und welche die durch die Feedback-Annotationen initiierten Änderungen einpflegen.

Rolle / Aufgabe	Design	Entwicklung	Nutzung
Atom- und Moduldesigner	Pflegt Verbesserungen ein	Verfasst Annotationen, pflegt Verbesserungen ein	
Kursdesigner	Verfasst Annotationen, pflegt Verbesserungen ein	Verfasst Annotationen, pflegt Verbesserungen ein	Verfasst Annotationen, pflegt Verbesserungen ein
Kurs-Presenter	Verfasst Annotationen	Verfasst Annotationen, bearbeitet Annotationen von Kurstutoren und Lernern	Verfasst Annotationen
Kurstutor	Verfasst Annotationen	Verfasst Annotationen	Verfasst Annotationen
Lerner	Verfasst Annotationen	Verfasst Annotationen	Verfasst Annotationen

Tabelle 5-7: Feedback-Annotationen und Rollen im Prozess

Feedback zur Qualität der Nutzung in der Lehre

Schwerpunkt dieser Art von Feedback ist es, die Nutzung der Lehrmaterialien, also ihren Einsatz für Lehrveranstaltungen, zu verbessern. Dies ist auch deswegen nötig, da bei der Erstellung der Lehrmaterialien zwar bestimmte Arten der Nutzung berücksichtigt wurden, jedoch unmöglich alle Arten der Nutzung, wie sie möglicherweise gewünscht werden, bedacht werden können. Dies kann Korrekturen und Änderungen verschiedener Art nach sich ziehen:

- Korrektur inhaltlicher Fehler.
- Verbesserung der Strukturierung der Lehrmaterialien, falls der bisherige Aufbau ungünstig oder nicht zur Lehrveranstaltung passend war.
- Korrektur der Parameter oder Constraints des Kurses mit dem Ziel, die Nebenbedingungen, unter denen der Kurs einsetzbar ist, präziser zu beschreiben. Die Abstimmung zwischen dem Kurs und den vorgesehenen Nutzungsarten wird so verbessert.
- Korrektur der Parameter oder Constraints einzelner Module mit der Folge, dass möglicherweise diese Module durch andere, passendere im Kurs ersetzt werden.

Feedback-Annotationen dieser Art werden gemeinhin von den Nutzern von Lehrmaterialien verfasst, also von Kurs-Presentern, Kurstutoren und Lernern. Die Feedback-Annotationen

von Kurs-Presentern werden unmittelbar an die Designer weitergereicht. Die Feedback-Annotationen von Lernern werden zuerst vom Kurs-Presenter bearbeitet, der verantwortlich für die Lehrveranstaltung ist. Er weiß, welche Inhalte er vermitteln will, und er hat eine klare Vorstellung davon, wie der Kurs gestaltet sein soll. Daher muss er Feedback-Annotationen der Kurstutoren und Lerner zuerst daraufhin untersuchen, ob er seine Lehrveranstaltung anpassen muss und ob Korrekturen am Kurs wirklich nötig sind. Erst wenn der Kurs-Presenter befindet, dass Änderungen am Kurs durchgeführt werden müssen, leitet er die Feedback-Annotation an die Designer weiter, die dann ihrerseits die Feedback-Annotation diskutieren und bearbeiten.

Feedback zur Verbesserung der Abstimmung zwischen den Prozesspartnern

Feedback-Annotationen von Endnutzern betreffen im wesentlichen die inhaltliche Korrektheit und den Aufbau der Lehrmaterialien (die Struktur sowie die im Kurs verwendeten Module) im Hinblick auf ihre Nutzbarkeit zur Wissensvermittlung. Im Gegensatz dazu liegt der Schwerpunkt des Feedback zur Verbesserung der Abstimmung zwischen den Prozesspartnern darauf, einen Abgleich der Interessen der Prozesspartner herbeizuführen und so den Inhalt und die Struktur von Lehrmaterialien insgesamt zu verbessern. Jeder Prozessteilnehmer – egal ob Designer oder auch Kurs-Presenter – kann solche Annotationen verfassen.

Gegenstand von von Designern verfassten Feedback-Annotationen ist die inhaltliche Korrektheit, die Struktur der Lehrmaterialien und deren Parametrisierung sowie die Konsistenz der Lehrmaterialien untereinander. Jede Rolle, die an der Erstellung von Lehrmaterialien beteiligt ist, hat spezielle Fachkenntnisse und Erfahrungen, die sie in ihre Lernobjekte einbringen, aber auch bestimmte Interessen.

> Beispielsweise werden Atom- und Moduldesigner das Interesse haben, fachlich hervorragende Module und Atome zu bieten, die möglichst gut wiederverwendbar sind: Im universitären Umfeld bedeutet eine weite Verbreitung der Lernobjekte einen hohen Bekanntheitsgrad und bei entsprechend hoher Qualität eine gute Reputation der Atom- bzw. Moduldesigner. Im kommerziellen Umfeld führt eine weite Verbreitung der Module und Atome zu mehr Einnahmen aufgrund von Nutzungs- oder Lizenzgebühren.
>
> Kursdesigner haben andere Interessen: Ihnen liegt daran, einen Kurs inhaltlich und in seiner Struktur ideal auf bestimmte Lehrveranstaltungen abzustimmen; eine hohe Wiederverwendbarkeit der einzelnen Lernobjekte (nicht des Kurses!) ist für sie nicht so wichtig.

Jede Feedback-Annotation von Designern ist so von Interessen geleitet. Lesen nun andere Designer die Feedback-Annotation, so wird eine Diskussion zwischen dem Verfasser der Feedback-Annotation und deren Lesern in Gang gesetzt, an deren Ende ein Ausgleich zwischen den einzelnen, möglicherweise entgegengesetzten Interessen steht. Indem sowohl die Interessen und Anliegen der Verfasser der Feedback-Annotationen als auch der Autoren der Lernobjekte berücksichtigt werden, wird letztlich die Qualität der betreffenden Lernobjekte erhöht.

In 5.3.2.3 wurde bereits der iterative Charakter des Lehrmaterialerstellungsprozesses angesprochen. Feedback-Annotationen sind der Auslöser einer Iteration.

5.4.2.4 Soziale Annotationen zur Verbesserung der Koordination

Soziale Annotationen dienen der besseren Abstimmung der Lehrmaterialdesigner (Atomdesigner, Moduldesigner, Kursdesigner) und der Kurs-Presenter untereinander. Soziale Annotationen stehen damit im Gegensatz zu Feedback-Annotationen nicht in inhaltlichem Zusammenhang zu den Lehrmaterialien, sondern dienen der Verbesserung des Prozessablaufs. Folgende Beispiele illustrieren, wie mittels Annotationen eine Verbesserung der Koordination

der Prozessteilnehmer insbesondere an Schnittstellen von Teilprozessen erreicht werden kann:

- Termine für die Verfügbarkeit von Lernobjekten mitteilen:

  An alle betreffenden Lernobjekte eines Moduls wird eine Annotation mit dem betreffenden Termin angehängt. Diese Annotation wird von den Lernobjektverantwortlichen gelesen; sie sind über den Termin informiert und können ihre Arbeiten an den Lernobjekten entsprechend ausplanen.

- Korrekturen an einem Lernobjekt anfordern:

  An das betreffende Lernobjekt wird eine Annotation angehängt, in der die durchzuführenden Änderungen detailliert beschrieben sind (Feedback-Annotation). Es kann sein, dass der Autor des Lernobjekts noch weitergehende Fragen hat oder verschiedene Änderungen diskutiert werden müssen. In diesem Fall wird die Annotation entsprechend bearbeitet oder selbst wieder annotiert. Die Annotation dient damit nun der Abstimmung zwischen dem Autor der Annotation und dem Autor des Lernobjekts und erhält so den Charakter einer sozialen Annotation.

Damit ein Prozessteilnehmer davon erfährt, dass ein Lernobjekt oder ein Kurs mit einer Annotation versehen ist, in der Termine vereinbart oder Korrekturen vorgeschlagen werden, müssen die Prozessteilnehmer benachrichtigt werden. Möglichkeiten, wie dies geschehen kann, wurden bereits in 5.3.2.2 beschrieben.

5.4.2.5 Zusammenfassung: Prozess mit Feedback- und sozialen Annotationen

Das folgende Prozessbild baut auf Abbildung 4.2 auf, in der der Prozess ohne Annotationen dargestellt wird. In Abbildung 5-6 sind nun die Feedback-Annotationen und sozialen Annotationen hinzugenommen. (Ein UML-Diagramm ist in Anhang C zu finden.)

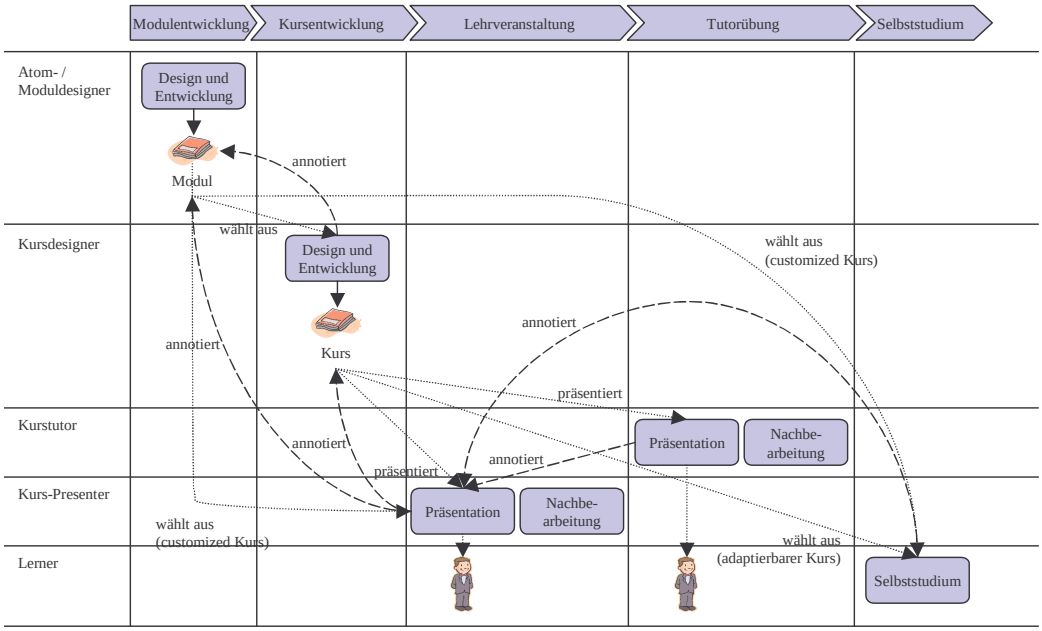


Abbildung 5-6: Prozess mit Feedback- und sozialen Annotationen (vereinfacht)

5.4.2.6 Prozessunterstützende Infrastruktur: Notwendigkeit und Anforderungen

In 5.4.1 und 5.4.2 wurde beschrieben, wie das Datenmodell und der Lehrmaterialerstellungsprozess angepasst werden müssen, damit die Qualität einerseits der Lehrmaterialien und andererseits des Prozessablaufs verbessert werden kann. Die Änderungen am Datenmodell sind minimal, da sich Annotationen nahtlos in das bestehende Datenmodell unter Verwendung der vorhandenen Elemente integrieren lassen. Lediglich ein neuer Assoziationstyp „annotiert“ sowie seine Verfeinerungen (vgl. 5.4.1.3) sind in das Datenmodell aufzunehmen. Für die Integration von Feedback-Annotationen und sozialen Annotationen in den Prozess hingegen ist mehr Aufwand zu treiben: Es muss eine Infrastruktur bereitgestellt werden, die über den ganzen Prozess hinweg die Verwendung von Annotationen ermöglicht.

Ausgangspunkt ist die Notwendigkeit der Kommunikation zwischen den Prozessteilnehmern, wobei die Prozessteilnehmer sich nicht persönlich kennen müssen bzw. es zuerst einmal nicht notwendigerweise eine Möglichkeit der direkten Kontaktaufnahme gibt. Trotzdem besteht der Bedarf, den Prozesspartnern Korrekturvorschläge zu Inhalt, Struktur und Parametrisierung von Lernobjekten oder Kursen zu unterbreiten und sich an Schnittstellen zwischen Teilprozessen über das Vorgehen abzustimmen. Damit eine effiziente Koordination zwischen den Lehrmaterialerstellern möglich wird, gibt es auch zeitliche Restriktionen hinsichtlich der Wahrnehmung und Bearbeitung von Annotationen. Nur wenn Annotationen nach ihrer Erstellung schnell gelesen und bearbeitet werden, kann eine zeitnahe Abstimmung mit den betreffenden Prozessteilnehmern stattfinden. Hierzu muss der Prozess mit einer geeigneten Infrastruktur unterstützt werden.

Eine Infrastruktur, die mittels Annotationen dies leisten kann, muss folgenden Anforderungen genügen:

- Es muss die Möglichkeit geben, an jedes Lernobjekt (Atome und Module) und jeden Kurs eine Annotation anzufügen. So kann eine Kommunikation, beispielsweise Feedback, initiiert werden.
- Ebenso muss es die Möglichkeit geben, unter Berücksichtigung der Rollen im Lehrmaterialerstellungsprozess Annotationen selbst wieder zu annotieren. Andernfalls wäre die Kommunikation mittels Annotationen nicht bidirektional, sondern nur unidirektional vom Verfasser der Annotation zu deren Lesern.
- Es muss einen Mechanismus der Benachrichtigung geben, wenn an ein Lernobjekt bzw. einen Kurs eine Annotation angefügt wird oder eine bereits existierende Annotation selbst wieder annotiert wird. Dieser Mechanismus muss alle Interessenten informieren.
- Die Verwendung von Freitext in der Annotation muss möglich sein, denn nur so lassen sich die Einschränkungen von Metadaten bezüglich ihrer Aussagekraft kompensieren. Ebenso muss es möglich sein, dass (komplexe) Module als Annotationen dienen.

Im nächsten Kapitel wird eine Infrastruktur vorgestellt, die diese Anforderungen erfüllt.

5.5 Zusammenfassung

In diesem Kapitel wurde untersucht, wie Annotationen die verschiedenen Probleme, die in 3.7 und 4.6 angesprochen wurden, lösen können. Zuerst wurde in 5.2 ein Überblick über Annotationen gegeben: Der Begriff sowie wesentliche Eigenschaften von Annotationen wurden vorgestellt. Des weiteren wurden mehrere Möglichkeiten vorgestellt, Annotationen zu kategorisieren, beispielsweise nach dem Grad ihrer Strukturierung (freiformulierte Annotationen,

semistrukturierte Annotationen und strukturierte Annotationen) oder ihrer Zugriffsrechte (private Annotationen, Gruppenannotationen und öffentliche Annotationen). Auch wurde eine Beziehung zu Systemen mit ähnlichen Konzepten, beispielsweise Diskussionssystemen, etc., hergestellt.

Anschließend wurde in 5.3 detailliert gezeigt, wie mit Hilfe von Annotationen Probleme des Datenmodells und des Prozesses gelöst werden können. Es wurde dargelegt, wie Subjektivität in Metadaten explizit gemacht werden kann, wie komplexe Constraints mittels Annotationen für die Designer und evtl. auch die Nutzer besser verständlich werden bzw. wie Abhängigkeiten zwischen Parametern beschrieben werden können. Ebenso wurde gezeigt, wie Annotationen genutzt werden können, um eine Kommunikation und, darauf aufbauend, eine Koordination über den gesamten Lehrmaterialerstellungsprozess hinweg zu initiieren und aufrechtzuerhalten, obwohl die Kommunikationspartner a priori nicht bekannt sein müssen.

In 5.4 schließlich wurde beschrieben, welche Anpassungen am Datenmodell und am Prozess nötig sind, um Annotationen nutzen zu können. Es hat sich gezeigt, dass eine Integration von Annotationen ins Datenmodell denkbar einfach ist: Annotationen werden auf Module abgebildet und die Beziehung einer Annotation zu einem Lernobjekt wird mittels einer Assoziation vom neu eingeführten Typ „annotiert“ und einer Verfeinerung dieses Typs realisiert. Die Integration von Annotationen in den Lehrmaterialerstellungsprozess hat sich etwas komplexer gestaltet, da mit Annotationen nun auch Informationsflüsse entgegen der Prozessrichtung möglich sind und Iterationen im Prozess berücksichtigt werden mussten.

Damit Annotationen in der Lehrmaterialerstellung sinnvoll und effizient eingesetzt werden können, ist eine durchgängige Unterstützung der Prozessteilnehmer zur Nutzung Annotationen nötig. Es muss eine Infrastruktur geschaffen werden, die während allen Prozessschritten die Erstellung und Bearbeitung von Annotationen erlaubt. Eine solche Infrastruktur wird im folgenden Kapitel vorgestellt.

6 Konzeption der Systemarchitektur

6.1 Überblick

In den letzten Kapiteln wurde die Bedeutung eines informellen Informationsaustauschs für eine effiziente Erstellung von möglichst gut wiederverwendbaren Lernobjekten und Kursen herausgearbeitet. Es hat sich gezeigt, dass in dem sozialen und organisatorischen Umfeld, in dem Lernobjekte und Kurse erstellt und genutzt werden, besonders Annotationen geeignet sind, diesen Informationsaustausch zu fördern: Annotationen können verwendet werden, um sowohl den Inhalt der Lernobjekte (Atome und Module) und Kurse als auch deren Erstellung bzw. Nutzung zu optimieren. Und sie können eingesetzt werden, um die Wiederverwendung der Lernobjekte und Kurse zu verbessern, obwohl die Autoren bzw. die Verantwortlichen für die fraglichen Lernobjekte und Kurse möglicherweise nicht direkt kontaktiert werden können.

Diese Effizienzsteigerungen hinsichtlich der Qualität und des Prozesses sind nur möglich, wenn Annotationen über den gesamten Prozess hinweg durchgängig eingesetzt werden können – von der Erstellung der Atome über die Gestaltung von Modulen und Kursen bis hin zur Präsentation durch die Kurs-Presenter und das Selbststudium der Lerner. Um Annotationen hier möglichst durchgängig einsetzen zu können, ist es nötig, den gesamten Prozess mit einer Infrastruktur zu hinterlegen, die in jedem Prozessschritt die Nutzung von Annotationen ermöglicht. Dies betrifft auch Aspekte der Speicherung, der Verwaltung und des Abrufs von Lernobjekten, denn Lernobjekte sind es, die in erster Linie annotiert werden, und es ist nötig, einen Weg aufzuzeigen, wie bereits bestehende Lernobjekte dynamisch mit Annotationen versehen werden können. Aber auch Aspekte der Awareness bezüglich neuer Annotationen müssen berücksichtigt werden, denn Annotationen können ihrer Aufgabe nur dann gerecht werden, wenn sie auch gelesen und genutzt werden. Dies setzt voraus, dass potenzielle Interessenten von der Existenz der Annotationen wissen.

In 6.2 wird zuerst ein generisches Modell von Lernplattformen vorgestellt. Es wird untersucht, wie eine Annotationsinfrastruktur in dieses Modell integriert werden kann.

Im nächsten Schritt (Abschnitt 6.3), dem Hauptteil dieses Kapitels, wird dann die Annotationsinfrastruktur vorgestellt. In diesem Abschnitt wird insbesondere auf die einzelnen Komponenten der Annotationsinfrastruktur und ihr Zusammenspiel eingegangen.

In 6.4 wird schließlich vorgestellt, wie aus Lernobjekten konkrete Kurse erstellt werden. Die theoretischen Grundlagen hierfür wurden bereits in 3.6 beschrieben. Hier soll nun betrachtet werden, wie diese Schritte in einem konkreten System umgesetzt werden. Es wird hierbei

auch im Detail erläutert, wie die Annotationen in die Kurse integriert werden und welche Rahmenbedingungen dabei berücksichtigt werden müssen.

6.2 Umfeld

6.2.1 Grundlagen: Lernplattformen

Lernplattformen, auch Lernmanagementsystem (LMS) genannt, bilden den Kern komplexer, Web-basierter E-Learning-Infrastrukturen. Eine vollständige Lehr-/Lernplattform (in folgenden kurz „Lernplattform“[77]), die alle Phasen zur Erstellung und Nutzung von Lehrmaterialien abdeckt, integriert Authoring- und Lernsysteme, welche über Schnittstellen miteinander verknüpft werden können. Eine Lernplattform konzentriert sich dabei auf inhaltliche Aspekte (content), deren Bewertung (assessment), die beschreibenden Metadaten sowie entsprechende Tools und Architekturen.

Eine Lernplattform erfüllt dabei zwei Aufgaben: Sie realisiert ein Datenmodell, das idealerweise dem ähnelt, das in Kapitel 3 besprochen wurde, und sie unterstützt den Prozess der Lehrmaterialerstellung und –nutzung, vgl. Kapitel 4. Das Datenmodell, das konkreten, bereits realisierten Lernplattformen zugrunde liegt, variiert jeweils. Jedoch können aufgrund der Gestaltung des Datenmodells in Kapitel 3 konkrete Datenmodelle auf das in Kapitel 3 vorgestellte Datenmodell zurückgeführt werden. Für die Lehrmaterialerstellungs- und –nutzungsprozesse, wie sie von konkreten Lernplattformen unterstützt werden, gilt dies analog: Die jeweils implementierten bzw. unterstützten Prozesse variieren sowohl in Gestalt als auch in Umfang, aber ihre Grundzüge finden sich im generischen Prozess, der in Kapitel 4 vorgestellt wurde, wieder.

In Anlehnung an [Schu03] wird im folgenden eine idealtypische Architektur einer Lernplattform vorgestellt. Eine Lernplattform besteht demnach aus einer Systemschicht und einer Datenhaltungsschicht. Die Systemschicht und die Datenhaltungsschicht sind hierbei nicht voneinander isoliert; es gibt auch Querbezüge.

In der Datenhaltungsschicht werden alle Lehrmaterialien inkl. Lernobjekte, Metadaten, Constraints, etc. und alle Nutzerdaten (Prozessmodell, Rechte, Rollen, etc.) abgelegt. Dies wird beispielsweise über Dateisysteme oder Datenbanken realisiert.

Die Systemschicht umfasst ein Authoring-System sowie ein Lernsystem. Mit dem Authoring-System werden die Lehrmaterialien erstellt, mit dem Lernsystem werden sie den Lernern präsentiert.

Ein Authoring-System wird oft aus einer Autoren- und einer Management-Umgebung kombiniert, wobei in der Autorenumgebung Lernobjekte erstellt und bearbeitet und in der Management-Umgebung die Metadaten und Zugriffsmethoden auf Lernobjekte geregelt werden können.

Ein Lernsystem wird in Informationssystem, Selbstlern-, Übungs- und Gruppenlernumgebung untergliedert. Reine Informationssysteme sind lediglich für die Darstellung von Inhalten der Lernobjekte zuständig, wie beispielsweise Web-Browser. In der Selbstlernumgebung dagegen, welche häufig über eine Benutzerverwaltung verfügt, kann der Lerner einen Kurs entsprechend der eigenen Nutzung adaptieren und dann mit ihm lernen. In der Übungsumgebung kann der Lerner Übungsaufgaben zur Selbstkontrolle seines Wissenstands lösen. In der

[77] Es gibt in dieser Arbeit keine explizite Unterscheidung zwischen den Begriffen „Lernplattform“ und „Lehrplattform“. Es wird nur der Begriff „Lernplattform“ verwendet.

Gruppenlernumgebung liegt oft der Fokus auf den Kommunikationsmethoden und der Kooperation in der Gruppe [Xu00].

Beide Systeme – Authoring-System und Lernsystem – bestehen aus einer Benutzerschnittstelle und der Anwendungslogik. Die Anwendungslogik wird hierbei typischerweise von Diensten realisiert, die meist außerhalb des Systems als Module realisiert sind. Auf diese Dienste kann über definierte Schnittstellen zugegriffen werden. Aus den typischen Eigenschaften der Lehrmaterialien (vgl. 2.2.2) und den Operationen bzw. Prozesse darauf (vgl. 4.3) können folgende Dienste von Authoring- und Lernsystemen im Kontext dieser Arbeit identifiziert werden:

- Recherchedienst

  Der Recherchedienst erlaubt das Auffinden und den Austausch von Lernobjekten und Metadaten. Er ermöglicht ebenfalls die Definition von Metadaten sowie die Auswertung bereits bestehender Metadaten.

- Sichtengenerierungsdienst

  Der Sichtengenerierungsdienst passt die Lehrmaterialien (Atome, Module, Kurse) den jeweiligen Teilprozessen hinsichtlich Rollen, Aufgaben, Ausführungen und Werkzeugen an. Diese Anpassung wird mittels Sichten realisiert, wobei die Prozessinformationen bzw. die zugrundeliegenden Daten der Prozessteilnehmer verfügbar sind.

- Layoutdienst

  Der Layoutdienst erlaubt den Lernern die Definition und Auswahl verschiedener Darstellungsformen und/oder Darstellungen desselben Kurses.

Ferner gibt es noch zwei Dienste, die parallel zu Benutzerschnittstelle und Anwendungslogik bereitgestellt werden: Einen Security-Dienst für die Nutzer- und Rechteverwaltung und einen Kommunikationsdienst, der eine Kommunikationsinfrastruktur für die Designer und Endnutzer zur Verfügung stellt. Beide Dienste realisieren Funktionalität, die in allen Systemen genutzt wird. Abbildung 6-1 liefert einen Überblick über die Architektur einer idealtypischen Lernplattform.

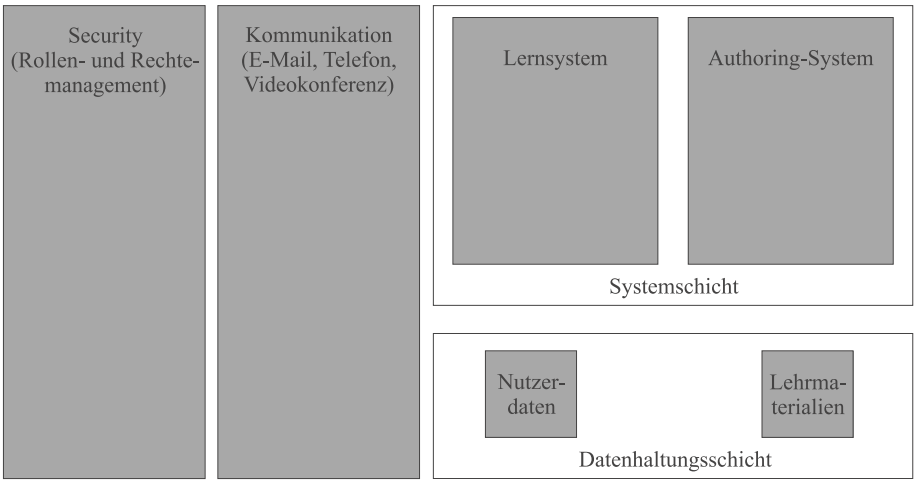


Abbildung 6-1: Überblick über Architektur einer Lernplattform

6.2.2 Einordnung der Annotationsunterstützung

Die Annotationsunterstützung besteht aus mehreren Elementen. Zum einen sind Komponenten nötig, mit denen Annotationen eingegeben und angezeigt werden können. Diese Komponenten sind mit dem Authoring-System und dem Lernsystem eng verbunden, denn in beiden Systemen werden Annotationen erstellt und genutzt. Eine entsprechende Infrastruktur muss also sowohl im Authoring-System, als auch im Lernsystem vorhanden sein. Zum anderen müssen Annotationen sowie die Verbindung von Annotationen zu den annotierten Lehrmaterialien (in Form von Assoziationen) gespeichert werden. Dies geschieht in einer Annotationsdatenbank, welche in der Datenhaltungsschicht angesiedelt ist. In Abbildung 6-2 ist zu sehen, wie die einzelnen Elemente einer Annotationsunterstützung in eine Lernplattform eingeordnet werden können.

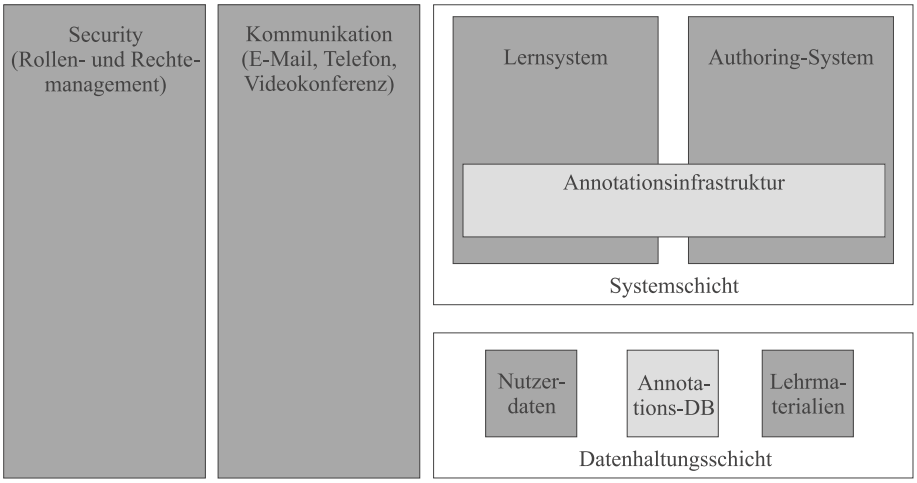


Abbildung 6-2: Architektur einer Lernplattform mit Annotationsunterstützung

Wichtig ist zu beachten, dass alle Elemente der Annotationsunterstützung analog zu 6.2.1 wieder den Security- und den Kommunikationsdienst nutzen.

6.2.3 Fachliche Anforderungen an ein System zur Annotationsunterstützung

Aus Sicht der Nutzer (Designer und Endnutzer) spielen vor allem zwei Aspekte eines Systems zur Annotationsunterstützung eine wesentliche Rolle: Das System muss eine einfache und möglichst intuitive Eingabe und Bearbeitung von Annotationen unterstützen, und das System – und damit auch die von ihm verwalteten Annotationen – muss für möglichst alle Lernplattformen (und damit auch über die Lehrmaterialformate beliebiger Systeme hinweg) verfügbar sein.

Die erste Anforderung ist essenziell, denn nur wenn die Eingabe und Bearbeitung von Annotationen einfach möglich ist, werden Annotationen genutzt. Dies erfordert eine möglichst enge Integration der Tools zur Erstellung und Bearbeitung von Annotationen in die Authoring- und Lernsysteme. Der Aspekt der einfachen Nutzung umfasst aber auch die Art und Weise, wie neue Annotationen den Nutzern vermittelt werden (Awareness). Die Art der Benachrichtigung muss einfach zu konfigurieren sein und mehrere, „verschieden aufdringliche“ Arten der Benachrichtigung erlauben.

Ebenso wichtig wie eine einfache Nutzung des Annotationsunterstützungssystems ist dessen Verfügbarkeit. Lehrmaterialien werden heute mit einer Vielzahl von Lernplattformen erstellt; eine Wiederverwendung von Lehrmaterialien ist nur dann effektiv möglich, wenn die Grenzen zwischen den Lernplattformen und den von ihnen genutzten Formaten für Lehrma-

terialien überwunden werden. Dies erfordert es einerseits, dass die Lehrmaterialien dem in Kapitel 3 vorgestellten Datenmodell entsprechen, so dass eine einheitliche Beschreibung aller Lehrmaterialien bereit steht, und andererseits, dass jede Lernplattform die Anreicherung der Lehrmaterialien mit Annotationen ermöglicht, um die in 3.7 und 4.6 aufgeführten Schwächen auszugleichen.

Bei beiden Aspekten – einfache Erstellung und Bearbeitung von Annotationen und Verfügbarkeit von Annotationen über Lernplattformgrenzen hinweg – müssen jeweils Sicherheitsaspekte, insbesondere der Zugriffsschutz bei privaten und Gruppenannotationen, berücksichtigt werden.

6.3 Ein System zur Unterstützung von Annotationen

6.3.1 Überblick: Funktionale Einheiten einer Annotationsinfrastruktur

Zur Herleitung der Komponenten einer Lernplattform sollen die folgenden Szenarien betrachtet werden:

Szenario 1 (Verfassen einer Annotation):

> Bei der Nachbereitung der Vorlesung „Computergestützte Gruppenarbeit“ anhand eines vorlesungsbegleitend bereitgestellten Skripts fällt dem Lerner *L* eine Unstimmigkeit auf Seite 8 des Vorlesungsskripts auf. An der Stelle des Fehlers erstellt er eine Annotation, in der er auf den Fehler hinweist und um eine Korrektur bittet.

In diesem Szenario sind mehrere Komponenten aktiv: Die Eingabe der Annotation findet in einem *Annotationseditor* statt. Diese Komponente könnte ein eigenständiges Programm sein, aber auch ein Plugin im Lehrmaterial-Viewer ist denkbar. Anschließend müssen die Annotation sowie die Assoziation zwischen den Lehrmaterialien und der Annotation persistent gespeichert werden. Dies geschieht in einem Repository, dem sogenannten *Annotations-Repository*. In diesem Annotations-Repository wird auch vermerkt, ob es sich bei der Annotation um eine private, eine öffentliche oder eine Gruppenannotation handelt.

Szenario 2 (Benachrichtigung über eine Annotation und Lesen der Annotation):

> Der Kurs-Presenter *P*, der die Vorlesung „Computergestützte Gruppenarbeit“ liest, wird per E-Mail informiert, dass Lerner *L* eine Annotation zu dem von ihm bereitgestellten Vorlesungsskript verfasst hat. *P* ruft das Vorlesungsskript ab, liest die Annotation und korrigiert den Fehler im Vorlesungsskript. Dies geschieht über eine Änderung eines Lernobjekts, das Teil des Vorlesungsskripts ist und das im Lernobjekt-Repository abgelegt ist.

Zu den bei Szenario 1 genannten Komponenten (Annotations-Repository und Annotationseditor) kommen nun noch eine Notifikationskomponente (der sogenannte Notifikations-Service), ein Annotations-Viewer und ein Lernobjekt-Repository hinzu. Der *Notifikations-Service* informiert den Autor und die Interessenten eines Lernobjekts über Annotationen, die bearbeitet oder neu an das Lernobjekt angefügt wurden. Diese Notifikationskomponente hat zur Aufgabe, die betroffenen Prozessteilnehmer – Designer wie Nutzer – auf eine bestimmte, konfigurierbare Weise zu informieren, wenn Annotationen erstellt werden. Der *Annotations-Viewer* stellt Annotationen zusammen mit den betroffenen Lernobjekten dar. Das *Lernobjekt-Repository* ist ein logischer Datenspeicher, der alle Speicherorte von Lernobjekten zusammenfasst. Das Lernobjekt-Repository umfasst alle Bestände von Lernobjekten und abstrahiert vom konkreten Speicherort sowie von der konkreten Speicherart der Lehrmaterialien.

Szenario 3 (Überwachung der Konsistenz von Annotationen):

> Bei regelmäßig durchgeführten Abgleichen, die das Annotationsunterstützungssystem auf dem Lernobjekt-Repository und dem Annotations-Repository durchführt, wird festgestellt, dass Kurs-Presenter *P* am Skript zur Vorlesung „Computergestützte Gruppenarbeit“ eine Änderung vorgenommen hat. Da zu diesem Skript eine Annotation vorliegt, wird der Verfasser der Annotation, Lerner *L*, über die Änderung informiert, da diese seine Anmerkung (in der Annotation) betreffen könnte. *L* kann nun prüfen, ob durch die Änderungen am Vorlesungsskript der von ihm entdeckte Fehler korrigiert wurde.

In diesem Szenario wird eine neue Komponente aktiv: Der *Lernobjekt-Scanner*. Aufgabe dieser Komponente ist es, die Integrität bzw. Konsistenz des Annotations-Repositories sicherzustellen, also einen Abgleich zwischen Lernobjekten und den ihnen zugeordneten Annotationen durchzuführen[78]. Hierzu führt der Lernobjekt-Scanner Tests durch und meldet potenzielle Unstimmigkeiten an die Notifikationskomponente.

Tabelle 6-1 fasst die für die Annotationsinfrastruktur benötigten Komponenten zusammen und beschreibt kurz ihre Aufgabe.

Komponente	**Beschreibung**
Annotations-Repository	Datenspeicher (Dateisystem, Datenbank, etc), in dem die Annotationen abgelegt werden. Dieser Datenspeicher enthält ebenso die Assoziation zwischen einer Annotation und dem annotierten Element (Lernobjekt oder Annotation), also Informationen, welche Annotation welchem Element zugeordnet ist.
Lernobjekt-Repository	Logischer Datenspeicher, in dem alle Lernobjekte abgelegt sind. Das Lernobjekt-Repository abstrahiert von der konkreten Speicherart und dem konkreten Speicherort von Lernobjekten.
Lernobjekt-Scanner	Komponente zur Sicherung der Konsistenz der Zuordnung „Annotation – Lernobjekt“.
Notifikations-Service	Informiert potenzielle Interessenten über Ereignisse im Zusammenhang mit Lernobjekten und Annotationen.
Annotations-Viewer	Stellt Annotationen dar. Eine möglichst enge Integration der Präsentation der Annotation mit der Präsenta-

[78] Integrität und Konsistenz des Annotations-Repositories bedeutet grob gesagt, dass zu jeder Annotation auch ein annotiertes Element (ein Lernobjekt oder selbst wieder eine Annotation) existiert und dass die Assoziation, die das annotierte Element mit der Annotation verbindet, mit allen nötigen Parameterwerten versehen ist. Eine genaue Beschreibung der Fälle, die der Lernobjekt-Scanner überprüft, ist in 6.3.5 zu finden.

	tion der Lehrmaterialien ist anzustreben.
Annotationseditor[79]	Ermöglicht die Eingabe von Annotationen. Ebenso wie beim Annotations-Viewer ist eine möglichst enge Integration des Annotationseditors in das Lernsystem (bei Annotationen der Lerner bzw. Kurs-Presenter und Kurstutor) bzw. in das Authoring-System, mit dem die Lehrmaterialien erstellt werden (bei Annotationen der Designer bzw. Kurs-Presenter) anzustreben.

Tabelle 6-1: Komponenten und ihre Aufgaben

Das Zusammenspiel der einzelnen Komponenten ist in Abbildung 6-3 dargestellt. Die in Abbildung 6-2 aufgeführten Dienste „Security“, „Datenhaltung“, „Kommunikation“ und „Lernsystem und Authoring-System“ sind den jeweiligen Komponenten hinterlegt um darzustellen, welche Dienste einer Lernplattform von den einzelnen Komponenten genutzt werden.

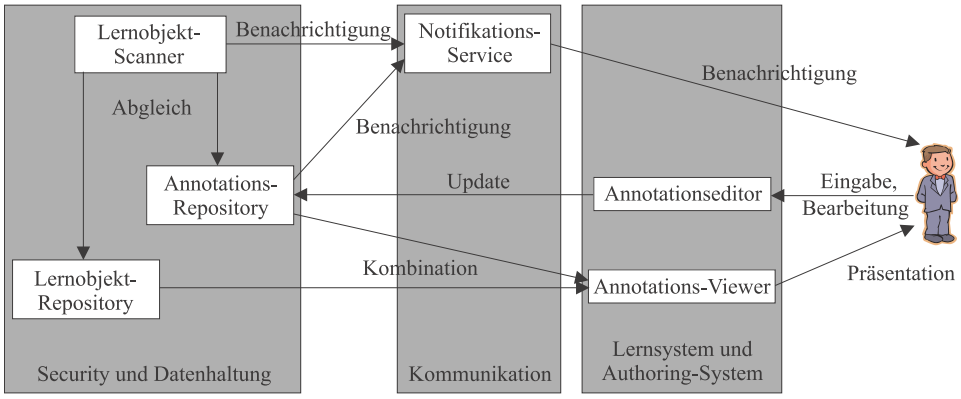


Abbildung 6-3: Zusammenspiel der Komponenten im Überblick

Wichtig ist zu beachten, dass in Abbildung 6-3 nur die Komponenten und Interaktionen zwischen Komponenten dargestellt sind, die für die Erstellung und Nutzung von *Annotationen* von Bedeutung sind. Komponenten und Interaktionen, die andere Nutzungen einer Lernplattform, beispielsweise die Verwendung bereits fertig erstellter Lehrmaterialien, die keine Annotationen enthalten, oder die Erstellung von Lernobjekten ohne die Nutzung von Annotationen, unterstützen, sind hier nicht aufgeführt[80].

Im folgenden werden die einzelnen Komponenten der Annotationsinfrastruktur sowie ihr Zusammenspiel im Detail erläutert. Ziel dieser Beschreibungen ist es, die Aufgabe jeder Komponente im Gesamtsystem deutlich zu machen. Implementierungsaspekte der Komponente werden hierbei nicht betrachtet; diese werden in Kapitel 7 geliefert.

[79] Annotations-Viewer und Annotationseditor können auch zu einem Tool kombiniert sein.

[80] Die Bearbeitung von Lernobjekten geschieht mit Authoring-Systemen, die direkt auf das Lernobjekt-Repository zugreifen. Ebenso geschieht die Erstellung von Lehrmaterialien aus Lernobjekten im wesentlichen in Zusammenarbeit mit dem Lernobjekt-Repository. (Nur wenn Annotationen in die Lehrmaterialien integriert werden sollen, kommt das Annotations-Repository ins Spiel.) Beide Vorgänge sind hier nicht eingezeichnet, da sie keine unmittelbare Anwendung der Annotationsinfrastruktur darstellen.

6.3.2 Einheitlicher Zugriff auf Lernobjekte: Das Lernobjekt-Repository

6.3.2.1 Aufgabe

Das Lernobjekt-Repository bietet eine einheitliche Zugriffsschicht auf alle verfügbaren Lernobjekte sowie die Strukturinformationen zwischen ihnen (Kompositionsbeziehungen und Assoziationen). Das Lernobjekt-Repository betrifft die Verbesserung der Wiederverwendbarkeit der Lernobjekte zwar nicht unmittelbar, aber der einheitliche Zugriff auf Lernobjekte, die entsprechend des in Kapitel 3 vorgestellten Datenmodells ausgestaltet sind, ist die Grundlage für den Einsatz von Annotationen und ermöglicht so erst Verbesserungen der Wiederverwendung der Lernobjekte und Optimierungen des Lehrmaterialerstellungs- und –nutzungsprozesses.

Die Lernobjekte werden im allgemeinen in unterschiedlichen Datenspeichern abgelegt; je nach Organisation werden unterschiedliche Datenspeicher verwendet. Ein Unternehmen verwendet beispielsweise das Dateisystem und nutzt Concurrent Versions System (CVS) [CVS] zur Versionierung der Lernobjekte, ein anderes Unternehmen legt die von ihm bereitgestellten Lernobjekte in einer Datenbank ab, eine Universität bietet auf Targeteam [TARGETEAM] aufbauende Lernobjekte an und nutzt Targeteam-Pools zur Speicherung der Lernobjekte, etc. Um einen reibungslosen Zugriff auf die Lernobjekte möglich zu machen, muss Zugriffstransparenz über die verschiedenen Systeme hinweg geschaffen werden; alle Lernobjekte müssen auf die gleiche Weise zugreifbar sein, so dass die Nutzung aller Lernobjekte – zumindest unter technischen Gesichtspunkten – sichergestellt ist. Dies wird durch eine Menge von Systemen realisiert, die in der Summe das Lernobjekt-Repository bilden: Das Lernobjekt-Repository bildet eine logische Datenhaltungsschicht, über die alle Lernobjekte auf einheitliche Weise genutzt werden können.

6.3.2.2 Aufbau und Funktionsweise

Das Lernobjekt-Repository besteht aus mehreren Teilsystemen, vgl. Abbildung 6-4: Ein Teilsystem, die *Lernobjektzugriffsschicht*, kapselt den Zugriff auf Lernobjekte, die in unterschiedlichen Datenspeichern (Dateisystem, Datenbank, Targeteam-Pool, etc.) abgelegt sind, und stellt eine für alle Datenspeicher einheitliche Zugriffsschicht zur Verfügung. Jede Organisation, die Lernobjekte anbietet, stellt diese über eine Lernobjektzugriffsschicht, die als zentraler Dienst in der Organisation betrieben wird, bereit[81]. Des weiteren sorgt die Lernobjektzugriffsschicht dafür, dass bereits existierende Lehrmittel wie Word-Dokumente, Powerpoint-Präsentationen, Text-, PDF- oder Postscript-Dateien, etc., die bereits in Lehrveranstaltungen genutzt werden, nach außen so dargestellt werden, dass sie dem in Kapitel 3 vorgestellten Datenmodell (bestehend aus Atomen und Modulen und beschrieben durch Parameter und Constraints) entsprechen und damit auch zusammen mit anderen Lernobjekten genutzt werden können. Die Lernobjektzugriffsschicht sorgt somit für Transparenz hinsichtlich dem Zugriff und dem Datenformat der Lernobjekte.

Ein weiteres Teilsystem, der *Lernobjekt-Service*, stellt einen Katalog bereit, in dem die Beschreibungen (Parameter) aller Lernobjekte geführt sind und in dem vermerkt ist, von welcher Organisation das jeweilige Lernobjekt angeboten wird. Hierzu hat der Lernobjekt-Service eine Verzeichnis- und eine Katalogkomponente. Der Lernobjekt-Service sorgt somit für die Transparenz hinsichtlich des Ortes, an dem Lernobjekte gespeichert sind.

Des weiteren werden alle Strukturinformationen zu den Lernobjekten im Lernobjekt-Service abgelegt, also Kompositionsbeziehungen zwischen Modulen bzw. Kursen und ihren Subelementen (Module und Atome) sowie Assoziationen zwischen Lernobjekten. Dies ermöglicht

[81] Zwischen der Lernobjektzugriffsschicht und den Datenspeichern besteht eine Client/Server-Beziehung: Die Lernobjektzugriffsschicht (Client) bezieht ihre Daten als von den Datenspeichern (Server).

es, die Struktur von Lernobjekten über den Lernobjekt-Service zu ermitteln, ohne auf die möglicherweise sehr umfangreichen Lernobjektinhalte zugreifen zu müssen. Die Beschreibungen (Parameter) der Kompositionen und Assoziationen werden ebenfalls vom Lernobjekt-Service verwaltet.

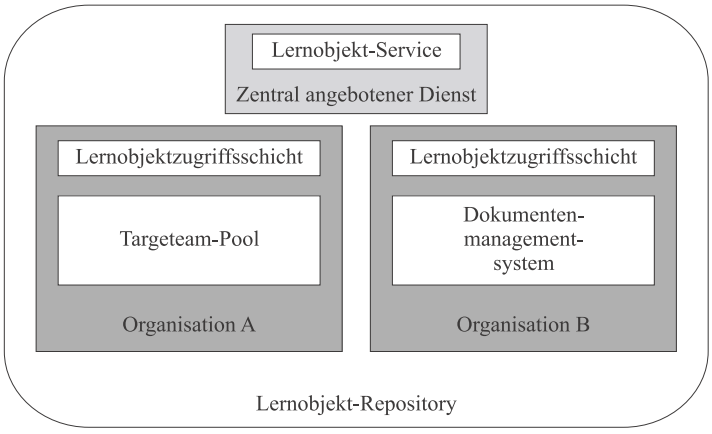


Abbildung 6-4: Aufbau des Lernobjekt-Repositories

6.3.2.3 Kapselung organisationsabhängiger Lernobjekt-Datenspeicher: Die Lernobjektzugriffsschicht

Aufgabe der Lernobjektzugriffsschicht ist es, einen einheitlichen Zugriff auf Lernobjekte unabhängig von der Art ihrer Speicherung und des evtl. eingesetzten Zugriffskontrollsystems zu ermöglichen. Hierzu wird auf den Rechnern eines jeden Lernobjektanbieters mindestens eine Instanz der Lernobjektzugriffsschicht installiert[82]. Hierbei müssen zwei Problemkreise berücksichtigt werden: Der Zugriffsschutz von Lernobjekten (dies ist insbesondere bei kostenpflichtigen Lernobjekten von Bedeutung) und der Zugriff auf unterschiedliche Datenspeicher, die die Lernobjekte beinhalten.

Zugriffsschutz

Der Zugriff auf Lernobjekte unterscheidet sich je nachdem, ob sie kostenfrei zur Verfügung gestellt werden oder ob für ihre Nutzung bezahlt werden muss. Um möglichst flexibel zu bleiben, müssen beide Formen der Bereitstellung von Lernobjekten – kostenpflichtig und kostenfrei – betrachtet werden:

- Kostenpflichtige Lernobjekte können erst nach Zahlung einer bestimmten Nutzungsgebühr genutzt werden. Die Nutzungsgebühr kann eine pauschale Zahlung sein, aber auch verschiedene andere Lizenzmodelle sind möglich.

  Unabhängig von der Art der Zahlung müssen hier zwei Besonderheiten berücksichtigt werden:

  - Die Lernobjekte selbst (ihre Inhalte) sind nicht allgemein zugänglich, sondern erst nach erfolgter Zahlung.

[82] Eine Instanz muss ein Lernobjektanbieter auf jeden Fall installieren. Diese Instanz kapselt die verschiedenen Lernobjekt-Datenspeicher des Lernobjektanbieters. Es ist jedoch auch möglich, dass ein Lernobjektanbieter mehrere Instanzen der Lernobjektzugriffsschicht installiert. Dies kann beispielsweise aus Gründen der Lastverteilung geschehen.

  - Um die Lernobjekte trotzdem nutzen zu können, müssen ihre Beschreibungen (die beschreibenden Parameter) allgemein zugreifbar sein[83].

- Kostenfreie Lernobjekte sind immer frei zugänglich. Dies betrifft sowohl ihre Inhalte als auch ihre Beschreibung. Es ist allerdings möglich, dass der Zugang erst nach erfolgter (kostenfreier) Registrierung gestattet wird.

Die Lernobjektzugriffsschicht abstrahiert von der konkreten Umsetzung des Zugriffsschutzes. Sie kapselt das konkrete Zugriffskontrollsystem, mit dem der Datenspeicher, in dem die Lernobjekte abgelegt sind, gegebenenfalls geschützt ist und erlaubt einen einheitlichen Zugriff auf die Lernobjekte.

Abstraktion der konkreten Speicherung der Lernobjekte in Datenspeichern

Ein weiterer Punkt betrifft die unterschiedlichen Möglichkeiten, Lernobjekte zu speichern. Die Datenspeicher, die in den unterschiedlichen Organisationen und Unternehmen zur Speicherung der Lernobjekte eingesetzt werden, bieten keine einheitliche Schnittstelle nach außen; die verwendete Software zur Speicherung der Lernobjekte (Dokumentenmanagementsysteme, Dokumentenspeicherung mittels Versionskontrollsystemen, Targeteam-Pools, einfache Ablage im Dateisystem, etc.) unterscheiden sich im Zugriff und in der Art und Weise, wie Lernobjekte abgelegt werden, grundsätzlich.

Um auf die verschiedenen Datenspeicher zugreifen zu können, sind somit verschiedene Implementierungen des Zugriffs nötig – je nach Ausprägung des Datenspeichers. Die Lernobjektzugriffsschicht abstrahiert hier von den konkreten Datenspeichern (hierbei ist es auch möglich, mehrere, unterschiedliche Datenspeicher durch eine Lernobjektzugriffsschicht zu verwalten) und bietet eine einheitliche Schnittstelle zum Zugriff auf die Lernobjekte an.

Dienste der Lernobjektzugriffsschicht

Die Lernobjektzugriffsschicht bietet nach außen Dienste gemäß Tabelle 6-2 an.

Parameter und Ergebnis	**Beschreibung**
Object get (*ObjID*, *UserID*)	
ObjID: Identifikator des gesuchten Lernobjekts *UserID*: Identifikator des Nutzers. Der Nutzer muss dem Datenspeicher, von dem die Lernobjektzugriffsschicht bedient wird, bekannt sein. *Object*: Das gesuchte Lernobjekt	Liefert das Lernobjekt mit dem Identifikator *ObjID*. Voraussetzung ist, dass der Nutzer *UserID* die Berechtigung hat, das Lernobjekt anzufordern. Gegebenenfalls muss vor der Anforderung mit get() eine Authentifizierung mittels authenticate() durchgeführt werden.

[83] Es ist ebenso denkbar, die Beschreibungen der Lernobjekte nur zusammen mit den Lernobjekten nach einer entsprechenden Zahlung bereitzustellen. So würde jedoch der Käufer eines Lernobjekts die Katze im Sack kaufen, da er nicht im Voraus prüfen kann, ob sich die betreffenden Lernobjekte in seinen Kurs bzw. sein Modul integrieren lassen. Es ist zweifelhaft, dass ein derartiges Nutzungsmodell von den Kunden (Modul-, Kursdesigner, Kurs-Presenter und Lerner) akzeptiert würde.

void put (*UserID, Object, Params*)	
UserID: Identifikator des Nutzers. Der Nutzer muss dem Datenspeicher, von dem die Lernobjektzugriffsschicht bedient wird, bekannt sein. *Object*: Das im Datenspeicher abzulegende Lernobjekt *Params:* Menge von Parametern, die das Lernobjekt *Object* beschreibt.	Legt das Lernobjekt *Object*, beschrieben durch die Parameter *Params*, im Datenspeicher ab. *Params* enthält auch den Identifikator des Lernobjekts. Voraussetzung ist, dass der Nutzer *UserID* die Berechtigung hat, das Lernobjekt abzulegen. Gegebenenfalls muss vor der Ablage mit put() eine Authentifizierung mittels authenticate() durchgeführt werden.
hasPermission authenticate(*UserID, Credential*)	
UserID: Identifikator des Nutzers. Der Nutzer muss dem Datenspeicher, von dem die Lernobjektzugriffsschicht bedient wird, bekannt sein. *Credential:* Ein vom jeweiligen Datenspeicher abhängiges Objekt zur Authentifizierung des Nutzers *UserID*, beispielsweise ein Passwort. *hasPermission:* Boolesches Flag, das zeigt, ob die Authentifizierung erfolgreich war	Führt die Authentifizierung der Nutzers *UserID* beim Datenspeicher durch. Die Authentifizierung erfolgt auf Grundlage des *Credential*-Objekts.

Tabelle 6-2: Dienste der Lernobjektzugriffsschicht

Lernobjekte können mittels des Dienstes get() abgerufen werden. Hierbei werden die Inhalte eines Lernobjekts als Ergebnis geliefert, nicht jedoch die Parameter, die das Lernobjekt beschreiben. Diese können über den Lernobjekt-Service mittels des Dienstes getDescr() ermittelt werden, vgl. Tabelle 6-3.

Mittels des Dienstes put() werden Lernobjekte an die Lernobjektzugriffsschicht zur Ablage im jeweiligen Datenspeicher übergeben. Hierbei wird auch eine Beschreibung des Lernobjekts in Form von Parametern (Metadaten) mit übergeben. Die Lernobjektzugriffsschicht speichert diese Parameter lokal auf dem Rechner, auf dem schon die Lernobjektzugriffsschicht läuft, um den Autoren und Verantwortlichen für dieses Lernobjekt einen schnellen Zugriff auf die Parameter zu gestatten. Zugleich leitet die Lernobjektzugriffsschicht die Parameter an den Lernobjekt-Service weiter, so dass die Beschreibung des Lernobjekts öffentlich zugänglich wird. Die Beschreibung des Lernobjekts ist damit an zwei Stellen gespeichert: Lokal bei der Lernobjektzugriffsschicht, die den Datenspeicher verwaltet, auf dem das Lernobjekt abgelegt ist, und im Lernobjekt-Service.

Die Konsistenz der Beschreibung wird durch die Lernobjektzugriffsschicht sichergestellt: Jede Änderung in den Parametern wird über die Lernobjektzugriffsschicht mittels des Dienstes put() durchgeführt; die geänderten Parameter werden dann an den Lernobjekt-Service propagiert. Eine Änderung der Parameter direkt über den Lernobjekt-Service ist nicht möglich.

6.3.2.4 Vermittlung von Lernobjekt-Datenspeichern: Der Lernobjekt-Service

Die dezentrale Natur der Speicherung der Lernobjekte in über Lernobjektzugriffsschichten gekapselten Datenspeichern erfordert einen Vermittlungsdienst, über den abgerufen werden kann, wo (auf welchem Server) bestimmte Lernobjekte zu finden sind. Dieser Vermittlungs-

dienst wird vom Lernobjekt-Service realisiert. Daneben realisiert der Lernobjekt-Service eine Suchfunktionalität, die die Identifikatoren aller Lernobjekte liefert, die bestimmten, als Suchwerten mitgegebenen Parametern entsprechen. Auch die Strukturinformationen von Modulen und Kursen – die Kompositions- und Assoziationsbeziehungen – werden vom Lernobjekt-Service verwaltet.

Aufbau des Lernobjekt-Service

Der Lernobjekt-Service besteht grundsätzlich aus zwei Teilen: Eine *Verzeichniskomponente*, in der die Zuordnung „Lernobjektidentifikator – Lernobjektzugriffsschichtadresse“ realisiert wird, und eine *Katalogkomponente*, die die Strukturinformationen zwischen Lernobjekten sowie die Beschreibungen von Lernobjekten, Kompositionen und Assoziationen verwaltet und eine Suchfunktionalität über diesen Beschreibungen anbietet. Diese Komponenten können auf verschiedene Weise realisiert sein. Im folgenden werden verschiedene Alternativen vorgestellt und bewertet.

Alternative 1: Realisierung der Verzeichniskomponente als zentraler, Realisierung der Katalogkomponente als verteilter Dienst
Die Verzeichniskomponente wird als zentraler Dienst realisiert, die Katalogkomponente wird verteilt und den jeweiligen Lernobjektzugriffsschichten zugewiesen. So wird jeder Lernobjektzugriffsschicht eine Instanz der Katalogkomponente anbei gestellt. Jede Katalogkomponente verwaltet nur die Beschreibungen der Lernobjekte, die über die ihr zugewiesenen Lernobjektzugriffsschicht abgerufen werden können. Die Suchfunktionalität nach Lernobjekten mit bestimmten Parametern, die ein Client benötigt, wird im anfragenden Client (Authoring-System, Lernsystem, etc.) realisiert.

In diesem Fall erledigt der Client, der die Lernobjekte identifizieren soll, einige Arbeiten selbst: Er ermittelt zuerst über die Verzeichniskomponente des Lernobjekt-Service die Adressen aller Lernobjektzugriffsschichten, über die auf Lernobjekte zugegriffen werden kann. Von diesen Adressen bezieht der Client über die Suchfunktionalität der Katalogkomponente des Lernobjekt-Service die Beschreibungen aller Lernobjekte, die seinen Anforderungen genügen. Ist dies erfolgt, hat der Client eine Liste von Identifikatoren der Lernobjekte, die die gewünschten Eigenschaften haben.

Da jede Katalogkomponente nur eine relativ überschaubare Anzahl von Beschreibungen von Lernobjekten verwaltet, können Suchanfragen schnell abgewickelt werden. Jedoch muss jeder Client die Logik zur Abfrage und Zusammenführung der Lernobjektbeschreibungen aufs Neue implementieren. Sofern nicht entsprechende Funktionalität in einer Bibliothek bereitgestellt wird, ist dies für die Client-Entwicklung zeitraubend und fehlerträchtig. Ein weiterer Nachteil ist, dass für das Abrufen der Lernobjektbeschreibungen relativ viele Netzverbindungen zu den einzelnen Katalogverbindungen nötig sind, was hohe Netzlast erzeugt. Dieser Nachteil kann jedoch durch den geschickten Einsatz von Caches abgeschwächt werden.

Alternative 2: Realisierung der Verzeichnis- und Katalogkomponente als zentraler Dienst
Beide Komponenten – die Verzeichniskomponente und die Katalogkomponente – werden als zentraler Dienst auf einem wohlbekannten Rechner realisiert. Die Verzeichniskomponente realisiert analog zu Alternative 1 die Zuordnung „Lernobjektidentifikator – Lernobjektzugriffsschichtadresse“ für alle Lernobjekte, die Katalogkomponente verwaltet die Beschreibungen aller derzeit verfügbaren Lernobjekte.

Die gesamte Logik der Suche passender Lernobjekte liegt im Lernobjekt-Service (genauer: in der Katalogkomponente) und kann direkt von Clients genutzt werden. Implementierungsaufwand für Auswertungen fällt für die Clients nicht an. Eine große Anzahl von Anfragen führt jedoch zu einer hohen Belastung der Server-Rechner, auf denen der Lehrmaterial-Ser-

vice läuft. Dieser Belastung kann jedoch mit geeigneten Optimierungsmaßnahmen wirkungsvoll begegnet werden.

Ergebnis
Im folgenden soll Alternative 2 weiter verfolgt werden: Die Katalogkomponente wird zusammen mit der Verzeichniskomponente als zentraler Dienst angeboten. So können die Clients hinsichtlich der Suche passender Lernobjekte schlank gehalten werden und es können die Algorithmen zur Auswahl der Lehrmaterialien gezielt an einer zentralen Stelle optimiert werden.

Dienste des Lernobjekt-Service

Der Lernobjekt-Service bietet Dienste gemäß Tabelle 6-3 an.

Parameter und Ergebnis	**Beschreibung**
*URL*s getServers(*ObjID*)	
ObjID: Identifikator des gesuchten Lernobjekts *URL*s: Menge der Server-URLs, auf denen das Lernobjekt *ObjID* abgelegt ist. Die Server-URLs verweisen auf die jeweiligen Lernobjektzugriffsschichten.	Liefert die Menge der URLs der Server, auf denen Lernobjekt *ObjID* gespeichert ist. Da Lernobjekt *ObjID* (theoretisch) auf mehreren Rechnern gespeichert sein kann, wird eine Menge als Ergebnis zurückgeliefert. Dieser Dienst wird von der Verzeichniskomponente des Lernobjekt-Service erbracht.
ObjIDs getObjIDs (*Params*)	
Params: Parameter, die die gesuchten Lernobjekte erfüllen müssen *ObjIDs:* Menge der Identifikatoren passender Lernobjekte	Liefert die Menge der Identifikatoren von Lernobjekten, die zu den übergebenen Parametern *Params* passen. Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.
Params getDescr(*ObjID*)	
ObjID: Identifikator des Lernobjekts, dessen Beschreibung angefordert wird *Params:* Menge von Parametern, die Lernobjekt *ObjID* beschreiben	Liefert die Beschreibung von Lernobjekt *ObjID* als Menge von Parametern. Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.
void register(*ObjID*, *Params*, *URL*)	
ObjID: Identifikator des zu registrierenden Lernobjekts *Params*: Parameter, die das Lernobjekt beschreiben *URL*: URL der Lernobjektzugriffsschicht, über die Lernobjekt *ObjID* abgerufen werden kann	Registriert Lernobjekt *ObjID* beim Lernobjekt-Service. Dieser Dienst wird von der Verzeichniskomponente des Lernobjekt-Service erbracht.

void addSubconnection (*ConID, ObjID, SubID*)	
ConID: Identifikator der neu anzulegenden Kompositionsbeziehung[84]. *ObjID*: Identifikator des Lernobjekts, dem ein Subelement hinzugefügt werden soll. *SubID*: Identifikator des hinzuzufügenden Subelements.	Definiert eine Kompositionsbeziehung mit Identifikator *ConID* zwischen Lernobjekt *ObjID* und einem Subelement *SubID*. Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.
void removeSubconnection(*ConID*)	
ConID: Identifikator der Kompositionsbeziehung, die aufgelöst werden soll.	Entfernt die Kompositionsbeziehung *ConID* zwischen zwei Lernobjekten. Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.
ConIDs getSubconnections(*ObjID*)	
ObjID: Identifikator des Lernobjekts, dessen Subelemente als Ergebnis geliefert werden sollen. *ConIDs*: Menge von Identifikatoren aller Kompositionen, die von *ObjID* ausgehen.	Liefert eine Menge *ConIDs* von Identifikatoren der Kompositionen, die von Modul bzw. Kurs *ObjID* ausgehen. Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.
void addAssociation (*ConID, ObjID, TargetID*)	
ConID: Identifikator der neu anzulegenden Assoziation. *ObjID*: Identifikator des Lernobjekts, von dem die Assoziation ausgeht. *TargetID*: Identifikator des assoziierten Lernobjekts.	Definiert eine Assoziation mit Identifikator *ConID* zwischen Lernobjekt *ObjID* und Lernobjekt *TargetID*. Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.
void removeAssociation(*ConID*)	
ConID: Identifikator der Assoziation, die aufgelöst werden soll.	Entfernt die Assoziation *ConID* zwischen zwei Lernobjekten. Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.

[84] Zur Identifikation einer Beziehung ist ein Identifikator nicht unbedingt nötig; Beziehungen können über sourceID, targetID und ihre Parameter eindeutig identifiziert werden. Daher wurde im Datenmodell in Kapitel 3 auch kein Identifikator für Beziehungen eingeführt. Die Einführung eines Identifikators erfolgt hier mit Blick auf die Realisierung und hat rein technische Gründe.

ConIDs getAssociations (*ObjID*)	
ObjID: Identifikator des Lernobjekts, von dem aus alle als Ergebnis gelieferten Assoziationen ausgehen.	Liefert eine Menge Con*IDs* von Identifikatoren von Assoziationen, die von Modul bzw. Kurs *ObjID* ausgehen.
ConIDs: Menge von Identifikatoren aller Assoziationen, die von *ObjID* ausgehen.	Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.

Tabelle 6-3: Dienste des Lernobjekt-Service

Werden neue Lernobjekte oder neue Versionen von Lernobjekten mittels des Dienstes put() der Lernobjektzugriffsschicht eingespielt oder deren Beschreibungen (Parameter) geändert, so sorgt die Lernobjektzugriffsschicht für die Registrierung des Lernobjekts beim Lernobjekt-Service. Hierzu wird der Dienst register() aufgerufen, mit dem das Lernobjekt dem Lernobjekt-Service bekannt gemacht wird. Ist das Lernobjekt dem Lernobjekt-Service bekannt, kann es genutzt werden.

Um interessierten Designern und Nutzern die Existenz eines neuen Lernobjekts mitzuteilen, sendet die Lernobjektzugriffsschicht eine Nachricht an den Notifikations-Service, vgl. 6.3.6. Dieser informiert potenzielle Interessenten.

6.3.3 Verwaltung von Annotationen: Das Annotations-Repository

6.3.3.1 Aufgabe

Die Annotationen, die von den Designern und Nutzern eingegeben werden. müssen persistent abgelegt werden. Dies umfasst mehrere Problemkreise: Zum einen muss der Inhalt der Annotationen selbst (beispielsweise textuelle Anmerkungen, Skizzen, etc.) gespeichert werden (Speicherung des Inhalts einer Annotation). Zum anderen muss die Assoziation, die die Verbindung zwischen einem Lernobjekt und einer Annotation bzw. zwischen einer Annotation und einer weiteren, diese annotierenden Annotation abgelegt werden, denn sonst ist eine Zuordnung der Annotationen zu den annotierten Elementen (Lernobjekte, Kurse oder selbst wieder Annotationen) nicht mehr möglich. Bei beiden Problemkreisen müssen Zugriffsrechte berücksichtigt werden.

All dies sind Aufgaben, die das Annotations-Repository leistet. Das Annotations-Repository sorgt für einen einfachen und einheitlichen Zugriff auf alle Annotationen und ist damit neben dem Lernobjekt-Repository die zweite wichtige Datenquelle für die Authoring-Systeme, Lernsysteme und Annotations-Tools, mit denen die Designer und Nutzer arbeiten.

6.3.3.2 Aufbau und Funktionsweise

Bei der Speicherung und Verwaltung von Annotationen müssen folgende Rahmenbedingungen berücksichtigt werden: Annotationen können frei zugänglich sein, während die Lernobjekte, auf die sich die Annotationen beziehen, nicht notwendigerweise frei verfügbar sind. Des weiteren müssen Annotationen, passende Rechte vorausgesetzt, für die Berechtigten zu jedem Zeitpunkt zugreifbar sein – unabhängig davon, ob der Verfasser einer Annotation gerade online ist oder nicht. Beide Rahmenbedingungen haben Auswirkungen auf die Art und Weise, wie Annotationen gespeichert werden.

Das Annotations-Repository besteht aus zwei Komponenten: Einem Datenspeicher, der die Annotationen sowie die Assoziationen zwischen einem annotierten Element (Lernobjekt, Kurs oder Annotation) und der Annotation selbst speichert, und eine Abstraktionsschicht, die den Zugriff auf die Annotationen und Assoziationen hinsichtlich der Zugriffsberechtigungen

und der Kapselung der verschienen Datenspeicherimplementierungen regelt. Diese Abstraktionsschicht wird *Annotationszugriffsschicht* genannt.

6.3.3.3 Speicherort des Annotations-Repositories

Die Annotationen sowie die Assoziationen, die die Annotationen mit den annotierten Lernobjekten oder Kursen verbinden, werden immer im Annotations-Repository gespeichert. Prinzipiell gibt es mehrere Orte, an denen das Annotations-Repository aufgebaut werden kann. Im folgenden werden einige Alternativen kurz erläutert und hinsichtlich ihrer Vor- und Nachteile bewertet.

Alternative 1: Realisierung des Annotations-Repositories auf den Rechnern der jeweils annotierten Lernobjekte

Die Annotationen werden in einem Annotations-Repository auf den Servern gespeichert, auf denen die zugehörigen Lernobjekte (falls sich die Annotationen auf den Inhalt der Lernobjekte beziehen) oder die zugehörigen Parameter (falls sich die Annotationen auf die Parameter der Lernobjekte beziehen) abgelegt sind. Ebenso werden Annotationen zu einer Annotation *A* in dem Annotations-Repository abgelegt, in dem bereits Annotation *A* gespeichert ist.

Bei dieser Art der Speicherung sind die Annotationen zu einem Lernobjekt einfach aufzufinden, da sie immer auf dem Server abgelegt sind, auf dem auch das Lernobjekt, auf das sie sich bezieht, gespeichert ist. Dies betrifft auch eine Annotation A_n, die selbst wieder eine Annotation A_{n-1} annotiert: In diesem Fall wird A_n im Annotations-Repository auf dem Server gespeichert, in dem bereits A_{n-1} abgelegt ist. Bei Diskussionen zu einem Lernobjekt *L*, die mittels Annotationen geführt werden und die über Antworten, Einwänden, Erwiderungen, etc. zu „Annotations-Threads“ (ähnlich den Threads in Usenet-Newsgroups) führen, liegen damit alle Annotationen A_i auf dem Rechner, auf dem Lernobjekt *L* gespeichert ist.

Zum Auffinden von Annotationen genügt es zu wissen, auf welche Lernobjekte sie sich beziehen und wo diese Lernobjekte gespeichert sind. Dies ist auf einfache Weise über eine Suchanfrage beim Lernobjekt-Service zu erreichen (Suchparameter: Identifikator des Lernobjekts). Auf dem Rechner, auf dem ein Lernobjekt gespeichert ist, sind dann auch alle Annotationen zu diesem Lernobjekt zu finden.

Ein wesentlicher Nachteil ist, dass auf den Servern, auf denen die Lernobjekte angelegt sind, Daten (die Annotationen) gespeichert werden, die zwar mit den Lernobjekten zu tun haben, jedoch nicht unbedingt mit der Organisation oder dem Unternehmen, das die Lernobjekte bereitstellt. Konkret würden beispielsweise bei einem Unternehmen, das eine große Zahl von Lernobjekten vertreibt, alle Annotationen zu diesen Lernobjekten in Annotations-Repositories auf den Servern des Unternehmens abgelegt werden – ein nicht unerheblicher Kostenfaktor (Netzkapazität und Kosten für die Bereitstellung ausreichend dimensionierter Datenspeicher) und aufgrund des offenen Zugangs über das Internet eine potenzielle Sicherheitslücke für das Unternehmen.

Alternative 2: Realisierung des Annotations-Repositories auf den Client-Rechnern

Die Annotationen werden auf den Client-Rechnern, auf denen sie verfasst wurden, gespeichert.

Diese Alternative hat den Vorteil, dass Speicherkosten für Annotationen (so diese Kosten aufgrund der Größe und Menge der Annotationen überhaupt eine beachtenswerte Höhe erreichen) nur bei den Verfassern der Annotationen anfallen. Außerdem ist so der Schutz der privaten Annotationen gewährleistet: Sie sind von außen gar nicht zugreifbar.

Diese Alternative ist allerdings aus anderen Gründen problematisch: Durch die Speicherung von Annotationen auf den Client-Rechnern sind die Annotationen nicht mehr jederzeit ver-

fügbar (ausgenommen sind hier die privaten Annotationen; diese dürfen lediglich vom Verfasser genutzt werden), sondern nur noch dann, wenn der Client-Rechner, auf dem eine fragliche Annotation abgelegt ist, online ist. Dies erschwert die Arbeit mit Annotationen sehr, sind doch möglicherweise nicht mehr alle Annotationen, die für Korrekturen an Lernobjekten oder Umstrukturierungen von Kursen nötig sind, verfügbar. Ein weiterer Nachteil liegt im Auffinden bestimmter Annotationen: Es wird ein Verzeichnis benötigt, das für jede Annotation (anhand ihres Identifikators) den Speicherort angibt. Die bei vielen Client-Rechnern übliche IP-Adressvergabe mittels DHCP[85] sorgt in diesem Zusammenhang für Probleme, denn wenn ein Client-Rechner täglich wechselnde IP-Adressen vom DHCP-Server zugewiesen bekommt, wird das Verzeichnis, das die Zuordnung „Annotation – Client-Rechner" vornimmt, zwangsläufig inkonsistent.

Alternative 3: Realisierung des Annotations-Repositories auf den Rechnern des Lernobjekt-Service

Die Annotationen werden auf den Rechnern, auf denen der Lernobjekt-Service läuft, gespeichert.

Im Gegensatz zu Alternative 2 ist das Auffinden von Annotationen kein Problem – die Adresse des Rechners bzw. der Rechner, auf denen der Lernobjekt-Service läuft, ist allgemein bekannt und ändert sich nicht. Der Zugriff auf Annotationen ist somit immer möglich.

Allerdings bleiben bei dieser Alternative die Kosten, die für die Speicherung der Annotationen anfallen, ungeklärt. Die Server, die eigentlich dafür gedacht sind, den Lernobjekt-Service am Laufen zu halten, werden nun zusätzlich mit der Speicherung von Daten (den Annotationen) belastet, die nicht einzelnen Organisationen oder Unternehmen zugeordnet werden können. Eine verursachergerechte Zuordnung dieser zusätzlichen Kosten ist damit nur schwer möglich.

Alternative 4: Realisierung des Annotations-Repositories auf dritten Rechnern

Die Annotationen werden auf Rechnern, die keine organisatorische Beziehung zu den Client-Rechnern oder den Lernobjekt-Server-Rechnern haben, gespeichert.

Diese Alternative ist hier lediglich der Vollständigkeit halber aufgeführt, denn sie bietet keine substanziellen Vorteile gegenüber den bisher genannten Alternativen. Genau wie in Alternative 2 ist hier zum Auffinden von Annotationen ein Verzeichnis nötig, das die Abbildung „Annotation → Server-IP-Adresse" realisiert. Zusätzlich handelt man sich mit der Nutzung dritter Rechner organisatorischen Mehraufwand ein, denn neben den Betreibern der Lernobjekt-Repositories (Organisationen und Unternehmen) und der Server, auf denen der Lernobjekt-Service läuft, hat man nun mit einem weiteren Betreiber zu tun: dem Betreiber des Verzeichnisses zum Auffinden der Annotationen. In dieser Alternative müssen drei Betreiber bei der Stange gehalten werden, wodurch die Attraktivität dieser Alternative spürbar sinkt.

Ergebnis

In der Summe ist Alternative 3 an sinnvollsten, auch wenn die Verrechnung der Kosten für die Speicherung der Annotationen nicht trivial ist. In diesem Problemfeld – falls dieses bei konkreten Installationen überhaupt relevant ist – kann man jedoch im Gespräch mit allen Beteiligten zu praktikablen, für alle zufriedenstellenden Verrechnungsmodellen kommen, so dass hierin kein Hinderungsgrund zu sehen ist.

---

[85] DHCP (Dynamic Host Configuration Protocol) ist ein Protokoll, mit dem Client-Rechnern dynamisch IP-Adressen zugeordnet werden können.

Alle Annotationen werden nun in einem Annotations-Repository gespeichert, das auf den Rechnern aufgebaut wird, auf denen schon der Lernobjekt-Service läuft. Die Assoziation, die die Verbindung des annotierten Elements (Lernobjekt, Kurs, Annotation) mit der Annotation vornimmt, wird im selben Annotations-Repository wie die Annotationen abgelegt. So werden Aufwand und Kosten für die Anlage eines weiteren Repositories bzw. der Einrichtung eines weiteren Verzeichnisses vermieden und es ist sichergestellt, dass auf die Annotationen immer zugegriffen werden kann. Zudem kann der Zugriff auf die Annotationen sehr performant realisiert werden.

Um einen schnelleren Zugriff zu erreichen und eine offline-Nutzung zu ermöglichen, bietet es sich an, einen Teil der Annotationen und der zugehörigen Assoziationen mittels Caching auf den Client-Rechnern zwischenzuspeichern.

6.3.3.4 Die Annotationszugriffsschicht

Entsprechend dem Ergebnis des letzten Absatzes wird für die Speicherung der Annotationen auf den Rechnern, auf denen schon der Lernobjekt-Service läuft, das Annotations-Repository eingerichtet. Das Annotations-Repository besteht aus zwei Komponenten: Einem Datenspeicher, der die Annotationen sowie die Assoziationen, die die annotierten Lernobjekte mit den Annotationen verbinden, speichert, und einer Annotationszugriffsschicht, die – analog zur Lernobjektzugriffsschicht in 6.3.2.3 – die Verwaltung der Annotationen übernimmt. Die Annotationszugriffsschicht hat die Aufgabe, einen transparenten Zugriff auf Annotationen zu ermöglichen und den Zugriff zu steuern. Hierbei müssen die Zugriffsrechte der Annotationen berücksichtigt werden.

Zugriffsschutz

Für den Zugriff auf Lernobjekte musste neben einer einheitlichen Zugriffsschnittstelle auch eine einheitliche Schnittstelle zu einer Vielzahl von Rechtesystemen realisiert werden. Dies lag daran, dass die Lernobjekte direkt bei den Unternehmen und Organisationen gespeichert sind, die sie erstellen, und jedes Unternehmen und jede Organisation seine eigene Sicherheitsinfrastruktur betreibt. Bei Annotationen ist dies nicht nötig, da die Annotationen in einem eigenen Annotations-Repository auf den Rechnern des Lernobjekt-Service gespeichert und verwaltet werden. Für die Annotationszugriffsschicht kann somit ein einheitliches Rechtesystem implementiert werden.

In 5.2.3.3 wurden die unterschiedlichen Berechtigungen zum Zugriff auf Annotationen bereits genannt: Private Annotationen sind nur für ihren Verfasser sicht- und zugreifbar und Gruppenannotationen nur für die Mitglieder einer bestimmten, registrierten Gruppe von Nutzern und Designern. Öffentliche Annotationen schließlich sind für jeden sicht- und zugreifbar. Annotationen können nur erzeugt werden und sind dann immer nur lesbar; Änderungsrechte für bereits existierende Annotationen gibt es nicht. Es ist allerdings möglich, Annotationen selbst wieder zu annotieren, um so quasi auf Annotationen zu antworten.

Im folgenden wird ein sehr einfaches Rechtesystem vorgestellt, wie es im Rest dieser Arbeit weiter verwendet wird. Es sind hier ausgefeilte Rollen- und Rechtesysteme einsetzbar, in dieser Arbeit soll hierauf aber nicht weiter eingegangen werden.

> Jeder Nutzer und jeder Designer hat eine eigene Kennung und wird über diese identifiziert. Ein Nutzer kann Mitglied in einer beliebigen Anzahl von Gruppen sein; eine Gruppe wird hier einfach als ungeordnete Menge von Kennungen repräsentiert. Abbildung 6-5 illustriert die Zusammenhänge.
>
> Da in jeder Annotation die Kennung ID_V ihres Verfassers vermerkt ist, ist es einfach zu entscheiden, ob ein Nutzer bzw. Designer mit einer bestimmten Kennung ID_N eine Annotation sehen darf oder nicht:

- Falls $ID_V = ID_N$, so ist die Annotation immer sichtbar und zugreifbar.
- Falls ID_V und ID_N in einer Gruppe sind und die Annotation eine Gruppenannotation[86] ist, ist sie sichtbar und zugreifbar.
- Falls die Annotation eine öffentliche Annotation ist, ist sie sichtbar und zugreifbar.
- In allen anderen Fällen ist die Annotation weder sichtbar noch zugreifbar.

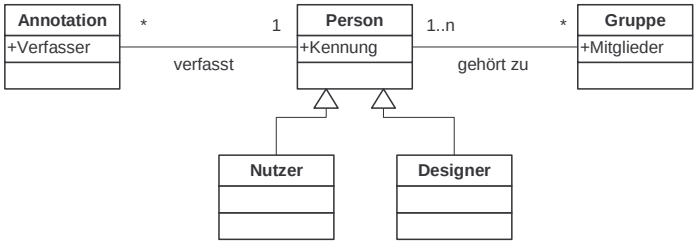


Abbildung 6-5: Zusammenhang Annotation – Nutzer / Designer – Gruppe

Wird bei Anforderungen von Annotationen die Kennung des Anforderers mit angegeben, kann so sehr einfach ermittelt werden, ob der Anforderer die Annotation einsehen darf.

Dienste der Annotationszugriffsschicht

Die Annotationszugriffsschicht muss zwei Gruppen von Diensten zur Verfügung stellen: Dienste zum Zugriff auf Annotationen und Dienste zur Verwaltung von Gruppen[87]. Tabelle 6-4 liefert einen Überblick über die Dienste zum Zugriff auf Annotationen, Tabelle 6-5 zeigt die Dienste zur Gruppenverwaltung.

Werden mittels der Dienste putAnnotation() und putAnnotationText() neue Annotationen zu einem Lernobjekt oder einer bereits bestehenden Annotation verfasst, so werden alle die Nutzer und Designer benachrichtigt, die mindestens Leserechte auf das annotierte Lernobjekt bzw. die Annotation haben[88]. Dies geschieht, indem die Annotationszugriffsschicht eine Nachricht an den Notifikations-Service (vgl. 6.3.6) schickt.

Neben den Diensten zum Zugriff liefert Tabelle 6-5 einen Überblick über die Gruppenverwaltung, die die Annotationszugriffsschicht bietet.

[86] Vgl. Parameter access in 5.4.1.2.

[87] Dies unterscheidet die Annotationszugriffsschicht von der Lernobjektzugriffsschicht des Lernobjekt-Repositories: Die Lernobjekte im Lernobjekt-Repository sind üblicherweise bereits durch (meist proprietäre) Zugriffsschutzsysteme vor unberechtigten Zugriffen geschützt. Die Lernobjektzugriffsschicht muss daher im Gegensatz zur Annotationszugriffsschicht keine eigene Rechteverwaltung realisieren, sondern statt dessen einen möglichst transparenten Zugriff auf die Lernobjekte unter Berücksichtigung des Zugriffsschutzsystems sicherstellen.

[88] Dies kann mittels einer Anfrage beim Lernobjekt-Repository bzw. Annotations-Repository unter Verwendung der in Tabelle 6-5 genannten Dienste leicht ermittelt werden.

Parameter und Ergebnis	**Beschreibung**
void putAnnotation(*ObjID*, *ObjArea*, *AnID*, *CourseID*, *UserID*)	
ObjID: Identifikator des Lernobjekts bzw. der Annotation, die mit der Annotation *AnID* verbunden werden soll[89] *ObjArea*: Beschreibung des Ausschnitts des Lernobjekts bzw. der Annotation, die annotiert wird[90] *AnID*: Identifikator der zu setzenden Annotation[91] *CourseID*: Identifikator des Kurses, in dem die Annotation gültig ist. Ist die Annotation nicht an einen Kurs gekoppelt, so ist *CourseID* Null. *UserID*: Kennung des die Annotation setzenden Nutzers	Hängt eine bereits existierende Annotation mit dem Identifikator *AnID* an Lernobjekt oder eine Annotation *ObjID*. Hierbei wird vorausgesetzt, dass die Annotation bereits als Modul mit dem Identifikator *AnID* gekapselt ist. Die Assoziation zwischen *ObjID* und *AnID* wird von der Annotationszugriffsschicht erzeugt, vgl. 7.4.3.3. Eine Prüfung der Zugriffsrechte ist hier nicht nötig, da jeder Annotationen zu beliebigen Lernobjekten verfassen kann. (Eine Authentifizierung von *UserID* wird jedoch durchgeführt.) Dieser Dienst ermöglicht es, auch bereits bestehende Lernobjekte oder Annotationen als Annotationen für andere Lernobjekte zu setzen.
void putAnnotationText[92](*ObjID*, *ObjArea*, *Text*, *CourseID*, *UserID*)	
ObjID: Identifikator des Lernobjekts bzw. der Annotation, die mit der Annotation verbunden werden soll *ObjArea*: Beschreibung des Ausschnitts des Lernobjekts bzw. der Annotation, die annotiert wird *Text*: Anmerkung des Nutzers als Freitext *CourseID*: Identifikator des Kurses, in dem die Annotation gültig ist. Ist die Annotation nicht an einen Kurs gekoppelt, so ist *CourseID* Null. *UserID*: Kennung des die Annotation setzenden Nutzers	Erzeugt ein neues Modul und ein neues Atom, fügt den Freitext *Text* an das Atom an, fügt das Atom in das Modul ein und verbindet das Lernobjekt bzw. die Annotation *ObjID* mit diesem Modul mittels einer Assoziation, die die Annotationszugriffsschicht erzeugt. Eine Prüfung der Zugriffsrechte ist hier nicht nötig, da jeder Annotationen zu beliebigen Lernobjekten verfassen kann. (Eine Authentifizierung von *UserID* wird jedoch durchgeführt.) Dieser Dienst bietet eine komfortable Variante des Dienstes putAnnotation(), wenn rein textuelle Annotationen erstellt werden sollen

[89] Vgl. Parameter anchor-id in 7.2.1.3.
[90] Vgl. Parameter anchor-area in 7.2.1.3.
[91] Vgl. Parameter annotation-id in 7.2.1.3.
[92] Analog zu dieser Methode kann auch ein Atom erzeugt werden, an das nicht nur Text, sondern auch ein Bild oder ein anderes multimediales Element anfügt.

Object getAnnotation(*AnID, UserID*)	
AnID: Identifikator der angeforderten Annotation *UserID*: Kennung des anfordernden Nutzers *Object*: Die angeforderte Annotation	Liefert die Annotation mit dem Identifikator *AnID*. Der Zugriff wird anhand der Kennung *UserID* geprüft.
*Annotation*s getAnnotations (*ObjID, CourseID, UserID*)	
ObjID: Identifikator des Lernobjekts, dessen Annotationen geliefert werden sollen *CourseID*: Identifikator des Kurses, in dem die Annotation gültig ist. Ist die Annotation nicht an einen Kurs gekoppelt, so ist *CourseID* Null. *UserID*: Kennung des anfordernden Nutzers *Annotation*s: Menge von Annotationen	Liefert die Menge der Annotationen, die für Lernobjekt *ObjID* verfasst wurden und für Nutzer *UserID* sichtbar sind.
*AnID*s getAnnotationIDs(*ObjID, CourseID, UserID*)	
ObjID: Identifikator des Lernobjekts, dessen Annotationen geliefert werden sollen *CourseID*: Identifikator des Kurses, in dem die Annotation gültig ist. Ist die Annotation nicht an einen Kurs gekoppelt, so ist *CourseID* Null. *UserID*: Kennung des anfordernden Nutzers *AnID*s: Menge von Identifikatoren von Annotationen	Liefert die Menge der Identifikatoren der Annotationen, die für Lernobjekt *ObjID* in Kurs *CourseID* verfasst wurden und für Nutzer *UserID* sichtbar sind.
void deleteAnnotation(*AnID, UserID*)	
AnID: Identifikator der zu löschenden Annotation *UserID*: Kennung des die Annotation löschenden Nutzers	Löscht Annotation *AnID*, indem die Assoziation zwischen Annotation *AnID* und dem annotierten Lernobjekt gelöscht wird, falls Nutzer *UserID* hierzu die Berechtigung hat.

Tabelle 6-4: Zugriffsdienste der Annotationszugriffsschicht

Parameter und Ergebnis	**Beschreibung**
void createGroup(G*roupID, UserID*)	
GroupID: Identifikator (eine beliebige Zeichenkette) der zu erzeugenden Gruppe *UserID*: Kennung des Nutzers, der die Gruppe erzeugt	Erzeugt eine Gruppe *GroupID* mit Nutzer *UserID* als einzigem Mitglied. Jeder darf Gruppen erzeugen.

void deleteGroup(*GroupID, UserID*)	
GroupID: Identifikator der zu löschenden Gruppe *UserID*: Kennung des Nutzers, der die Gruppe löschen will	Löscht Gruppe *GroupID*, falls *UserID* die Rechte hierzu hat.
void addToGroup(*GroupID, UserID*)	
GroupID: Identifikator der Gruppe, zu der Nutzer *UserID* hinzugefügt werden soll *UserID*: Kennung des Nutzers, der zu Gruppe *GroupID* hinzugefügt werden soll	Fügt den Nutzer mit der Kennung *UserID* zu Gruppe *GroupID* hinzu.
void removeFromGroup (G*roupID, UserID*)	
GroupID: Identifikator der Gruppe, aus der ein Nutzer entfernt werden soll *UserID*: Kennung des Nutzers, der aus Gruppe *GroupID* entfernt werden soll	Entfernt den Nutzer mit der Kennung *UserID* aus Gruppe *GroupID*.
UserIDs getUsers(*GroupID*)	
GroupID: Identifikator der Gruppe, deren Mitglieder ermittelt werden sollen *UserIDs*: Menge der Kennungen aller Nutzer, die in Gruppe *GroupID* sind	Liefert die Identifikatoren all der Nutzer, die Mitglied in Gruppe *GroupID* sind.
GroupID getGroup(*UserID*)	
UserID: Kennung des Nutzers, dessen Gruppen ermittelt werden sollen *GroupIDs*: Menge der Identifikatoren der Gruppen, in denen Nutzer *UserID* Mitglied ist	Liefert die Identifikatoren all der Gruppen, in denen Nutzer *UserID* Mitglied ist
visible isVisible[93](*AnID, UserID*)	
AnID: Identifikator einer Annotation *UserID*: Kennung des Nutzers, für den Annotation *AnID* geprüft werden soll *visible*: Boolescher Wert	Prüft, ob die Annotation mit dem Identifikator *An-ID* für den Nutzer mit der Kennung *UserID* sichtbar ist.

[93] Da jeder, der ein Zugriffsrecht auf das Annotationssystem besitzt, eine Annotation erstellen kann und eine bereits erstellte Annotation nicht verändert werden darf, um die Semantik nicht nachträglich verändern zu können, reicht es hier, nur die Überprüfung des Leserechts durchzuführen. Das gleiche gilt auch für private Annotationen.

void authenticate(*UserID, Password*)	
UserID: Identifikator des zu authentifizierenden Nutzers. *Password:* Passwort von Nutzer *UserID*	Führt die Authentifizierung der Nutzers *UserID* bei der Annotationszugriffsschicht durch. Die Authentifizierung erfolgt mittels Kennung und Passwort.

Tabelle 6-5: Gruppenverwaltungsdienste der Annotationszugriffsschicht

6.3.4 Nutzung von Annotationen: Annotationseditor und Annotations-Viewer

Designer und Nutzer arbeiten beide mit Annotationen. Sie annotieren Stellen in Lernobjekten, lesen die Annotationen anderer Nutzer und / oder Designer und annotieren vielleicht selbst wieder Annotationen.

Designer und Nutzer haben jedoch eine grundsätzlich unterschiedliche Sicht auf Lehrmaterialien: Designer (Modul-, Kurs- und Atomdesigner) sowie im Fall „Präsentation eines adaptierbaren oder customized Kurses“ die Kurs-Presenter haben eine Elementsicht[94] auf die Lehrmaterialien. Sie sehen nur Lernobjekte, also Module, Kurse und Atome, nicht notwendigerweise jedoch die Lehrmaterialien in einem bestimmten Präsentationsformat wie beispielsweise HMTL oder PDF. Lerner hingegen sehen ausschließlich Kurse[95], die bereits in einem bestimmten Format (HTML, PDF, etc.) vorliegen; sie können a priori nicht erkennen, wie ein Kurs in einzelne Lernobjekte aufgeteilt ist[96].

Diese grundsätzlich verschiedenen Sichten haben Einfluss darauf, wie Designer und Nutzer mit Annotationen arbeiten können. In den folgenden Abschnitten wird hierauf näher eingegangen.

6.3.4.1 Eingabe und Bearbeitung von Annotationen aus Sicht der Designer

Designer (Atom-, Modul- und Kursdesigner) und teilweise auch Kurs-Presenter haben eine Elementsicht auf Lehrmaterialien und Annotationen; sie sehen also Atome, Module und Kurse. Dies sind auch die Elemente, die von den Designern annotiert werden bzw. die anderen Lernobjekten und Annotationen selbst als Annotationen dienen können (Verwendung bereits bestehender Atome oder Module als Annotationen; die Verwendung bereits bestehender Kurse als Annotationen ist zwar denkbar, wird aber nicht oft genutzt werden).

Die präsentationsfertig formatierten Kurse, wie sie die Kurs-Presenter oder die Kurstutoren den Lernern nahe bringen, werden von den Designern nicht so oft verwendet. Diese Formen der Kurse werden von den Designern in vielen Fällen nur zur Kontrolle des Layouts und der didaktischen Wirkung der Elemente verwendet. Kurs-Presenter können Lehrmaterialien in beiden Gestalten sehen: Als Hierarchie von Lernobjekten (wie die Designer) oder präsentationsfertig zum Einsatz in der Lehre (für die Lerner).

Die Sichtbarkeit von Annotationen hängt von deren Zugriffsrechten ab. Die meisten Annotationen von Designern haben solche Rechte, dass die Lerner sie nicht sehen können – es sei denn, die Designer geben bestimmte Annotationen explizit für die Lerner frei, beispielsweise

---

[94] Je nach Designer ist dies eine Sicht auf hierarchisch strukturierte Lernobjekte (beispielsweise bei Kurs- oder Moduldesignern) oder eine Sicht auf eine unstrukturierte Menge von Lernobjekten (beispielsweise bei Atomdesignern).

[95] Ausgenommen sind hier die Fälle „Lernen durch Nutzung von adaptierbaren Kursen“ sowie „Lernen durch Aufbau und Nutzung von customized Kursen“.

[96] In vielen Fällen wird man erkennen können, wo ein Lernobjekt endet und wo ein neues beginnt. Besonders deutlich wird dies, wenn zwei Atome in verschiedenen Formaten aufeinander treffen, beispielsweise wenn eine Microsoft Powerpoint-Folie auf einen offensichtlich in LaTeX formatierten Absatz folgt.

um Korrekturvorschläge oder Feedback zu diskutieren. Im allgemeinen jedoch haben Annotationen von Designern mit Aspekten des Lehrmaterialerstellungsprozesses (Koordination des Prozesses, Feedback, etc.) zu tun und sind für die Lerner damit uninteressant. Umgekehrt jedoch können die Designer die (nicht-privaten) Annotationen der Lerner sehen, denn diese Annotationen können wertvolle Hinweise auf Verbesserungen und Korrekturen an Lernobjekten beinhalten.

Im folgenden wird beschrieben, wie Annotationen bei der Elementsicht gelesen und selbst verfasst werden können. Das Lesen und Verfassen von Annotationen von präsentationsfertigen Lehrmaterialien wird in 6.3.4.2 besprochen.

Zugriff auf Lehrmaterialien und Annotationen

Die Designer arbeiten unmittelbar mit Lernobjekten oder Kursen, die sie gegebenenfalls annotieren. Die Authoring- und Annotations-Tools, die die Designer nutzen, greifen direkt auf das Annotations-Repository und (über den Lernobjekt-Service und die Lernobjektzugriffsschicht) auf das Lernobjekt-Repository zu. Hierbei findet eine Authentifizierung des Anfragers und eine darauf aufbauende Zugriffskontrolle statt.

Wenn die Designer die Lernobjekte und Kurse beim Lernobjekt-Repository anfordern, geschieht dies über die Identifikatoren der Lernobjekte und Kurse, die den Lernobjektzugriffsschichten des Lernobjekt-Repositories mitgeteilt werden. Mittels diesen Identifikatoren können beim Annotations-Repository die zugehörigen Annotationen ermittelt, abgerufen (vgl. 6.3.3.4) und in den Authoring- und Annotations-Tools in geeigneter Form (beispielsweise in einem Extra-Fenster oder kombiniert mit den Lernobjekten) dargestellt werden.

Benutzerschnittstelle

Ausgangspunkt für die Arbeit mit den Annotations-Tools ist die Liste der verfügbaren Lernobjekte sowie die für sie verfassten Annotationen. Die Designer arbeiten mit den Lernobjekten und bekommen dabei die Annotationen, die den gerade bearbeiteten Lernobjekten angeheftet wurden, angezeigt – ausreichende Rechte des jeweiligen Designers vorausgesetzt. Abbildung 6-6 zeigt eine mögliche Darstellungsform, wie sie im Prototyp realisiert wurde: Sind Annotationen zu einem Lernobjekt verfügbar, können Sie durch Klick auf den entsprechenden Link angezeigt werden.

Ein hierzu alternativer Zugang zu den Lernobjekten und damit implizit auch zu den Annotationen ist über die Elementhierarchie (Baumstruktur) eines ausgewählten Moduls. Ausgehend von einem Modul, dessen Annotationen bei entsprechenden Rechten mit angezeigt werden, können dessen Submodule oder Atome ausgewählt werden; statt einen direkten Zugriff auf einzelne Lernobjekte, die beispielsweise in einer Liste aufgeführt werden, zu haben, navigiert man durch die Elementhierarchie zu dem gewünschten Modul.

Folgende Aktionen, ausreichende Berechtigungen vorausgesetzt, sind möglich:

- Hinzufügen von Annotationen zu Lernobjekten oder wiederum zu Annotationen

  Neben dem Inhalt können hierbei alle Teile von Lernobjekten und Annotationen annotiert werden, also Parameter, Constraints auf den Parametern und natürlich Teile des Inhalts.

- Anzeigen von Informationen zu Annotationen (Verfasser, Erstellungsdatum, etc.)

- Löschen von Annotationen, sofern die Annotation selbst noch nicht annotiert ist.

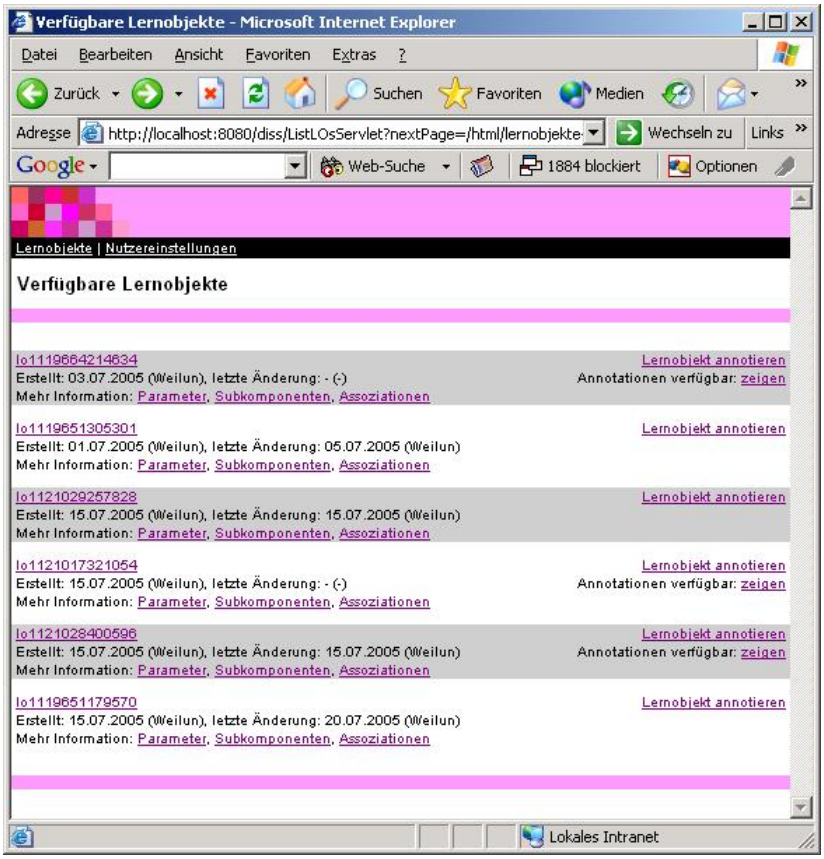


Abbildung 6-6: Liste der verfügbaren Lernobjekte im Prototyp (Screenshot)

Es ist nicht möglich, bereits existierende Annotationen zu löschen, wenn sie selbst bereits wieder annotiert sind. So eine Modifikation würde den Argumentationsfaden, dem die Sequenz der Annotationen ganz ähnlich einem Usenet-Thread folgt, verfälschen und nicht mehr nachvollziehbar machen.

6.3.4.2 Eingabe und Bearbeitung von Annotationen aus Sicht der Lerner

Lerner sehen nur präsentationsfertige Lehrmaterialien in PDF, HTML, Postscript, etc.; von deren hierarchischem Aufbau aus Atomen und Modulen sehen sie nichts. In präsentationsfertigen Lehrmaterialien sind bestimmte Annotationen bereits enthalten, beispielsweise als Fußnoten, Querverweise, in Popup-Fenstern (bei HTML als Präsentationsformat), etc. Wichtig ist hier zu berücksichtigen, dass bei der Erstellung der präsentationsfertigen Lehrmaterialien nicht alle Annotationen aufgenommen werden, sondern nur solche, die öffentlich sichtbar sind. Dies liegt daran, dass die fertigen Lehrmaterialien beliebig verteilt werden können und daher keinerlei Schutz privater Informationen (bei privaten Annotationen oder Gruppenannotationen) mehr durchgesetzt werden kann.

In präsentationsfertigen Lehrmaterialien sind die Annotationen damit fest in die Lehrmaterialien integriert und in ihrer Gestaltung an das Präsentationsformat angepasst. Viele Präsentationsformate erlauben allerdings keine unmittelbaren Annotationen von bestimmten Stellen der Lehrmaterialien. So ist es beispielsweise nicht so einfach möglich, HTML-Seiten zu annotieren. Hierzu sind zusätzliche Programme – für Web-Browser beispielsweise Plugins – nötig, die den Lernern extra bereitgestellt werden müssen.

Zugriff auf Lehrmaterialien und Annotationen

Die Lerner, und teilweise auch Kurs-Presenter und Kurstutoren, arbeiten mit präsentationsfertigen Lehrmaterialien, die potenziell überall abgelegt sein können; sie greifen damit nicht notwendigerweise auf das Lernobjekt-Repository zu. Die Annotations-Tools arbeiten hingegen direkt mit dem Annotations-Repository.

Wird eine Annotation in einem Annotations-Tool eingegeben, so führt das Annotations-Tool ein Mapping des markierten und zu annotierenden Lehrmaterialausschnitts auf eine Menge von Lernobjekten durch. Je nach Realisierung der Annotations-Tools können hierbei Zugriffe auf das Lernobjekt-Repository nötig sein. Anschließend fügt das Annotations-Tool die Annotation sowie die Assoziationen zwischen den fraglichen Lernobjekten und der Annotation ins Annotations-Repository ein. Hierbei wird auf das Annotations-Repository zugegriffen.

Benutzerschnittstelle

Wesentlich für die Annotations-Tools der Lerner ist eine möglichst enge Integration in die Lehrmaterial-Viewer (Web-Browser, PDF-Viewer, etc.). Für Lehrmaterialien in HTML wäre eine enge Integration in den Web-Browser nötig, für einen Satz Microsoft Powerpoint-Folien wären Erweiterungen von Powerpoint-Viewern denkbar, etc.

Wie auch immer die konkrete Implementierung der Annotations-Tools aussieht (als Plugin im Web-Browser, als ActiveX-Control zur Erweiterung von Microsoft Powerpoint, als eigenständiges Programm, etc.), so müssen doch immer die gleichen Schritte durchgeführt werden:

1. Zuerst muss das Annotations-Tool erkennen, welcher Ausschnitt der Lehrmaterialien annotiert werden soll.
2. Dieser Ausschnitt muss auf einen Teilbaum der Atom- und Modulhierarchie des Kurses, der den Lehrmaterialien zu Grunde liegt, abgebildet werden[97]. Hierzu ist eine Interaktion zwischen dem Annotations-Tool und dem Lernobjekt-Repository, das den Teilbaum enthält, nötig. Da die Identifikation des zu annotierenden Ausschnitts stark vom verwendeten Präsentationsformat der Lehrmaterialien abhängt, ist es sinnvoll, diesen Schritt nicht im Lernobjekt-Repository, sondern im Annotations-Tool durchzuführen. So kann leichter auf die Eigenschaften und Besonderheiten des jeweiligen Präsentationsformats eingegangen werden.
3. Anschließend muss die Annotation im Annotations-Tool selbst erfasst werden. Ist dies geschehen, kann die Annotation sowie die Assoziation zwischen der Annotation und dem annotierten Teilbaum im Annotations-Repository abgelegt werden.

6.3.5 Konsistenzsicherung von Annotationen: Der Lernobjekt-Scanner

Lernobjekte, wie sie in dieser Arbeit verstanden werden, sind keine statischen Daten, sondern unterliegen ständigen Veränderungen, die durch Designer (Modul-, Atom- und Kursdesigner) und Nutzer (Kurs-Presenter, Kurstutoren und Lerner) angestoßen werden. Es werden Annotationen (Feedback-Annotationen, Hinweise, Korrekturvorschläge, etc.) verfasst und an die betreffenden Lernobjekte bzw. Annotationen geheftet, woraufhin evtl. Änderungen in den Lernobjekten durchgeführt werden. Durch die Änderungen wiederum können die Annotationen ihre Grundlage verlieren.

[97] Ein Algorithmus hierzu ist in 7.4.4 im Detail beschrieben.

Das Problem wird noch deutlicher, wenn ein Lernobjekt, das mit einer Annotation versehen wurde, aus dem fraglichen Kurs entfernt wird. Was soll dann mit der Annotation geschehen?

Die beiden gerade beschriebenen Fälle können die Konsistenz zwischen den Elementen des Lernobjekt-Repositories (die Lernobjekte) und den Elementen des Annotations-Repositories (den Annotationen) gefährden. Es wird daher eine Komponente benötigt, die solche Problemfälle aufdeckt und gegebenenfalls entsprechende Maßnahmen zur Konsistenzsicherung durchführt. Diese Komponente ist der *Lernobjekt-Scanner*.

6.3.5.1 Inkonsistenzen zwischen Lernobjekten und Annotationen bzw. zwischen Annotationen

Inkonsistenzen zwischen Lernobjekten und Annotationen können auf verschiedene Weise entstehen. Der Lernobjekt-Scanner erkennt folgende Probleme:

a) Ein Lernobjekt, dem eine Annotation angeheftet ist, wird geändert.

   Die Änderung des Lernobjekts könnte die Ursache, warum die Annotation an das Lernobjekt angefügt wurde, beseitigen. In diesem Fall hätte die Annotation ihren Zweck erfüllt und der Verfasser der Annotation könnte darüber nachdenken, ob die Annotation gelöscht oder aus Gründen der Nachvollziehbarkeit der Änderungen beibehalten werden soll. Über die Notwendigkeit der Annotation kann jedoch nicht der Lernobjekt-Scanner entscheiden, dies kann nur der Verfasser der Annotation. Der Lernobjekt-Scanner markiert daher die Annotation und initiiert über den Notifikations-Service (vgl. 6.3.6) die Benachrichtigung ihres Verfassers.

b) Ein Lernobjekt, dem eine Annotation angeheftet ist, wird aus dem Kurs entfernt. Es bleibt jedoch im Lernobjekt-Repository verfügbar.

   In diesem Fall hat die Annotation in dem Zusammenhang, in dem sie verfasst wurde, keinen Bezugspunkt mehr: „Ihr“ Lernobjekt existiert in dem Kurs, in dessen Zusammenhang die Annotation verfasst wurde, nicht mehr. Den thematischen Rahmen der Annotation bildet hierbei der Kurs, den der Verfasser der Annotation gerade nutzte, als der die Annotation verfasst hat. Dieser Fall ist nur für Kursdesigner sowie die Nutzer von Kursen, also Kurs-Presenter, Kurstutoren und Lerner, von Interesse, denn nur diese betrachten ganze Kurse.

   Der Lernobjekt-Scanner darf die Annotation nicht automatisch löschen, denn dies würde vom Verfasser der Annotation womöglich nicht verstanden und als Fehler des Systems interpretiert werden. Falls die Annotation eine didaktische Funktion (vgl. 5.4.2.2) und weiteren Inhalt (beispielsweise wenn ein Lerner in eigenen Worten einen Absatz zusammenfasst) hat, würde durch ein automatisches Löschen sogar ein für alle Adressaten der Annotation schmerzhafter Informationsverlust entstehen. Statt dessen markiert der Lernobjekt-Scanner wieder die Annotation und initiiert über den Notifikations-Service die Benachrichtigung aller Kursdesigner sowie Nutzer, also Kurs-Presenter, Kurstutoren und Lerner, die mindestens Leserechte für die Annotation haben. Diese entscheiden dann, wie weiter mit der Annotation (und damit der Assoziation, die die Annotation mit dem Lernobjekt verbindet) umgegangen werden soll.

c) Ein Lernobjekt, dem eine Annotation angeheftet ist, wird dauerhaft gelöscht (aus dem Lernobjekt-Repository entfernt).

   Auch in diesem Fall geht der Bezugspunkt der Annotation verloren. Im Gegensatz zum Fall b) ist dies jedoch für alle Nutzer und alle Designer (auch für die Moduldesigner) von Interesse, denn das Lernobjekt steht für eine Nutzung nicht weiter zur

Verfügung; Annotationen, die an das Lernobjekt angeheftet wurden, sind nun bedeutungslos. Ebenso bleibt die Assoziation, die das Lernobjekt mit der Annotation verbindet, in einem inkonsistenten Zustand zurück. Auch hier ist aus den selben Gründen wie im Fall b) ein automatisches Löschen der Annotation nicht angebracht. Analog zum vorigen Fall markiert der Lernobjekt-Scanner wieder die Annotation und initiiert die Benachrichtigung – diesmal allerdings die Benachrichtigung aller Nutzer und Designer, die mindestens Leserechte für die Annotation haben[98] – über den Notifikations-Service, so dass wieder die Nutzer und Designer über den weiteren Umgang mit der Annotation und der Assoziation entscheiden können.

d) Ein Lernobjekt, das für ein anderes Lernobjekt als Annotation dient, wird gelöscht.

   Dieser Fall ist gleichbedeutend mit dem Löschen einer Annotation. Auch in diesem Fall bleibt jedoch eine inkonsistente Assoziation zurück, über die der Designer oder Nutzer, der die Assoziation zwischen den beiden Lernobjekten angelegt hat, informiert werden muss[99]. Der Designer oder Nutzer, der diese Assoziation angelegt hat, muss dann entscheiden, ob die Assoziation gelöscht oder auf ein anderes Lernobjekt „umgebogen" werden soll.

Der wesentliche Punkt bei allen Aktionen ist immer, dass der Lernobjekt-Scanner nicht von sich aus eine Bereinigung des Annotations-Repositories vornimmt und die möglicherweise „bezugslosen" Annotationen und / oder Assoziationen löscht. Statt dessen wird das Annotations-Repository bewusst in einem möglicherweise inkonsistenten Zustand belassen, wobei allerdings die problematischen Annotationen markiert werden, damit sie später leicht aufgefunden werden können. Wie letztlich mit den „bezugslosen" Annotationen und Assoziationen bzw. den in ihnen abgelegten Informationen umgegangen werden soll, liegt ausschließlich in der Verantwortung ihrer Verfasser.

Prüfung der Konsistenz

Um zu entscheiden, ob die Annotation *A* zu einem Lernobjekt *L* im Rahmen eines Kurses *K* konsistent ist, wird folgendes Prädikat consistent(*A*, *L*, *K*) über den Parametern von *A* und *L* ausgewertet:

$$\begin{aligned}\text{consistent}(A, L, K) := (&K = \text{undefined} \vee \text{isSubElement}(L, K)) \wedge \\ &((\text{last-modified}(A) > \text{last-modified}(L)) \vee \\ &(\text{created}(A) > \text{last-modified}(L)) \vee \\ &(\text{created}(A) > \text{created}(L)))\end{aligned}$$

IsSubElement(*L*, *K*) ist ein Prädikat, das angibt, ob Lernobjekt *L* in Kurs *K* als Submodul oder Atom vorkommt. Last-modified und created sind Parameter von Lernobjekten und Annotationen, die angeben, wann ein Lernobjekt bzw. eine Annotation erzeugt bzw. zuletzt geändert wurde.

Falls *A* bzw. *L* nicht existieren, wird consistent(*A*, *L*, *K*) nicht ausgewertet. Consistent(*A*, *L*, *K*) ist dann in jedem Fall false.

---

[98] In diesem Fall betrifft die Löschung des Lernobjekts alle Kurse und Lernobjekte, die das fragliche Lernobjekt verbaut haben. Damit ist dieses Ereignis potenziell für alle Designer und Nutzer von Interesse. In Fall b) oben wird das Lernobjekt nur aus einem Kurs entfernt und hat damit nur für den betreffenden Kurs eine Bedeutung. Dementsprechend ist dieses Ereignis auch nur für die Kursdesigner und Nutzer von Interesse, die mit dem fraglichen Kurs zu tun haben. Der Interessentenkreis ist damit deutlich kleiner.

[99] Durch das Löschen des Lernobjekts initiiert das Lernobjekt-Repository bereits eine Benachrichtigung aller der Nutzer und Designer, die mindestens Leserechte auf das gelöschte Lernobjekt hatten. Diese Nutzer und Designer müssen daher nicht auch noch vom Annotations-Repository über das Wegfallen des Lernobjekts informiert werden.

Das Prädikat consistent(*A*, *B*, *K*) zwischen zwei Annotationen *A* und *B* im Rahmen eines Kurses *K* ist analog definiert.

Umgekehrt sind also eine Annotation *A* und ein Lernobjekt *L* inkonsistent, wenn *L* nach dem Verfassen von *A* geändert oder gelöscht wird, also last-modified(*L*) > last-modified(*A*) oder last-modified(*L*) > created(*A*). Dies gilt analog für zwei Annotationen *A* und *B*, wenn Annotation *A* durch eine Annotation *B* annotiert wird. Auch hier sind *A* und *B* inkonsistent, wenn last-modified(*A*) > last-modified(*B*) oder last-modified(*A*) > created(*B*). Die Parameter last-modified und created können leicht über den Lernobjekt-Service und das Annotations-Repository abgefragt werden.

Falls die Konsistenz zwischen *A* und *L* ohne Bezug auf einen Kurs *K* ermittelt werden soll, wird *K* = undefined gesetzt.

Durchführung der Konsistenzprüfung

Um nun zu prüfen, ob eine Annotation *A* zu einem Lernobjekt *L* (ohne Bezug auf einen Kurs *K*) konsistent ist, berechnet der Lernobjekt-Scanner in regelmäßigen Abständen das Prädikat consistent(*A*, *L*, undefined). Die Berechnung muss paarweise zwischen allen Lernobjekten und Annotationen bzw. zwischen allen Annotationen durchgeführt werden[100]. Werden hierbei Inkonsistenzen entdeckt, so schickt der Lernobjekt-Scanner eine Nachricht an den Notifikations-Service. Dieser informiert potenzielle Interessenten über die Inkonsistenz.

6.3.6 Benachrichtigung von interessierter Prozessteilnehmer: Der Notifikations-Service

Annotiert ein Nutzer oder ein Designer ein Lernobjekt oder eine Annotation, so wird er im allgemeinen interessiert daran sein zu erfahren, wenn das annotierte Lernobjekt bzw. die annotierte Annotation geändert, gelöscht oder wiederum von anderen annotiert wird. Schließlich könnten die Änderungen am annotierten Lernobjekt bzw. an der annotierten Annotation seine Anmerkungen betreffen. Ein Beispiel hierfür wären Fehlerkorrekturen an einem Lernobjekt, die aufgrund eines Hinweises in einer Annotation durchgeführt werden. Umgekehrt ist sicher auch der Autor eines Lernobjekts daran interessiert zu erfahren, wenn sein Lernobjekt annotiert wird.

Damit die Verfasser von Annotationen an Änderungen in den von ihnen annotierten Lernobjekten oder Annotationen erfahren, müssen sie informiert werden. Das gleiche gilt auch für andere Aktionen wie beispielsweise das Anlegen, Ändern oder Löschen von Lernobjekten. Die Bereitstellung und Verteilung dieser Information ist Aufgabe einer eigenen Komponente in der Annotationsinfrastruktur. Diese Komponente wird *Notifikations-Service* genannt.

Bei der Information der Interessenten wird auf bereits bestehenden Kommunikationsinfrastrukturen aufgesetzt. So wird E-Mail, Instant Messaging, etc. genutzt, denn diese Kommunikationsmedien sind von den Designern und Nutzern akzeptiert und aus ihrer täglichen Arbeit bekannt.

6.3.6.1 Arten von Benachrichtigungen

Der Notifikations-Service unterstützt zwei Arten von Benachrichtigungen: Aktive und passive Benachrichtigungen. Ausgangspunkt einer Benachrichtigung ist immer ein Ereignis, beispielsweise das Verfassen einer Annotation zu einem Lernobjekt, aber auch eine mögliche

[100] Hierbei wird man sinnvollerweise nicht alle Lernobjekte und alle Annotationen prüfen, sondern man wird nur die Lernobjekte und Annotationen genauer betrachten, zwischen denen eine Assoziation vom Typ „annotiert“ oder einer Spezialisierung von „annotiert“ gemäß Tabelle 5-6 besteht.

Inkonsistenz zwischen einem Lernobjekt und einer Annotation, die vom Lernobjekt-Scanner ermittelt wurde.

Aktive Benachrichtigungen sind Hinweise, die an Interessenten eines Ereignisses verschickt werden. Ein Beispiel hierfür sind E-Mails, die der Notifikations-Service an die Interessenten schickt, oder Popup-Fenster.

Passive Benachrichtigungen sind dezenter und weniger aufdringlich. Sie treten erst dann in Erscheinung, wenn sich der Interessent mit dem Lernobjekt oder der Annotation, das das Ereignis ausgelöst hat, beschäftigt. Ein Beispiel hierfür sind farbliche Markierungen der fraglichen Annotationen.

Bei aktiven Benachrichtigungen übernimmt der Notifikations-Service die Benachrichtigung der Nutzer. Passive Benachrichtigungen hingegen werden anders behandelt: Für passive Benachrichtigungen hält der Notifikations-Service die Informationen, welche Annotationen potenziell inkonsistent sind, vor; das Abrufen dieser Informationen sowie die Darstellung der fraglichen Annotationen erfolgt in den Tools, die die Nutzer und Designer verwenden (beispielsweise Lehrmaterial-Viewer, Authoring-Tools, etc.). Hierzu rufen diese Tools die Informationen über passive Notifikationen vom Notifikations-Service ab. Diese Tools steuern auch, wann die Informationen über passive Notifikationen wieder aus dem Notifikations-Service entfernt werden. Dies kann beispielsweise geschehen, nachdem die entsprechende Notifikation den Designern angezeigt wurde.

6.3.6.2 Aufbau und Funktionsweise

Der Notifikations-Service besteht aus vier Komponenten: Ein Nachrichtenpuffer, der die vom Lernobjekt-Scanner oder anderen Komponenten übermittelten Ereignisdaten speichert, ein Nutzerprofil-Repository, das Angaben darüber speichert, wie die einzelnen Nutzer über Änderungen informiert werden wollen, einen Report-Generator, der die Informationen für die Nutzer entsprechend ihren Wünschen (niedergelegt im Nutzerprofil-Repository) zusammenstellt, und einen Nachrichten-Publisher, der die Verteilung der Informationen an die Nutzer übernimmt. Im folgenden werden diese Komponenten sowie ihr Zusammenspiel vorgestellt.

Nachrichtenpuffer

Der Nachrichtenpuffer nimmt Ereignisdaten von Lernobjekt-Scanner, Annotations-Repository und Lernobjekt-Repository entgegen und speichert sie so lange, bis sie an die jeweiligen Interessenten weitergeleitet werden. Ob und wie lange Ereignisdaten für einen Nutzer im Nachrichtenpuffer gespeichert werden, ist im Nutzerprofil-Repository vermerkt.

Nutzerprofil-Repository

Das Nutzerprofil-Repository stellt einen Datenspeicher für Nutzerprofile sowie Dienste zum Zugriff auf die Nutzerprofile und zur Auswertung der Nutzerprofile bereit. Das Nutzerprofil-Repository hält insbesondere Einstellungen der Nutzer vor, wie sie über Änderungen informiert werden wollen, also ob sie mittels aktiver oder passiver Benachrichtigungen informiert werden wollen und, falls sie eine aktive Benachrichtigung wünschen, über welche Kanäle (E-Mail, etc.) und in welcher Häufigkeit.

Die Daten, die im Nutzerprofil-Repository gespeichert sind, können auch durch eine umfassende Nutzerprofilverwaltung, die Nutzerprofile für mehrere Anwendungen vorhält, verwaltet werden. In diesem Fall ist es nicht nötig, dass im Notifikations-Service ein eigenes Nutzerprofil-Repository integriert wird. Statt dessen genügt es, eine Anbindung an die bereits existierende Nutzerprofilverwaltung der jeweiligen Systeme bereitzustellen.

Report-Generator

Der Report-Generator hat die Aufgabe, die Ereignisdaten, die vom Nachrichtenpuffer zum Versand freigegeben wurden, so aufzubereiten, dass sie für die Nutzer gut lesbar sind. Dies wird durch einen Template-Mechanismus realisiert, der vorformulierte Schablonen verarbeitet. Das exakte Layout (Schriftart, -farbe, etc.) hängt von der konkreten Art des Versands ab[101] und wird im Nachrichten-Publisher erledigt. Die aufbereiteten Nachrichten werden dann dem Nachrichten-Publisher zum Versand übergeben.

Nachrichten-Publisher

Der Nachrichten-Publisher hat die Aufgabe, Benachrichtigungen an die fraglichen Nutzer und Designer entsprechend der Art der Benachrichtigung zu formatieren (basierend auf Style Sheets) und dann weiterzuleiten. Der Nachrichten-Publisher erlaubt mehrere Arten der Benachrichtigung, beispielsweise mittels E-Mail, Instant Messages, etc. Die Auswahl, auf welche Art der Nutzer informiert werden soll, wird entsprechend seines Nutzerprofils vorgenommen; ebenso werden auch die jeweiligen Kontaktdaten des Nutzers aus seinem Nutzerprofil ermittelt. Hierzu wird eine Anfrage an das Nutzerprofil-Repository durchgeführt.

Dienste des Notifikations-Service

Der Notifikations-Service stellt folgende Dienste zur Verfügung (vgl. Tabelle 6-6):

Parameter und Ergebnis	**Beschreibung**
void notify(*Nachricht)*	
Nachricht: Eine Nachricht, die Informationen über ein Ereignis vermittelt	Meldet dem Notifikations-Service ein Ereignis in Form einer Nachricht. Dieser Dienst wird vom Lernobjekt-Scanner, vom Annotations-Repository und vom Lernobjekt-Repository aufgerufen.
*AnID*s getAnnotationIDs(*ObjID, UserID*)	
ObjID: Identifikator des Lernobjekts, dessen Annotation potenziell inkonsistent ist *UserID*: Kennung des anfordernden Nutzers *AnIDs*: Menge von Identifikatoren von potenziell inkonsistenten Annotationen	Liefert die Menge der im Nachrichtenpuffer gespeicherten Identifikatoren der Annotationen von Lernobjekt *ObjID*, die neu oder potenziell inkonsistent und für Nutzer *UserID* sichtbar sind. Dieser Dienst wird von Authoring-Tools und Lernsystemen genutzt, mit denen die Lernobjekte erstellt, geändert, verwaltet, dargestellt, etc. werden. Die Lernobjekte können in den Authoring-Tools dann besonders markiert werden (passive Notifikation).

Tabelle 6-6: Dienste des Notifikations-Service

[101] Nachrichten, die an Instant Messenger geschickt werden, müssen anderen Rahmenbedingungen hinsichtlich Textlänge, Formatierung, etc. genügen als Nachrichten, die als E-Mail den Interessenten zugestellt werden. Diese Abhängigkeiten können so aufgelöst werden, dass diese Anpassungen erst in der Komponente vorgenommen werden, in der die Art des Versands (Instant Message, E-Mail, etc.) bereits feststeht.

6.3.7 Zusammenspiel der Komponenten anhand gängiger Use Cases

Um das Zusammenspiel der einzelnen Komponenten zu verdeutlichen, werden im folgenden häufig durchgeführte Anwendungsszenarien (Use Cases) beschrieben. Ziel dieses Abschnitts ist es, nach der Vorstellung der einzelnen Komponenten das Verständnis für ihr gemeinsames Wirken zu vertiefen. Die Abbildungen in diesem Abschnitt sollen ein intuitives Verständnis für die Abläufe schaffen; eine formalere Darstellungsform (Sequenzdiagramme in UML) ist in Anhang E zu finden.

6.3.7.1 Verfassen einer neuen Annotation zu einem Lernobjekt (Designer)

In der Ausgangssituation arbeitet der Designer mit einem Authoring-Tool an einem Lernobjekt *L*. Bereits verfügbare Annotationen an dem Lernobjekt werden in einem Annotations-Viewer (als eigenes Tool oder integriert in das Authoring-Tool) dargestellt. Wenn ein Designer – Atomdesigner, Moduldesigner oder Kursdesigner – ein Lernobjekt annotiert, werden folgende Schritte durchgeführt (Abbildung 6-7):

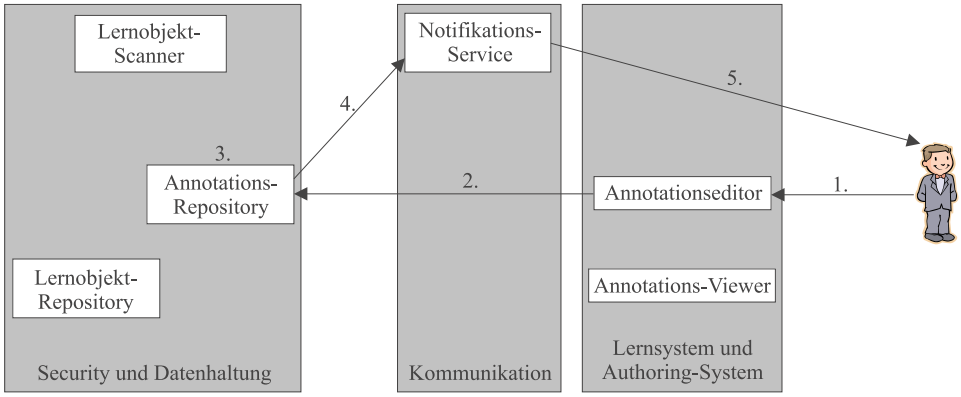


Abbildung 6-7: Verfassen einer neuen Annotation zu einem Lernobjekt

Schritt 1: Der Designer verfasst die Annotation zu einem Lernobjekt *L* im Annotationseditor (als eigenes Tool oder integriert in das Authoring-Tool).

Schritt 2: Der Annotationseditor überträgt die Annotation zusammen mit dem Identifikator des gerade bearbeiteten Lernobjekts *L* in das Annotations-Repository, das sich auf einem wohlbekannten Rechner befindet. Hierzu nutzt der Annotationseditor den Dienst putAnnotation() bzw. putAnnotationText() des Annotations-Repositories.

Schritt 3: Das Annotations-Repository erzeugt eine Assoziation zwischen Lernobjekt *L* und der Annotation und speichert die Annotation sowie die Assoziation im internen Datenspeicher.

Schritt 4: Das Annotations-Repository schickt eine Benachrichtigung an den Notifikations-Service, dass eine neue Annotation vorliegt.

Schritt 5: Der Notifikations-Service informiert alle potenziellen Interessenten[102], dass eine neue Annotation zu Lernobjekt *L* vorliegt.

6.3.7.2 Verweis auf ein bereits bestehendes Lernobjekt als Annotation zu einem Lernobjekt (Designer)

In der Ausgangssituation arbeitet der Designer mit einem Authoring-Tool an einem Lernobjekt *L*. Bereits verfügbare Annotationen an dem Lernobjekt werden in einem Annotations-

[102] Interessenten bei neuen Annotationen zu einem Lernobjekt *L* sind beispielsweise die Autoren von Lernobjekt *L* sowie alle Verfasser von bereits existierenden Annotationen zu Lernobjekt *L*, aber (abhängig von den Zugriffsrechten der Annotation) auch alle Lerner, die Lernobjekt *L* nutzen.

Viewer (als eigenes Tool oder integriert in das Authoring-Tool) dargestellt. Der Designer will nun ein bestehendes Lernobjekt als Annotation für Lernobjekt *L* verwenden. Hierbei werden folgende Schritte durchgeführt (Abbildung 6-8):

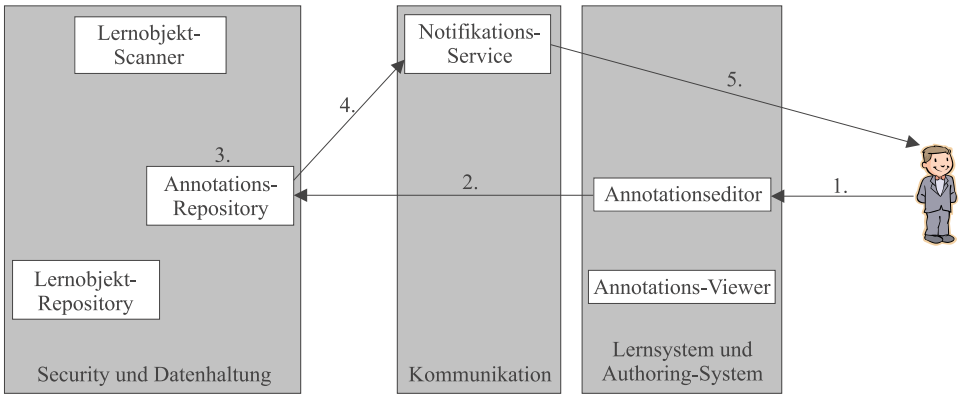


Abbildung 6-8: Nutzung eines bestehenden Lernobjekts als Annotation zu einem anderen Lernobjekt

Schritt 1: Der Designer markiert im Annotationseditor ein bereits bestehendes Lernobjekt als Annotation. Dies kann auch im Authoring-Tool geschehen – je nach Realisierung des Annotationseditors bzw. des Authoring-Tools.

Schritt 2: Der Annotationseditor überträgt den Identifikator des markierten Lernobjekts zusammen mit dem Identifikator des gerade bearbeiteten Lernobjekts *L* in das Annotations-Repository, das sich auf einem wohlbekannten Rechner befindet. Hierzu nutzt der Annotationseditor den Dienst putAnnotation() des Annotations-Repositories.

Schritt 3: Das Annotations-Repository erzeugt eine Assoziation zwischen Lernobjekt *L* und dem markierten Lernobjekt und speichert die Assoziation im internen Datenspeicher.

Schritt 4: Das Annotations-Repository schickt eine Benachrichtigung an den Notifikations-Service, dass eine neue Annotation vorliegt.

Schritt 5: Der Notifikations-Service informiert alle potenziellen Interessenten, dass eine neue Annotation zu Lernobjekt *L* vorliegt.

6.3.7.3 Verfassen einer neuen Annotation zu präsentationsfertigen Lehrmaterialien (Kurs-Presenter, Kurstutoren und Lerner)

In der Ausgangssituation arbeiten die Kurs-Presenter, Kurstutoren oder Lerner mit bereits präsentationsfertigen Lehrmaterialien, beispielsweise als vorlesungsbegleitendes Skript in PDF. Um diese Lehrmaterialien nutzen zu können, verwenden sie einen entsprechenden Viewer („Lehrmaterial-Viewer“). Daneben verwenden sie einen Annotationseditor, der entweder als eigenständiges Tool realisiert oder in den Lehrmaterial-Viewer integriert ist. Folgende Schritte werden durchgeführt (Abbildung 6-9):

Schritt 1: Der Nutzer markiert die zu annotierende Stelle in den präsentationsfertigen Lehrmaterialien.

Schritt 2: Der Nutzer verfasst eine Annotation im Annotationseditor.

Schritt 3: Der Annotationseditor identifiziert die markierte Stelle in den Lehrmaterialien mittels Informationen, die der Lehrmaterial-Viewer über Schnittstellen nach außen anbietet, und unter Verwendung geeigneter Algorithmen. Hierbei werden die Lernobjekte L_1, ..., L_n identifiziert, die der Nutzer markiert hat.

Schritt 4: Der Annotationseditor überträgt die Annotation zusammen mit den Identifikatoren der Lernobjekte L_1, ..., L_n in das Annotations-Repository, das sich auf einem wohlbekannten Rechner befindet. Hierzu nutzt der Annotationseditor für jedes L_i den Dienst putAnnotation-Text() des Annotations-Repositories.

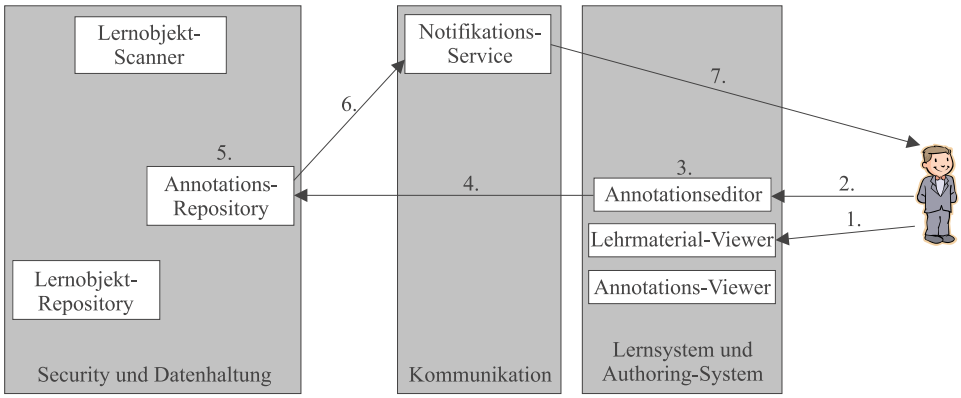


Abbildung 6-9: Verfassen einer neuen Annotation zu präsentationsfertigen Lehrmaterialien

Schritt 5: Das Annotations-Repository erzeugt für jedes L_i eine Assoziation zwischen Lernobjekt L_i und der Annotation und speichert die Annotation sowie die Assoziationen im internen Datenspeicher.

Schritt 6: Das Annotations-Repository schickt für jedes L_i eine Benachrichtigung an den Notifikations-Service, dass eine neue Annotation vorliegt.

Schritt 7: Der Notifikations-Service informiert alle potenziellen Interessenten, dass eine neue Annotation zu den Lernobjekten L_1, ..., L_n vorliegt.

6.3.7.4 Verfassen einer neuen Annotation zu einer bereits bestehenden Annotation

In der Ausgangssituation will ein Designer oder Nutzer eine bereits bestehende Annotation *A* annotieren, die in einem Annotations-Viewer, Lehrmaterial-Viewer oder in einem Authoring-Tool dargestellt ist. Hierzu nutzt er wieder einen Annotationseditor, der als eigenständiges Tool realisiert oder in den Annotations-Viewer integriert sein kann. Wenn der Designer oder Nutzer seine neue Annotation zur bereits bestehenden Annotation *A* erstellt hat, werden folgende Schritte durchgeführt (vgl. Abbildung 6-10):

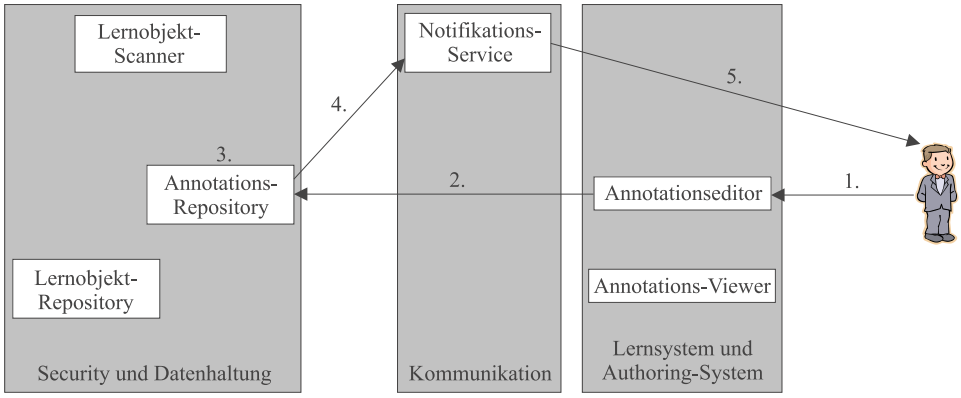


Abbildung 6-10: Verfassen einer neuen Annotation zu einer bereits bestehenden Annotation

Schritt 1: Der Designer verfasst die Annotation im Annotationseditor (als eigenes Tool oder integriert in den Annotations-Viewer).

Schritt 2: Der Annotationseditor überträgt die Annotation zusammen mit dem Identifikator der gerade bearbeiteten Annotation *A* in das Annotations-Repository, das sich auf einem wohlbekannten Rechner befindet. Hierzu nutzt der Annotationseditor den Dienst putAnnotation() bzw. putAnnotationText() des Annotations-Repositories.

Schritt 3: Das Annotations-Repository erzeugt eine Assoziation zwischen der bereits bestehenden Annotation *A* und der neuen Annotation und speichert die neue Annotation sowie die Assoziation im internen Datenspeicher.

Schritt 4: Das Annotations-Repository schickt eine Benachrichtigung an den Notifikations-Service, dass eine neue Annotation vorliegt.

Schritt 5: Der Notifikations-Service informiert alle potenziellen Interessenten, dass eine neue Annotation zu Annotation *A* vorliegt.

6.3.7.5 Überarbeiten eines annotierten Lernobjekts

Ausgangssituation ist ein Designer, der im Authoring-Tool ein bereits annotiertes Lernobjekt *L* überarbeitet. Solange der Designer Lernobjekt *L* nicht wieder ins Lernobjekt-Repository überführt und damit öffentlich zugänglich macht, werden keine Aktionen durchgeführt; die Änderungen an *L* bleiben lokal. Sobald jedoch Lernobjekt *L* ins Lernobjekt-Repository eingespielt werden soll, werden folgende Schritte durchgeführt (vgl. Abbildung 6-11):

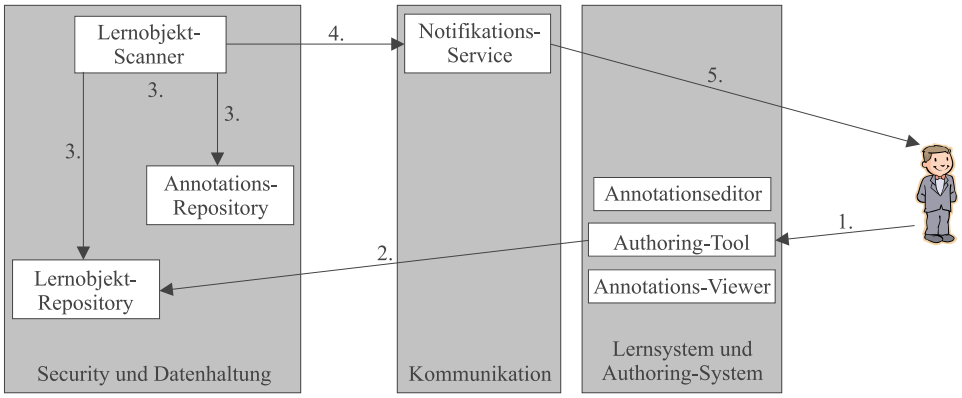


Abbildung 6-11: Überarbeiten eines annotierten Lernobjekts

Schritt 1: Der Designer überarbeitet Lernobjekt *L* mit einem Authoring-Tool.

Schritt 2: Das Authoring-System überträgt Lernobjekt *L* ins Lernobjekt-Repository. Damit liegt nun im Lernobjekt-Repository eine Version von *L* mit aktuellem Zeitstempel vor. Dieser Zeitstempel ist aktueller als der Zeitstempel der Annotation *A*, die zur alten Version von Lernobjekt *L* verfasst wurde.

Schritt 3: Zu einem späteren, nicht durch die Designer oder Nutzer bestimmbaren Zeitpunkt nimmt der Lernobjekt-Scanner eine Überprüfung der Konsistenz aller Lernobjekte und Annotationen vor. Dabei identifiziert er die Assoziation zwischen Annotation *A* und Lernobjekt *L* als inkonsistent, da der Zeitstempel von *L* neuer ist als der von *A*.

Schritt 4: Der Lernobjekt-Scanner schickt eine Benachrichtigung an den Notifikations-Service, dass Lernobjekt *L* und Annotation *A* möglicherweise inkonsistent sind.

Schritt 5: Der Notifikations-Service informiert alle potenziellen Interessenten darüber, dass Annotation *A* und Lernobjekt *L* möglicherweise inkonsistent sind.

6.3.7.6 Entfernen eines annotierten Lernobjekts aus einem Kurs

In der Ausgangssituation existiert zu einem Lernobjekt *L* eine Annotation *A*, die im Zusammenhang mit Kurs *K* verfasst wurde. Der Kursdesigner oder im Fall eines customized Kurses der Kurs-Presenter oder der Lerner will nun Lernobjekt *L* aus Kurs *K* entfernen. Hierzu nutzt er ein Authoring-Tool, um den Kurs zu bearbeiten. Soll der so modifizierte Kurs in das Lernobjekt-Repository zurückgeschrieben werden, werden folgende Schritte durchgeführt (vgl. Abbildung 6-12):

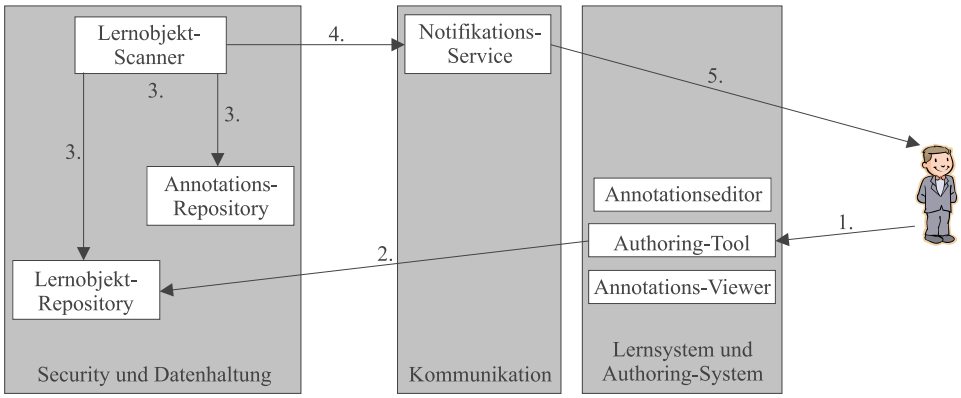


Abbildung 6-12: Entfernen eines annotierten Lernobjekts aus einem Kurs

Schritt 1: Der Kursdesigner, Kurs-Presenter oder Lerner entfernt Lernobjekt *L* aus dem Kurs, indem die Kompositionen und Assoziationen, über die Lernobjekt *L* mit anderen Lernobjekten des Kurses *K* verbunden ist, entfernt werden. Dies geschieht unter Verwendung eines Authoring-Tools. Lernobjekt *L* ist dann nicht mehr Subelement von Kurs *K*.

Schritt 2: Das Authoring-System überträgt Kurs *K* ins Lernobjekt-Repository. Damit ist nun auch im Lernobjekt-Repository vermerkt, dass Lernobjekt *L* nicht mehr Subelement von Kurs *K* ist.

Schritt 3: Zu einem späteren, nicht durch die Designer oder Nutzer bestimmbaren Zeitpunkt nimmt der Lernobjekt-Scanner eine Überprüfung der Konsistenz aller Lernobjekte und Annotationen vor. Dabei bemerkt er, dass zu Lernobjekt *L* eine Annotation *A* existiert, die im Zusammenhang mit Kurs *K* verfasst wurde, aber *L* nun nicht mehr in *K* enthalten ist. Es liegt also möglicherweise eine Inkonsistenz vor.

Schritt 4: Der Lernobjekt-Scanner schickt eine Benachrichtigung an den Notifikations-Service, dass Lernobjekt *L* und Annotation *A* möglicherweise inkonsistent sind.

Schritt 5: Der Notifikations-Service informiert alle potenziellen Interessenten darüber, dass Annotation *A* und Lernobjekt *L* möglicherweise inkonsistent sind.

6.3.8 Einsatz der Annotationsinfrastruktur für Lernplattformen

6.3.8.1 Voraussetzungen für den Einsatz

Die in diesem Kapitel beschriebene Infrastruktur muss sich potenziell in jede denkbare Lernplattform integrieren lassen. Hierzu muss jedoch die Lernplattform über gewisse Eigenschaften verfügen.

Die Lernplattform muss Lehrmaterialien unterstützen, die modular (also aus Lernobjekten) aufgebaut sind. Dies ist eine wesentliche Voraussetzung dafür, dass überhaupt eine Wiederverwendung der Lehrmaterialien stattfinden kann, denn monolithische Lehrmaterialien sind üblicherweise in sich abgeschlossen und bieten von daher bereits nur eine sehr stark eingeschränkte Wiederverwendbarkeit. Prinzipiell kann die in diesem Kapitel vorgestellte Annota-

tionsinfrastruktur zwar schon für monolithische Lehrmaterialien eingesetzt werden, aber dann liegt der Schwerpunkt ihres Einsatzes weniger auf der Wiederverwendung dieser Lehrmaterialien, sondern eher auf der Erhöhung ihrer Qualität.

Eine weitere zentrale Forderung an die Lernplattformen ist eine gewisse Offenheit. Die Annotationsinfrastruktur muss, um funktionieren zu können, mit bestimmten Daten versorgt werden. So muss beispielsweise sichergestellt sein, dass für jedes Lernobjekt eine Beschreibung über den Lernobjekt-Service verfügbar ist. Diese Beschreibung kann nur direkt von den Designern oder aus der Lernplattform kommen. Des weiteren muss für jedes Lernobjekt über den Lernobjekt-Service bekannt sein, wo (über welche URL) es abgerufen werden kann. Diese Forderungen setzen voraus, dass die Lernplattform, in die sich die Annotationsinfrastruktur integrieren muss, über (offiziell zugängliche und dokumentierte) Schnittstellen verfügt, über die die Annotationsinfrastruktur entweder aktiv von der Lernplattform mit den nötigen Informationen versorgt wird (Push-Prinzip) oder über die sie selbst die benötigten Informationen abgreifen kann (Pull-Prinzip). Die Bereitstellung von Schnittstellen ist hierbei insbesondere für das Lernobjekt-Repository von Bedeutung.

6.3.8.2 Import bereits existierender Lernobjekte

Üblicherweise sind bereits Lernobjekte vorhanden, wenn die Annotationsinfrastruktur eingesetzt wird. Auch diese Lernobjekte sollen über die Annotationsinfrastruktur annotiert werden können. Hierzu ist es notwendig, dass die Lernobjekte der Annotationsinfrastruktur bekannt sind.

Dabei genügt es, dass auf das Lernobjekt über ein Lernobjekt-Repository zugegriffen werden kann und dass das Lernobjekt dem Lernobjekt-Service bekannt ist. Angenommen, ein Moduldesigner will ein Modul über die Annotationsinfrastruktur annotierbar machen, und er hat bereits eine Verzeichnisstruktur auf seiner Festplatte, die als Lernobjekt-Repository auch von anderen Rechnern aus nutzbar ist. Dann genügt es, wenn der Moduldesigner das Lernobjekt in die Verzeichnisstruktur kopiert (beispielsweise mittels Drag and Drop in Microsoft Windows) und folgende Informationen an den Lernobjekt-Service übermittelt: Die Adresse, unter der das Lernobjekt in der Verzeichnisstruktur zugreifbar ist, und ein Parameterblock, der das Lernobjekt möglichst genau beschreibt. Ein Datentransfer des Lernobjekts in einen eigenen Datenspeicher der Annotationsinfrastruktur ist nicht nötig.

6.3.8.3 Versionierung von Lernobjekten

Lernplattformen bieten oft auch ein Versionskontrollsystem für die von ihnen verwalteten Lernobjekte an. Die Annotationsinfrastruktur muss daher auch mit versionierten Lernobjekten umgehen können. Da die in dieser Arbeit beschriebene Annotationsinfrastruktur keine eigene Lernobjektverwaltung durchführt, sondern auf bereits bestehenden Lernobjektverwaltungssystemen, die von den Lernplattformen geliefert werden, mittels der Lernobjektzugriffsschicht aufsetzt (vgl. 6.3.2.3), verfügt sie auch nicht über ein eigenes Versionskontrollsystem.

Versionen eines Lernobjekts werden von der Annotationsinfrastruktur wie getrennte Lernobjekte behandelt; sie haben komplett getrennte Parameter für ihre Beschreibung. Über Parameter von Lernobjekten lassen sich jedoch Versionsnummern sowie Verweise auf Vorgängerversionen von Lernobjekten realisieren, so dass sich auf diese Weise ein Versionsbaum nachbilden lässt.

Wird ein Lernobjekt *L* überarbeitet, so können folgende Fälle eintreten:

- Der alte Inhalt von Lernobjekt *L* wird vom neuen Inhalt überschrieben; es wird kein neues Lernobjekt erzeugt und es wird lediglich die Versionsnummer in den Parametern des Lernobjekts hochgezählt.

In diesem Fall ist die alte Version von *L* nicht mehr verfügbar, da sie von der neuen Version ersetzt wurde. Andere Lernobjekte, die auf Lernobjekt *L* zugreifen, sehen unmittelbar den neuen Inhalt.

- Es wird ein neues Lernobjekt *L'* angelegt, das den überarbeiteten Inhalt enthält. Die Parameter von *L* werden für *L'* kopiert; lediglich die Versionsnummer von *L'* wird angehoben.

  In diesem Fall ist die alte Version, Lernobjekt *L*, weiterhin verfügbar. Andere Lernobjekte, die *L* nutzen, sehen weiterhin den alten Stand. Erst wenn statt Lernobjekt *L* Lernobjekt *L'* genutzt wird, wird der neue Inhalt genutzt.

6.4 Erzeugung präsentationsfertiger Kurse

Alle Kurse liegen intern als Baum von Modulen und Atomen vor: Der Kurs ist die Wurzel des Baums, die Module sind die inneren Knoten und die Atome sind die Blätter des Baums. Bevor nun ein Kurs von den Nutzern für Lehrveranstaltungen oder für das Selbststudium verwendet werden kann, muss er in das endgültige Präsentationsformat überführt werden. Bei dieser Anpassung erzeugt man aus einem Kurs eine Kursvariante, vgl. 3.5.3, die an die Zielgruppe und die Art der geplanten Nutzung in der Lehre angepasst ist. Der Kurs selbst stellt hierbei das „Rohmaterial" dar, der erst in Form einer entsprechend angepassten Kursvariante konkret nutzbar ist. Hierbei sind einige Aufgaben durchzuführen, die im folgenden beschrieben werden.

6.4.1 Erstellung von Kursvarianten

Bei diesen Aufgaben spielen die Parameter, die in den Nutzungsklassen „Lehre", „Präsentation", etc. (vgl. 3.5.2) definiert wurden, eine entscheidende Rolle: Sie definieren die endgültige Gestalt der präsentationsfertigen Kursvariante. Die Nutzungsklassen werden durch den Kursdesigner oder – im Falle eines adaptierbaren bzw. customized Kurses – vom Kurs-Presenter festgelegt, wenn er die präsentationsfähige Fassung des Kurses (die Kursvariante) erstellt: Er legt fest, für welche Zielgruppe der Kurs vorbereitet wird und damit welchen Schwierigkeitsgrad er maximal haben darf, und ob die Kursvariante vorlesungsbegleitend oder als Skript eingesetzt wird und damit welcher Detaillierungsgrad sinnvoll ist. Des weiteren legt der Kurs-Presenter fest, in welchem Ausgabeformat (HTML, PDF, Postscript, etc.) die Kursvariante zu erzeugen ist. Diese Wahl bestimmt unter anderem, ob eine Sequenzialisierung der Module und Atome vorgenommen werden muss.

Schritt 1: Filterung der dargestellten Inhalte

Das Ausgangsmaterial für einen Kurs (der oben beschriebene Kursbaum) kann erheblich umfangreicher sein als es für die Zielgruppe, für die eine Kursvariante erstellt werden soll, sinnvoll ist. Der Kursdesigner bei der Entwicklung eines Kurses bzw. der Kurs-Presenter im Falle eines adaptierbaren bzw. customized Kurses erhält nun die Möglichkeit, den Kurs für die jeweilige Zielgruppe quasi zurechtzuschneiden und überflüssige oder zu schwierige Themen auszublenden.

Hier kommen die Parameter der Nutzungsklasse „Lehre" ins Spiel: Über sie wird unter anderem die Zielgruppe festgelegt. Der Kursbaum wird nun in einer Breitensuche durchkämmt und jeder Knoten wird anhand der Parameter der Nutzungsklasse „Lehre" und eventuell definierter Constraints daraufhin überprüft, ob er darzustellen ist. Hierzu wird der Schwierigkeitsgrad des Knotens mit dem in den Parametern der Nutzungsklasse „Lehre" abgelegten maximalen Schwierigkeitsgrad abgeglichen. Die nicht darzustellenden Knoten werden als unsichtbar markiert und in den folgenden Schritten nicht weiter berücksichtigt.

Als Ergebnis dieses Schritts liegt ein der Zielgruppe entsprechend zugeschnittener Kursbaum vor, dessen Module und Atome höchstens so schwer zu verstehen sind, wie es für die Zielgruppe entsprechend der Parametrisierung angemessen ist.

Schritt 2: Festlegung des Detaillierungs- und Abstraktionsgrads der dargestellten Inhalte

Je nach Lehrform sind unterschiedliche Detaillierungsgrade der darzustellenden Module und Atome nötig. Präsentationsunterlagen, die während einer Vorlesung mit einem Beamer auf eine Leinwand projiziert werden, sollten nur wenig Text beinhalten und die wesentlichen Punkte in kurzen, einfachen und einprägsamen Aussagen darstellen. Vorlesungsbegleitende Skripten hingegen sollten deutlich ausführlicher sein.

Anhand der Parameter der Nutzungsklasse „Lehre“ kann der Detaillierungsgrad des Kurses gesteuert werden. Ebenso müssen hier die technischen Rahmenbedingungen berücksichtigt werden, denn diese legen die Bedingungen fest, unter denen der Kurs später genutzt wird. Die Submodule und Atome eines Moduls sind üblicherweise nicht fest verdrahtet, sondern über Platzhalter, die mit Parametern versehen sind, spezifiziert. In jeden Platzhalter können potenziell mehrere Submodule oder Atome eingesetzt werden, die zu den Parametern des Platzhalters passen müssen. Mittels der Parameter der Nutzungsklasse „Lehre“ werden zusätzlich zu den Parametern eines Platzhalters noch weitere Bedingungen an die Submodule oder Atome, die in einen Platzhalter eingefügt werden sollen, gestellt. Dies betrifft unter anderem den Detaillierungsgrad der Submodule oder Atome.

Als Ergebnis dieses Schritts liegt der Kursbaum in einer der geplanten Nutzung (lehrveranstaltungsbegleitende Präsentation, Skript, etc.) angepassten Parametrisierung vor.

Schritt 3: Festlegung der Reihenfolge der Lernobjekte

Für bestimmte Präsentationsformate ist es nötig, die Reihenfolge der Submodule und Atome eines Moduls festzulegen. Grundlage sind hier die Kompositionsbeziehungen zwischen dem Modul und seinen Submodulen und Atomen: Die Kompositionsbeziehungen sind mit einem Parameter „kind“ versehen, der die Semantik des Submoduls oder Atoms innerhalb des übergeordneten Moduls wiedergibt.

Jedes Modul ist (mindestens) einer Kategorie zugewiesen, vgl. 3.2.1.3 Jede Kategorie legt ihrerseits fest, aus welchen semantischen Einheiten das Modul besteht und wie die einzelnen semantischen Einheiten angeordnet sind. Die Kategorie, der ein Modul zugeordnet ist, liefert so die Reihenfolge der Submodule und Atome des Moduls.

Angenommen, ein Modul besteht aus den Submodulen *A* (Kompositionstyp „statement“), *B* (Kompositionstyp „example“), *C* (Kompositionstyp „statement“) und *D* (Kompositionstyp „definition“). Weiterhin sei angenommen, das Modul ist der Kategorie „standard-kapitel“ zugeordnet, wobei die Kategorie „standard-kapitel“ durch den regulären Ausdruck „definition, statement+, example*“ definiert ist. Die Submodule werden entsprechend der Kategorie in folgende Reihenfolge gebracht: *D, A, C, B.*

Der Kursbaum, der als Ergebnis von Schritt 2 vorliegt, wird nun wieder in einer Breitensuche durchlaufen und für jedes Modul *M* werden dessen Submodule entsprechend der Kategorie von *M* in ihrer Reihenfolge numeriert.

Als Ergebnis des Schritts erhält man einen Kursbaum, in dem alle Module und Atome in der korrekten Reihenfolge gemäß den jeweiligen Kategorien der Module angeordnet sind. Man beachte, dass zu diesem Zeitpunkt noch nicht feststeht, ob die Submodule oder Atome mittels Hyperlinks (oder vergleichbaren Mechanismen) mit dem Modul verbunden werden oder ob die Submodule oder Atome in das Modul eingebettet werden („inline“).

Schritt 4: Integration von Annotationen

An dieser Stelle werden die Annotationen in den Kursbaum integriert. Der Kursbaum wird hierzu durchlaufen und für jeden sichtbaren Knoten (Modul oder Atom) des Kursbaums wird überprüft, ob er annotiert wurde. Ist für einen Knoten eine Annotation verfügbar, die nicht öffentlich sichtbar ist, so wird die Annotation als unsichtbar markiert[103].

Als Ergebnis erhält man einen Kursbaum, in dem alle Annotationen, die nicht öffentlich sichtbar sein dürfen, als unsichtbar markiert sind. Annotationen zu Modulen oder Atomen, die nicht als unsichtbar markiert sind, werden im nächsten Schritt in die endgültige Präsentationsform des Kurses mit übernommen.

Schritt 5: Festlegung des Präsentationsformats

Im letzten Schritt wird schließlich der Kursbaum in ein konkretes Dokument umgewandelt. Maßgeblich sind hier wieder die Stylesheets, die unter anderem auch das Präsentationsformat festlegen.

Je nach Präsentationsformat werden Submodule und Atome eines Moduls als Hyperlinks oder „inline“ eingebunden – bei Präsentationsformaten mit Graphstruktur (beispielsweise Hypertexte in HTML) werden Hyperlinks eingesetzt, bei Präsentationsformaten mit sequenzieller Struktur (beispielsweise Texte in Postscript oder PDF) findet eine Sequenzialisierung der Submodule und Atome Anwendung. Mit Annotationen wird analog verfahren; auch hier wird über das Präsentationsformat gesteuert, auf welche Weise sie eingebunden werden sollen.

Der Kursbaum wird nun in einer Tiefensuche durchlaufen. Jedes Modul, jedes Atom und jede Annotation wird mittels Style Sheets in das gewünschte Ausgabeformat umgewandelt und – je nach Präsentationsformat – entweder als eigenständiges Dokument gespeichert und mittels Hyperlinks angebunden oder aber unmittelbar an die bisherige Ausgabe angefügt. Wird beispielsweise als Präsentationsformat HTML gewünscht, so werden die einzelnen Atome, Module und Annotationen als HTML-Dateien abgelegt und mittels Hyperlinks miteinander verbunden. Wird als Präsentationsformat beispielsweise PDF angegeben, so werden die einzelnen Module und Atome aneinandergehängt und anschließend nach PDF konvertiert; Annotationen werden als PDF-Annotationen umgesetzt. Dieses Vorgehen ist dem sehr ähnlich, das in Targeteam (vgl. beispielsweise [TARGETEAM]) angewandt wird.

Wichtig ist hierbei, dass die Style Sheets lediglich das Aussehen der Lernobjekte bestimmen, nicht jedoch ihre Reihenfolge. Die Reihenfolge der Lernobjekte wird bereits in Schritt 3 vorgenommen. Ebenso wird das Ausblenden einzelner Module und Atome nicht durch Style Sheets vorgenommen, sondern dies wird in Schritten 1 und 2 erledigt. Dies ist wichtig, da für die spätere Nutzung – beispielsweise bei Annotationen – die Baumstruktur des präsentationsfertigen Kurses vorgehalten werden muss.

Als Ergebnis dieses Schritts erhält man den präsentationsfähigen Kurs. Dieser kann nun in den Lehrveranstaltungen oder zum Selbststudium genutzt werden. Die Baumstruktur, die diesem Kurs zugrunde liegt, wird im Lernobjekt-Repository vorgehalten, falls sie für spätere Arbeiten benötigt wird. Sie kann über den Identifikator des Kurses beim Lernobjekt-Service lokalisiert und dann mittels des Dienstes get() bei der entsprechenden Lernobjektzugriffsschicht angefordert werden.

[103] Der Kurs wird für die Allgemeinheit erstellt. Dies bedeutet, dass er keine privaten Informationen und Annotationen – also insbesondere auch keine privaten Annotationen oder Gruppenannotationen – im endgültigen Präsentationsformat enthalten darf.

6.4.2 Bereitstellung von Kursvarianten „on the fly“

Bei dem gerade beschriebenen Verfahren werden Annotationen gleich in das Ergebnis – eine zur Lehrveranstaltung und zur Zielgruppe passende Kursvariante – integriert. Werden Annotationen nach der Erstellung der Kursvariante verfasst, so werden sie in der Kursvariante nicht mehr berücksichtigt.

Dieses Problem kann durch dynamische Erstellung der Lehrmaterialien gemildert werden: Der Kurs-Presenter legt die Parameter der Nutzungsklassen „Lehre“ und „Präsentation“ einmal fest und führt dann die Schritte 1 bis 3 durch. So entstehen halb fertige Kursvarianten, in die aber noch keine Annotationen integriert sind. Diese Kursvarianten werden für Anfragen der Nutzer nach den präsentationsfertigen Lehrmaterialien vorgehalten.

Fordert nun ein Nutzer die präsentationsfähigen Lehrmaterialien an, so werden auf der Grundlage der vorgehaltenen halbfertigen Kursvarianten und den aktuell verfassten Annotationen die Schritte 4 und 5 durchgeführt und damit die endgültigen präsentationsfertigen Lehrmaterialien mit den aktuellen Annotationen erstellt. Diese können dann an die Anforderer der Lehrmaterialien ausgegeben werden. Wird beispielsweise ein Kurs online als HTML-Skript angeboten, so können bei jedem Aufruf einer HTML-Seite die Schritte 4 und 5 durchgeführt und so alle aktuell vorhandenen Annotationen integriert werden. Werden hingegen Lehrmaterialien einmal generiert und dann in dieser Form über einen längeren Zeitraum genutzt, beispielsweise bei PDF-Skripten, die ausgedruckt zum Selbststudium verwendet werden, so können natürlich nur die Annotationen integriert werden, die zum Zeitpunkt der Generierung der Lehrmaterialien vorlagen. Welche Annotationen also in die Lehrmaterialien integriert werden, hängt vom Zeitpunkt der Generierung der Lehrmaterialien ab: Es können immer nur die Annotationen berücksichtigt werden, die zum Zeitpunkt der Generierung der Lehrmaterialien im System gespeichert waren; später hinzukommende Annotationen können erst wieder in späteren Generierungen berücksichtigt werden.

6.5 Zusammenfassung

In diesem Kapitel wurde eine Infrastruktur vorgestellt, die den Prozess der Lehrmaterialerstellung möglichst gut mittels Annotationen unterstützen soll. Die Unterstützung liegt hierbei im wesentlichen in der Ermöglichung einer Kommunikation und damit einer Abstimmung zwischen den Prozesspartnern, und in der Folge in der Koordination des ansonsten sehr unübersichtlich und anonym verlaufenden Lehrmaterialerstellungs- und -nutzungsprozesses. Ferner tragen die Annotationen je nach Inhaltstyp zur Verbesserung der Qualität der Lernobjekte bei: Sie ermöglichen es, Feedback und Korrekturvorschläge in den Lehrmaterialerstellungs- und –nutzungsprozess einzubringen und erlauben so eine gezielte Verbesserung der einzelnen Lernobjekte und Kurse.

In einem ersten Schritt wurde in 6.2 geklärt, wie sich eine Annotationsinfrastruktur in die bestehenden Lernplattformen eingliedern muss, um effektiv wirken zu können. Hierbei wurde festgestellt, dass die Annotationsinfrastruktur Basisfunktionalität für sowohl Authoring-Systeme als auch Lernsysteme bietet. Zugleich nutzt sie die Querschnittsfunktionen „Kommunikation“ und „Security“, die in umfassenden Lernplattformen realisiert sind.

Auf dieser Einordnung aufbauend wurden zentrale Anforderungen an eine Annotationsinfrastruktur identifiziert: Die Eingabe, Bearbeitung und Anzeige von Annotationen muss für alle Prozessbeteiligten so einfach wie möglich sein. Des weiteren muss die Annotationsinfrastruktur generisch gestaltet sein, also für alle Lernplattformen und über alle Lehrmaterialformate hinweg verfügbar sein. In der Folge wurde eine Annotationsinfrastruktur vorgestellt, die diese Anforderungen erfüllt.

In 6.3 wurden die funktionalen Einheiten, aus denen die Annotationsinfrastruktur besteht, vorgestellt. Basis der Annotationsinfrastruktur ist die Datenhaltung: Die Datenhaltung umfasst die Speicherung der Lernobjekte in einem Lernobjekt-Repository ebenso wie die Speicherung der Annotationen in einem Annotations-Repository. Da jedes Lernobjekt annotiert werden kann, ist es nötig, einen einheitlichen Zugriff auf die Lernobjekte zu ermöglichen, und zwar unabhängig davon, wo und wie die Lernobjekte konkret realisiert sind (Dateisystem, Targeteam-Pool, etc.). Hierzu wurde eine logische Zugriffsschicht auf Lernobjekte eingeführt, die aus einer Lernobjektzugriffsschicht, die die unterschiedlichen Implementierungen der Lernobjektdatenspeicher kapselt, und einem Lernobjekt-Service, der einerseits die Ortstransparenz der Lernobjekte realisiert und andererseits eine Suche nach Lernobjekten mit bestimmten Eigenschaften erlaubt, besteht.

Da bei der Bearbeitung der Lernobjekte leicht Inkonsistenzen zwischen einem Lernobjekt und den es referenzierenden Annotationen auftreten können, wurde weiterhin ein Lernobjekt-Scanner eingeführt, der die Annotationen regelmäßig mit den Lernobjekten abgleicht und gegebenenfalls interessierte Designer und Nutzer bei Inkonsistenzen informiert. Diese Information wird nicht direkt vom Lernobjekt-Scanner durchgeführt, sondern von einer eigens für die Information der Nutzer eingeführten Komponente, dem Notifikations-Service. Bei diesem laufen alle Ereignisse, die während der Erstellung und Bearbeitung der Lernobjekte entstehen, zusammen. Der Notifikations-Service nimmt die Ereignisse entgegen, sammelt sie gegebenenfalls und leitet sie entsprechend den Vorgaben in den Nutzerprofilen an die Interessenten weiter.

Um das Verständnis für die Annotationsinfrastruktur zu vertiefen und das Zusammenspiel der einzelnen Komponenten zu verdeutlichen, wurden anschließend häufig vorkommende Use Cases vorgestellt und im Detail beschrieben.

Im folgenden wird nun betrachtet, wie das in diesem Kapitel vorgestellte Konzept in einem Prototypen umgesetzt wurde und welche Besonderheiten bei der Implementierung zu berücksichtigen waren.

7 Prototypische Realisierung

7.1 Überblick

Im letzten Kapitel wurden die Konzepte vorgestellt, die die Grundlage der Annotationsinfrastruktur bilden. Es wurde gezeigt, aus welchen Komponenten die Annotationsinfrastruktur besteht, welche Arbeiten sie jeweils verrichten und wie sie zusammenspielen, um die Funktionalität der Annotationsinfrastruktur bereitzustellen. In diesem Kapitel wird gezeigt, wie diese Komponenten im Prototypen realisiert wurden.

Im ersten Teil, Abschnitt 7.2, werden wichtige Grundlagen der Realisierung der Lernobjekte und Annotationen erläutert: Es wird vorgestellt, wie die Identifikatoren der Lernobjekte und Annotationen gestaltet sind und wie die Beschreibung der Lernobjekte und Annotationen durch Parameter realisiert ist. Es wird außerdem erläutert, wie die Verbindung zwischen einem zu annotierenden Ausschnitt eines Lernobjekts und der zugehörigen Annotation mittels Parametern von Assoziationen realisiert ist.

Da die Annotationsinfrastruktur aus mehreren, miteinander kommunizierenden Komponenten besteht, muss weiterhin geklärt werden, auf der Grundlage welcher Technologie die Kommunikation stattfinden soll. Hierauf wird in 7.3 eingegangen.

In 7.4 schließlich wird die Realisierung der einzelnen Komponenten im Detail besprochen. Dieser Abschnitt, der Hauptabschnitt dieses Kapitels, ist analog zu Abschnitt 6.3 in Kapitel 6 aufgebaut, um möglichst leicht von der Beschreibung des Konzepts, das hinter einer Komponente steht, zu der konkreten Implementierung zu finden. Im Unterschied zu den entsprechenden Abschnitten in Kapitel 6 wird hier allerdings auf die technischen Rahmenbedingungen, unter denen eine Komponente arbeiten soll, und die konkrete Umsetzung eingegangen.

7.2 Modellierung von Lernobjekten und Annotationen

Lernobjekte und Annotationen gleichermaßen bestehen aus zwei Teilen: Zum einen den Nutzdaten, die den Inhalt der Lernobjekte bzw. Annotationen ausmachen, zum anderen deren Beschreibungen durch Parameter. Im folgenden wird betrachtet, wie Lernobjekte und Annotationen in ihren Datenstrukturen modelliert werden.

Annotationen haben, wie bereits in 5.4.1.2 beschrieben, konzeptionell den gleichen Aufbau wie Lernobjekte: Sie werden als Module, die gegebenenfalls ein Atom enthalten, modelliert. Die folgenden Ausführungen gelten daher für Lernobjekte und Annotationen gleichermaßen.

7.2.1 Aufbau von Lehrmaterialien und Annotationen

7.2.1.1 Identifikatoren

Jedes Lernobjekt und jede Annotation wird durch einen weltweit eindeutigen Identifikator benannt. Eine solche Adressierung wird üblicherweise durch URIs (Universal Ressource Identifier) entsprechend ihrer Definition durch das W3C (vgl. [URI]) geleistet. Der Nachteil von URIs ist allerdings, dass gleich der Speicherort des Lernobjekts bzw. der Annotation in Form eines Rechnernamens bzw. einer IP-Adresse enthalten ist. Da ein Lernobjekt (beispielsweise aus Gründen der Verfügbarkeit) auf mehreren Datenspeichern im Lernobjekt-Repository repliziert sein kann, ist die Nennung eines Speicherortes nicht sinnvoll. Es wird daher folgendes Namensschema vorgeschlagen:

```
<userID>@<currentIP>.<currentTime>
```

<UserID> ist dabei die Kennung, mit der sich der Lernobjektautor beim System <currentIP> angemeldet hat, und <currentTime> ist der Zeitpunkt, zu dem das Lernobjekt bzw. die Annotation initial erstellt wird. <UserID> kann beispielsweise eine Unix-Kennung sein, die vom System mit verwendet wird, oder eine Kennung, die sich der Lernobjektautor beim System selbst aussuchen konnte.

Beispiel für einen Identifikator: zhuang@127.0.0.1.52354

Die Kombination dieser drei Elemente ist weltweit eindeutig, enthält aber keine Aussage darüber, wo das Lernobjekt bzw. die Annotation letztlich gespeichert ist. Die Parameter atomID und moduleID (vgl. 3.1.5.3), die Atome und Module identifizieren, werden daher im folgenden durch Identifikatoren entsprechend dieses Schemas dargestellt.

7.2.1.2 Beschreibung von Lernobjekten und Annotationen durch Parameter

Die Beschreibung der Lernobjekte und Annotationen geschieht, wie in 3.3.1.5 beschrieben, durch Parameter. Im Prototyp werden diese in XML abgelegt. Hierbei muss zwischen folgenden Arten von Parametern unterschieden werden:

- Parameter, die den Inhalt, die Struktur, die Nutzung und die Präsentation der Lernobjekte und Annotationen beschreiben.
- Parameter, die Prozessinformationen enthalten, beispielsweise der letzte Bearbeiter eines Lernobjekts oder einer Annotation
- Parameter, die Werte beinhalten, die für die technischen Abläufe der Annotationsinfrastruktur von Bedeutung sind.

Die ersten beiden Arten von Parametern wurden bereits in Kapitel 3 bzw. Kapitel 4 besprochen; die dritte Art von Parametern kommt hier neu hinzu. Im folgenden sollen nur noch die Parameter betrachtet werden, die für die technische Umsetzung der Annotationsinfrastruktur benötigt werden. Tabelle 7-1 führt diese Parameter auf; sie sind insbesondere für die Konsistenzsicherung zwischen Annotationen und den Elementen, auf die sie sich beziehen (Lernobjekte sowie Annotationen), wichtig.

Parameter	Belegung
created	Zeitstempel, der gesetzt wird, wenn das Lernobjekt bzw. die Annotation initial erstellt wird.
created-by	Identifikator des initialen Verfassers („Eigentümers“) des Elements.
last-modified	Zeitstempel, der gesetzt wird, wenn das Lernobjekt bzw. die Annotation geändert wird (Inhalt oder Parameter).

last-modified-by	Name des Designers oder Nutzers, der das Lernobjekt bzw. die Annotation (in den Parametern) zuletzt geändert hat.
access	Regelt Art des Zugriffs auf die Annotation. Kann „public“, „private“ oder „group“ sein (vgl. 5.2.3.3).
group	Identifikator der Gruppe, deren Mitglieder Zugriff auf die Annotation haben.
contact	Kontaktdaten des initialen Verfassers.

Tabelle 7-1: Attribute für die technische Umsetzung

7.2.1.3 Verbindung von Annotation und annotiertem Element

Annotationen können sich sowohl auf den Inhalt von Lernobjekten als auch auf deren Beschreibung beziehen. Dementsprechend können ganze Lernobjekte, Teile des Inhalts, aber auch Parameter und Constraints annotiert werden. Es genügt damit nicht, in der Assoziation, die die Verbindung zwischen einer Annotation und dem annotierten Element (Lernobjekt oder Annotation) realisiert, lediglich den Identifikator des annotierten Elements anzugeben. Zusätzlich muss beschrieben werden, auf welchen Teil des annotierten Elements sich die Annotation bezieht: Es muss das Problem der Adressierung des annotierten Bereichs eines Lernobjekts bzw. einer Annotation gelöst werden.

Hierzu wird der Begriff des Ankers eingeführt. Ein *Anker* bezeichnet einen Referenzpunkt. Er markiert den Teil eines Elements (Lernobjekt oder Annotation), auf den sich die Annotation bezieht. Ein Anker wird durch die beiden Parameter anchor-id und anchor-area realisiert. Mittels dieser Parameter wird der Identifikator (im Parameter anchor-id) und der Bereich des Elements (im Parameter anchor-area) spezifiziert, auf den sich eine Annotation bezieht. Die Annotation selbst, die einem Bereich oder einem Parameter zugeordnet ist, wird ebenfalls über einen Parameter festgelegt. Hierzu wird der Parameter annotation-id verwendet. Abbildung 7-1 illustriert dies.

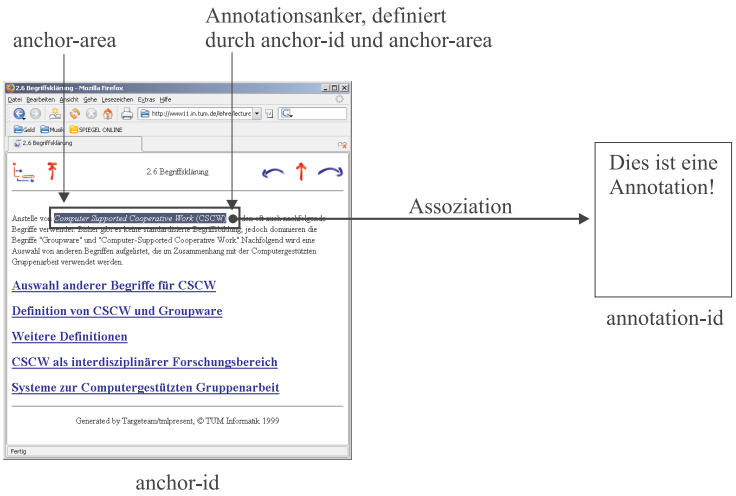


Abbildung 7-1: Adressierung von Annotationen

Je nachdem, welchen Teil eines Elements eine Annotation adressiert, sieht der Anker unterschiedlich aus:

- Die Adressierung eines kompletten Lernobjekts – inklusive dessen Beschreibung und Inhalt – bzw. einer kompletten Annotation ist einfach über seinen Identifikator (vgl. 7.2.1.1) realisierbar. Der Anker ist hier der Identifikator des annotierten Lern-

objekts bzw. der annotierten Annotation. Der Parameter anchor-area wird nicht gesetzt.

Beispiel:

anchor-id = zhuang@127.0.0.1.52354

bezeichnet den Identifikator eines Lernobjekts bzw. einer Annotation.

- Die Adressierung von Parametern oder Constraints, auf die sich eine Annotation bezieht, ist ebenfalls recht einfach zu realisieren: Als Anker dient der Identifikator des Lernobjekts, dessen Parameter oder Constraint annotiert werden soll, und der Parametername bzw. der logische Ausdruck, der die Constraint repräsentiert.

  Beispiel:

  anchor-id = zhuang@127.0.0.1.52354
  anchor-area = „created-by“

  bezeichnet den Parameter „created-by“ des Lernobjekts mit dem Identifikator “zhuang@127.0.0.1.52354”.

- Die Adressierung von bestimmten Teilen des Inhalts eines Lernobjekts ist komplizierter. Für den Parameter anchor-id wird wieder der Identifikator des zu annotierenden Elements genommen. Für den Parameter anchor-area jedoch ist man vom Medientyp des Inhalts abhängig, vgl. Tabelle 7-2, in der auch Beispiele genannt sind: Bei in XML kodierten Daten kann jedes Tag und jedes Datum innerhalb eines Tags als Anker dienen, bei Texten textuelle Strukturelemente wie Wörter, Absätze oder Kapitel. Bei Bildern und Grafiken wiederum können Bereiche durch eine passende Beschreibung des Bereichs – beispielsweise ähnlich den Image Maps von HTML – als Anker dienen, bei Videos und Animationen kommt neben der Beschreibung des Bereichs noch eine Beschreibung der zeitlichen Ausdehnung hinzu. Alternativ können bei Videos und Animationen auch Motion Tracking-Technologien [Moti], wie sie bei der digitalen Videobearbeitung zum Einsatz kommen, zur Adressierung eingesetzt werden. All diese Ausschnitte des Inhalts von Lernobjekten können im Anker enthalten sein.

Medientyp	**Beschreibung des Ausschnitts (Parameter „anchor-area“)**	**Beispiel**
XML und XHTML	Tupel bestehend aus XPath-Ausdruck[104] für Beginn und XPath-Ausdruck für Ende des annotierten Bereichs.	(kap1/subkap[position()=2], kap1/subkap[position()=3])
unstrukturierter Text	Tupel bestehend aus Position des ersten Buchstabens und Position des letzten Buchstabens des annotierten Bereichs. Ist das zweite Argument (die Position des letzten Buchstabens) 0, so ist der Ausschnitt beginnend mit der Position des ersten Buchstabens bis zum Ende des Texts gemeint. Analog wenn das erste Argument 0 ist.	(10, 25)

[104] XPath-Ausdrücke ermöglichen die exakte Darstellung eines Pfads innerhalb einer Hierarchie [XPATH].

Grafiken	Tupel bestehend aus der Position der linken oberen Ecke eines Rechtecks sowie dessen Breite und Höhe.	(20,30,160,100)
Animationen und Videos	Tupel bestehend aus der Position der linken oberen Ecke eines Rechtecks sowie dessen Breite und Höhe. Zusätzlich werden im Tupel noch zwei Zeitpunkte (in Sekunden) angegeben, die den Beginn und das Ende der annotierten Sequenz markieren.	(20,30,160,100, 360, 600)
Klänge und Musik	Tupel bestehend aus zwei Zeitpunkten (in Sekunden): Dem Beginn und dem Ende der annotierten Sequenz.	(30,45)

Tabelle 7-2: Beispiele für Beschreibungen des annotierten Bereichs (anchor-area)

Die Verbindung zwischen einem annotierten Element und seiner Annotation wird mittels einer Assoziation realisiert. Der Anker, der den annotierten Bereich anchor-area eines Elements anchor-id adressiert, sowie der Identifikator annotation-id der Annotation, die auf diesen Bereich verweist, werden in der Assoziation abgelegt. In Tabelle 7-3 sind die hier verwendeten Attribute zusammenfassend aufgeführt.

Parameter	Belegung
anchor-id	Identifikator des Elements (Lernobjekt oder Annotation), das annotiert wird
anchor-area	Beschreibung des Ausschnitts des annotierten Elements, auf den sich die Annotation bezieht. Bleibt leer, falls die Annotation das ganze Element umfasst
annotation-id	Identifikator der Annotation

Tabelle 7-3: Attribute zur Verbindung von annotiertem Element und Annotation

Ein Anker (die „Quellseite“ der Assoziation) wird durch die beiden Parameter anchor-id und anchor-area der Assoziation realisiert. Die Annotation selbst, die einem Bereich oder einem Parameter zugeordnet ist, wird ebenfalls über einen Parameter der Assoziation (die „Zielseite“ der Assoziation) festgelegt: Hierzu wird der Parameter annotation-id der Assoziation verwendet.

Im Prototyp werden nur einfache Anker betrachtet (nur anchor-id wird gesetzt, nicht hingegen anchor-area). Des weiteren werden im Prototyp derzeit nur rein textuelle Annotationen unterstützt. Dies dient dazu, den Prototypen für die Annotationsinfrastruktur einfach und überschaubar zu halten. Eine Beschränkung der Allgemeinheit des in dieser Arbeit vorgestellten Ansatzes erfolgt dadurch nicht.

7.2.2 Trennung von Inhalt und Beschreibung

Es bietet sich an, den konkreten Inhalt eines Lernobjekts bzw. einer Annotation getrennt von seiner Beschreibung zu verwalten. Die Gründe hierfür liegen einerseits in der konkreten Ausgestaltung, beispielsweise dem Medientyp des Lernobjekts bzw. der Annotation, die eine gemeinsame Speicherung von Inhalt und Beschreibung nicht immer zulässt, und andererseits in den für eine effiziente Nutzung benötigten unterschiedlichen Rechten (vgl. hierzu 6.3.2.4).

Auch eine Suche nach Lernobjekten mit bestimmten, durch Parameter beschriebenen Eigenschaften kann effizienter durchgeführt werden, wenn nicht die Lernobjekte selbst, sondern nur deren Beschreibungen durchsucht werden müssen.

Ein Lernobjekt bzw. eine Annotation kann einen beliebigen Datentyp haben – es kann eine Microsoft Powerpoint-Präsentation oder eine Menge von HTML-Seiten sein, um nur zwei Beispiele zu nennen. Die konkrete Gestalt des Lernobjekts bzw. einer Annotation sowie dessen Speicherort ist egal, solange es nur über ihren Identifikator ansprechbar ist. Die Beschreibung des Lernobjekts bzw. der Annotation hingegen ist zweckmäßigerweise in XML oder in einer Datenbank abgelegt, da diese Speicherformen gute Möglichkeiten zur Strukturierung von Daten bieten und plattformübergreifend genutzt werden kann.

Allein aus technischen Gründen (Dateiformate) ist schon eine Verquickung von Inhalt und Beschreibung nicht sinnvoll. Ein weiterer Grund ist, dass bei Änderungen in der Beschreibung bei einer Trennung von Inhalt und Beschreibung der Datenbestand, der den Inhalt enthält, nicht verändert werden muss, sondern lediglich der Datenbestand, der die Beschreibung enthält. (Man beachte, dass hier bewusst der Begriff „Datei“ vermieden wird. Die Speicherung von Inhalt und Beschreibung kann sowohl im Dateisystem, als auch in beliebigen anderen Datenspeichern wie beispielsweise Datenbanken geschehen. Um nicht gedanklich auf die Speicherung im Dateisystem fixiert zu sein, wird hier der neutrale Begriff „Datenbestand“ gewählt.)

Ein weiterer Punkt betrifft insbesondere Lernobjekte: Für die Nutzung von Lernobjekten und ihren Beschreibungen werden unterschiedliche Rechte vergeben. Die Beschreibungen der Lernobjekte sind im allgemeinen öffentlich zugänglich, während die Inhalte der Lernobjekte je nach Autor auch kommerziell genutzt werden können. Konkret bedeutet dies, dass die Inhalte der Lernobjekte vor unberechtigten Zugriffen geschützt werden müssen. Bei den Beschreibungen der Lernobjekte ist dies nicht nötig.

7.3 Komunikation zwischen den Komponenten der Annotationsinfrastruktur

7.3.1 Kommunikation mittels Web Services

Die einzelnen Komponenten der im letzten Kapitel vorgestellten Architektur können sehr unterschiedlich realisiert sein. Um hier größtmögliche Flexibilität sicherzustellen, wird die gesamte Kommunikation der einzelnen Komponenten untereinander auf der Basis von Web Service-Technologien realisiert. Web Services sichern die Plattformunabhängigkeit der einzelnen Komponenten und ermöglichen eine weitgehende Entkopplung der Komponenten voneinander. Es ist so ohne Probleme möglich, eine Komponente gegen eine andere auszutauschen, ohne dass man sich Gedanken über Versionen von Kommunikationsprotokollen, Client- und Server-Stubs machen müsste. An dieser Stelle soll nicht weiter auf die Ideen, Konzepte und Technologien, die hinter Web Services stehen, eingegangen werden. Der interessierte Leser sei auf die verfügbare Literatur, beispielsweise auf [KuWö02], verwiesen.

7.3.2 Transport von Modulen, Atomen und Kursen mittels Wrapper-Klassen

In einigen Fällen müssen Module, Atome oder ganze Kurse als Argumente oder Ergebnisse übertragen werden. Für Module und Kurse ist dies nicht weiter problematisch, da sie im wesentlichen Eigenschaften und Identifikatoren von Subkomponenten enthalten. Beide Arten von Informationen lassen sich gut in XML-Dateien ablegen und in Objekten kapseln, die dann serialisiert übertragen werden können.

Für Atome ist dies nicht so einfach möglich, da sie sowohl aus XML-Daten bestehen, die die verschiedenen Eigenschaften des Atoms beschreiben, als auch aus den Daten selbst, die das Atom enthält. Beide Datenblöcke – die XML-Beschreibung sowie die Daten (Texte, Bilder, Videos, etc.) – werden als getrennte Einheiten im jeweiligen Datenspeicher (Dateisystem, Datenbank, etc.) abgelegt.

Um den Transport von Atomen, Modulen und Kursen einheitlich gestalten zu können, werden im Prototypen Wrapper-Klassen verwendet. Für Module und Kurse und für Atome gibt es verschiedene Wrapper-Klassen. Die Wrapper-Klassen für Module und Kurse implementieren folgendes Interface:

```
public interface ModuleWrapperInterface
{
  // Zugriff und Modifikation von Subelementen

  public insertElementID(String componentID, int position);
  public String getElementID(int position);
  public Iterator getElementIDs();

  // Zugriff und Modifikation von Parametern

  public void setParameter(String name, String value);
  public String getParameter(String name);
  public Properties getParameters();
}
```

Die Wrapper-Klassen für Atome implementieren dieses Interface:

```
public interface AtomWrapperInterface
{

  // Zugriff auf Inhaltsdaten des Atoms

  public setData(Object data);
  public Object getData();

  // Zugriff und Modifikation von Parametern

  public void setParameter(String name, String value);
  public String getParameter(String name);
  public Properties getParameters();
}
```

Wichtig ist, dass die Daten des Atoms serialisierbar sind, also dass sie in einem Format dargestellt werden können, das über das Internet transportiert werden kann. Andernfalls ist ein Transport der Atomdaten nicht möglich und es kommt zu einer Fehlermeldung.

7.4 Realisierung der Komponenten

In diesem Abschnitt wird darauf eingegangen, wie die einzelnen Komponenten, die in 6.3 bereits konzeptionell beschrieben wurden, realisiert sind. Um einfach zwischen der Beschreibung des Konzepts und der jeweiligen Realisierung hin und herwechseln zu können, ist dieser Abschnitt weitgehend identisch zu Abschnitt 6.3 aufgebaut.

7.4.1 Umsetzung der Lernobjektzugriffsschicht

7.4.1.1 Überblick über die betrachteten Aspekte

Aufgabe der Lernobjektzugriffsschicht ist es, einen einheitlichen Zugriff auf Lernobjekte unabhängig von der Art des Datenspeichers zu schaffen, in dem sie abgelegt sind. Damit dies gelingt, muss die Lernobjektzugriffsschicht so gestaltet sein, dass sie alle Besonderheiten des Zugriffs aller unterstützter Datenspeicher (Datenbank, Dateisystem, etc.) kapselt.

Der Zugriff auf die Lernobjekte ist eng gekoppelt mit den verschiedenen Rechtesystemen, die den Zugriff auf die Lernobjekte regeln. Ebenso wie es verschiedene Datenspeicher gibt, in denen die Lernobjekte abgelegt werden, gibt es auch eine Vielzahl von Zugriffskontrollsystemen. Auch für diese Systeme muss ein einheitlicher Zugriff realisiert werden.

Beiden Aufgaben – einheitlicher Zugriff auf verschiedene Datenspeicher und einheitliche Nutzung unterschiedlicher Rechtesysteme – kommt zentrale Bedeutung zu. Im folgenden wird daher beschrieben, wie ein einheitlicher Zugriff auf verschiedene Datenspeicher und eine einheitliche Nutzung der verschiedenen Rechtesysteme im Prototypen realisiert ist.

7.4.1.2 Architektur der Lernobjektzugriffsschicht

Die Lernobjektzugriffsschicht kapselt den Zugriff auf eine Vielzahl unterschiedlicher Datenspeicher, die jeweils verschiedene Wege des Zugriffs auf die Lernobjekte bieten. So sind die Abläufe bei Zugriffen auf Lernobjekte bei einem Dateisystem als Datenspeicher grundsätzlich anders als beispielsweise bei Dokumentenmanagementsystemen.

Die Anpassung der Lernobjektzugriffsschicht an die unterschiedlichen Datenspeicher erfolgt mittels eines Plugin-Mechanismus. Die Lernobjektzugriffsschicht besteht somit aus zwei Teilen (vgl. Abbildung 7-2): Einer *Schnittstellenschicht*, die einheitliche Dienste entsprechend Tabelle 6-2 als Web Services nach außen anbietet, und *Plugins*, die die konkret verwendeten Datenspeicher und Rechtesysteme an die Schnittstellenschicht anbinden, indem sie die einheitlichen Dienste der Schnittstellenschicht auf Dienste des Datenspeichers bzw. des Rechtesystems abbilden. Die Plugins werden von der Schnittstellenschicht verwaltet.

Die Dienste, die die Lernobjektzugriffsschicht nach außen anbietet, werden in der Schnittstellenschicht der Lernobjektzugriffsschicht realisiert. Die Schnittstellenschicht ist damit der Teil der Lernobjektzugriffsschicht, der von anderen Komponenten beim Aufruf von Diensten der Lernobjektzugriffsschicht angesprochen wird. Sie arbeitet als „Fassade“ (vgl. [GHJV96]) für die von der Lernobjektzugriffsschicht verwalteten Datenspeicher und Rechtesysteme.

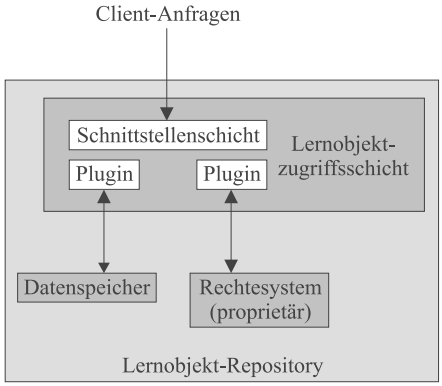


Abbildung 7-2: Zusammenspiel Lernobjektzugriffsschicht – Datenspeicher

Hier ist wichtig zu bemerken, dass das Rechtesystem, das den Zugriff auf einen Datenspeicher regelt, proprietär ist und je nach Realisierung des Datenspeichers anders implementiert sein kann. Ein Web-basierter Datenspeicher könnte beispielsweise Netegrity SiteMinder

(vgl. [SiteMinder]) verwenden, während sich ein dateibasierter Datenspeicher auf die im Betriebssystem vorhandenen Dateizugriffsrechte abstützt. Die Lernobjektzugriffsschicht vereinheitlicht somit nicht nur den Zugriff auf die Lernobjekte, sondern bietet auch eine einheitliche Schnittstelle für verschiedene, untereinander auch konzeptionell völlig unterschiedliche Rechtesysteme.

Die Plugins zur Kapselung der Datenspeicher und der Rechtesysteme sind Java-Klassen, die bestimmte Interfaces implementieren. Wie die Anbindung der Datenspeicher und der Rechtesysteme im Detail funktioniert, wird im folgenden behandelt.

7.4.1.3 Anbindung der Datenspeicher für Lernobjekte

Für jeden angebundenen Datenspeicher hält die Schnittstellenschicht eine Instanz eines Plugins, das den Zugriff auf den Datenspeicher kapselt. Ein Plugin realisiert somit das Entwurfsmuster „Adapter“ [GHJV96], um die spezifischen Eigenschaften des jeweiligen Datenspeichers zu verbergen. Angenommen, die Schnittstellenschicht verwaltet ein Dokumentenmanagementsystem und zwei Datenbanken, so wird eine Instanz eines Plugins, das das Dokumentenmanagementsystem anbindet, und zwei Instanzen eines Plugins, das die Datenbanken ansprechen kann, angelegt.

Die Plugins zur Anbindung der Datenspeicher implementieren folgendes Interface:

```
public interface DataStoreInterface
{
  public void init(Properties configData);
  public boolean objAccess_isAvailable(String userID, String
    objID);
  public Object objAccess_get(String userID, String objID)
    throws SecurityException;
  public Object objAccess_put(String userID, String objID,
    ModuleWrapperInterface obj) throws SecurityException;
  public Object objAccess_put(String userID, String objID,
    AtomWrapperInterface obj) throws SecurityException
}
```

Um Atome oder Module in einem Datenspeicher abzulegen, stellt das Plugin die beiden Methoden objAccess_put() bereit. Bei Aufruf dieser Methoden legt das Plugin ein Atome bzw. Modul entsprechend der Realisierung des Datenspeichers ab. Als Name des Atoms bzw. des Moduls wird dessen Identifikator verwendet. So ist eine einfache Abbildung „Identifikator → (Datei-)Name“ möglich.

Das Plugin implementiert in der Methode objAccess_get() die Abläufe, die bei einem bestimmten Datenspeicher nötig sind, um Lernobjekt objID zu liefern. Analog implementiert das Plugin in der Methode objAccess_available() die Abläufe um zu prüfen, ob ein Lernobjekt in einem bestimmten Datenspeicher abgelegt ist.

Bei Aufruf des Dienstes get() der Lernobjektzugriffsschicht (genauer: der Schnittstellenschicht, denn diese stellt den Dienst get() nach außen bereit) muss zuerst geprüft werden, in welchem der Datenspeicher (Datenbank, Dateisystem, Dokumentenmanagementsystem, etc.), den die Lernobjektzugriffsschicht verwaltet, das Lernobjekt objID abgelegt ist. Hierzu führt die Schnittstellenschicht zuerst bei allen installierten Plugins – für jeden Datenspeicher eine Instanz – die Methode objAccess_isAvailable() des Plugins aus[105]. Liefert objAccess_isAvailable() den Wert true, so wird sofort die Methode objAccess_get() des Plugins aufgeru-

[105] Aus Performance-Gründen ist es sinnvoll, dass jedes Plugin einen Katalog unterhält, in dem abgelegt ist, welche Lernobjekte in dem von ihm verwalteten Datenspeicher abgelegt sind.

fen, womit das Lernobjekt vom Datenspeicher angefordert wird[106]. Wird Lernobjekt objID in keinem Datenspeicher gefunden, so wird der Komponente, die den Dienst get() der Schnittstellenschicht aufgerufen hat, ein Fehler signalisiert.

Falls Nutzer userID den Dienst get() aufruft, er aber nicht berechtigt ist, auf Lernobjekt objID zuzugreifen oder er sich noch nicht mittels des Dienstes authenticate() der Schnittstellenschicht authentifiziert hat, wird eine SecurityException geworfen. Der Komponente, die den Dienst get() aufgerufen hat, wird daraufhin ein Fehler signalisiert.

Der Dienst put() der Lernobjektzugriffsschicht ist analog implementiert.

Die Konfiguration eines Plugins wird in der Methode init() realisiert. Diese Methode bekommt eine Menge configData von Attribut-Wertpaaren, die alle Daten beinhaltet, um auf einen Datenspeicher zugreifen zu können. Tabelle 7-4 nennt einige Beispiele. Im Prototypen wurde lediglich ein Plugin für das Dateisystem als Datenspeicher implementiert. Plugins für weitere Datenspeicher (Datenbanken, etc.) können analog durch Implementierung des DataStoreInterface (vgl. vorherige Seite) erstellt werden.

Datenspeicher	Attributname	Beschreibung
Datenbank	connectString	URL, unter der die Datenbank angesprochen wird. Der connectString enthält Host-Namen, Port-Nummer, Nutzerkennung, Datenbankname, etc.
Dateisystem	login	Kennung, unter der alle Dateisystemzugriffe durchgeführt werden.
	password	Passwort für die Kennung
	documentRoot	Wurzelverzeichnis, unter dem alle Dokumente abgelegt werden

Tabelle 7-4: Beispiele für Konfigurationsdaten des Datenspeicher-Plugins

7.4.1.4 Anbindung externer Rechtesysteme

Vor dem ersten Zugriff auf ein Lernobjekt muss sich der Nutzer gegenüber dem Datenspeicher, in dem das Lernobjekt abgelegt ist, authentifizieren. Dies geschieht mit dem Dienst authenticate(), den die Lernobjektzugriffsschicht (genauer: die Schnittstellenschicht) bereitstellt. Ohne eine erfolgreiche Authentifizierung ist kein Zugriff auf Lernobjekte möglich.

Der Zugriff auf einen Datenspeicher kann unterschiedlich geregelt werden; verschiedene Rechtesysteme sind einsetzbar. Es ist möglich, von vorneherein im Datenspeicher verfügbare Rechtesysteme zu nutzen (beispielsweise das Rechtesystem, das moderne Betriebssysteme anbieten, falls das Dateisystem als Datenspeicher verwendet wird), aber auch externe Rechtesysteme können für den Zugriffsschutz eingesetzt werden. Welche Rechtesysteme zum Schutz der Lernobjekte eingesetzt werden, liegt in der Verantwortung der Betreiber der Datenspeicher, in denen die Lernobjekte abgelegt sind, und kann hier nicht beeinflusst werden. Umso wichtiger ist eine möglichst flexible Anbindung der Schnittstellenschicht an das Rechtesystem.

Je nach eingesetztem Rechtesystem sind verschiedene Abläufe bei der Authentifizierung nötig. Analog zum Zugriff auf verschiedene Datenspeicher werden diese Abläufe auch hier über einen Plugin-Mechanismus realisiert; die Plugins für die Rechtesysteme sind wieder als

[106] Bei diesem Zugriff wird zugleich auch durch das Rechtesystem geprüft, ob der Nutzer, unter dessen Kennung der Dienst get() aufgerufen wurde, die Berechtigung hat, das Lernobjekt objID anzufordern. Dies geschieht im Datenspeicher, der hierzu das Rechtesystem kontaktiert, und läuft für das Plugin transparent ab.

Adapter implementiert und werden in die selbe Schnittstellenschicht eingehängt wie die Plugins zur Anbindung der verschiedenen Datenspeicher. Ebenso wie die Schnittstellenschicht die Plugins zur Kapselung der Datenspeicher verwaltet, verwaltet sie auch die Plugins zur Anbindung der Rechtesysteme. Für jedes Rechtesystem gibt es eine Instanz des Plugins.

Ein Plugin zur Anbindung eines Rechtesystems ist wieder eine Java-Klasse, die in die Schnittstellenschicht eingehängt wird und die folgendes Interface implementiert:

```
public interface SecurityInterface
{
  public void init(Properties configData);
  public boolean objAccess_authenticate(String userID, Object
    credentials) throws SecurityException;
}
```

Das Plugin implementiert in der Methode objAccess_authenticate() den Ablauf, der nötig ist, um einen Nutzer userID beim Rechtesystem zu authentifizieren. Dieser Schritt ist nötig um zu prüfen, ob Nutzer userID überhaupt auf ein Lernobjekt zugreifen darf. Diese Methode wird genutzt, wenn eine Komponente den Dienst authenticate() der Lernobjektzugriffsschicht (genauer: der Schnittstellenschicht) aufruft. Als Beweis der Authentizität der Nutzers userID wird ein Objekt credentials mit übergeben, das die Authentifizierungsdaten – Passwort, X509-Zertifikat, etc. – enthält. Anhand dieses Objekts erfolgt im Rechtesystem die Authentifizierung. Schlägt die Authentifizierung fehl, so wird eine Security-Exception erzeugt und der aufrufenden Komponente ein Fehler signalisiert.

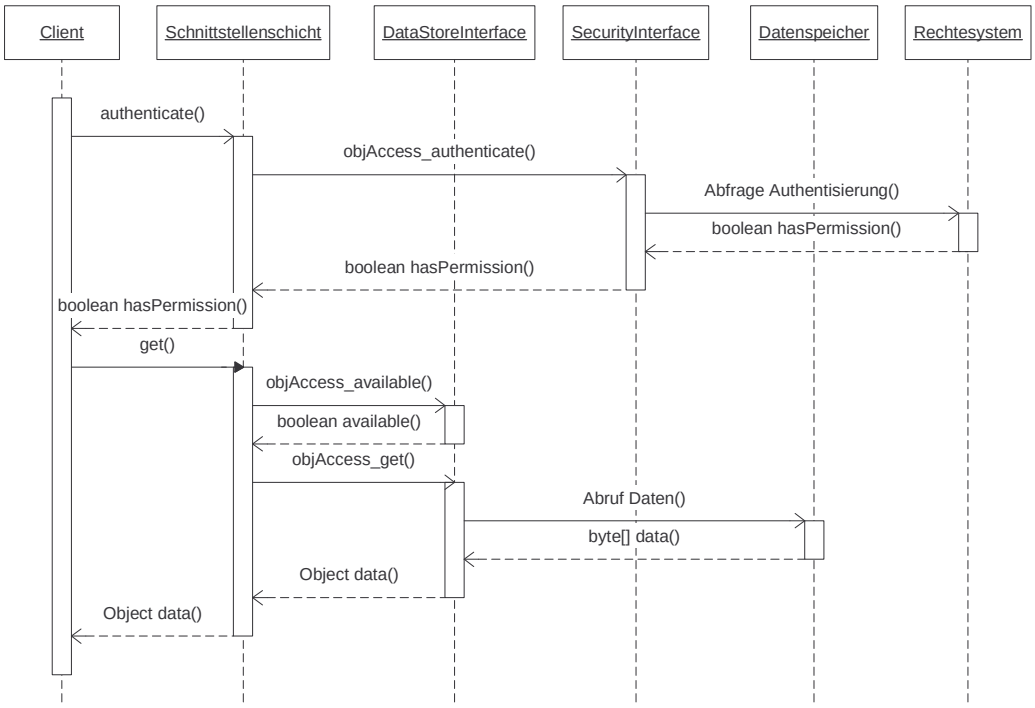


Abbildung 7-3: Zusammenspiel von Datenspeicher und Rechtesystem

Die Konfiguration des Plugins wird analog zu den Plugins zur Anbindung der Datenspeicher in der Methode init() realisiert. Diese Methode bekommt eine Menge Attribut-Wertpaare configData, die alle Daten beinhaltet, um das Rechtesystem nutzen zu können. Welche Daten hierbei in configData abgelegt sind, hängt vom verwendeten Rechtesystem ab; denkbar sind URLs von Policy-Servern, Pfade zu Dateien, die Nutzer- oder Gruppeninformationen enthalten, etc.

7.4.1.5 Zusammenfassung: Funktionsweise der Lernobjektzugriffsschicht

Für die Dienste, die die Lernobjektzugriffsschicht bereitstellt, werden Datenspeicher und die sie schützenden Rechtesysteme zusammen angesprochen. Abbildung 7-3 zeigt das Zusammenspiel von Datenspeicher und Rechtesystem am Beispiel des Dienstes get().

7.4.2 Umsetzung des Lernobjekt-Service

7.4.2.1 Überblick über die betrachteten Aspekte

Die dezentrale Natur der Speicherung der Lernobjekte erfordert einen Vermittlungsdienst, über den abgerufen werden kann, wo (unter welcher URL) bestimmte Lernobjekte zu finden sind (Abbildung „Lernobjekt-ID → URL der Lernobjektzugriffsschicht“). Des weiteren ist ein Dienst nötig, der anhand einer Beschreibung von Lernobjekten durch Parameter konkrete Lernobjekte liefert, die diesen Beschreibungen genügen (Abbildung „Menge von Parametern → Menge von Lernobjekt-IDs“). Diese Dienste werden vom Lernobjekt-Service realisiert. Der Lernobjekt-Service stellt hierzu die in Tabelle 6-3 genannten Dienste als Web Services zur Verfügung.

Je nach Anfragetyp ergeben sich damit folgende Abläufe: Falls der Client den Identifikator des gesuchten Lernobjekts kennt, liefert eine Anfrage beim Lernobjekt-Service die URL der Lernobjektzugriffsschicht, die das fragliche Lernobjekt verwaltet. Über diese URL kann der Client dann auf das Lernobjekt zugreifen (vgl. Abbildung 7-4).

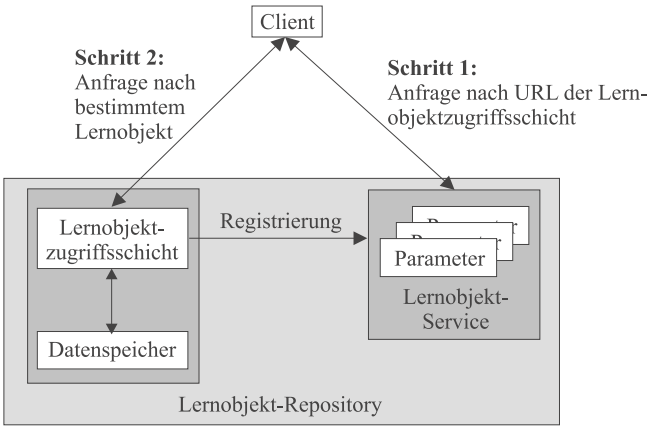


Abbildung 7-4: Anfrage nach einem Lernobjekt anhand dessen Identifikator

Falls der Client den Identifikator des Lernobjekts nicht kennt, sondern lediglich eine Beschreibung von passenden Lernobjekten in Form einer Menge von Parametern hat, ist ein weiterer Schritt nötig: Zuerst stellt der Client eine Anfrage an den Lernobjekt-Service nach Lernobjekten, die zu der Beschreibung passen. Der Lernobjekt-Service liefert dann eine (möglicherweise leere) Menge von Identifikatoren von Lernobjekten zurück. Der Client ermittelt dann in einem zweiten Schritt erneut über eine Anfrage an den Lernobjekt-Service für jeden Identifikator eines Lernobjekts die URL der jeweils für das Lernobjekt zuständigen Lernobjektzugriffsschicht. Über diese URL kann der Client dann das Lernobjekt nutzen (vgl. Abbildung 7-5).

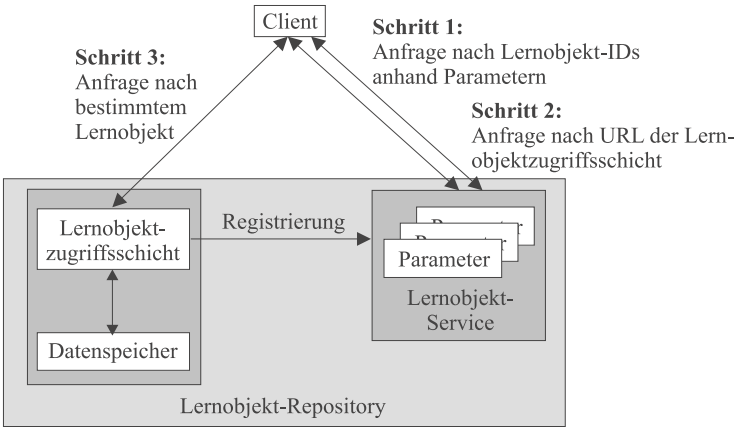


Abbildung 7-5: Anfrage nach einem Lernobjekt anhand dessen Beschreibung

7.4.2.2 Vermittlung von Lernobjektzugriffsschichten

Dieser Vermittlungsdienst des Lernobjekt-Service liefert mit dem Dienst getServers() (vgl. Tabelle 6-3) die URLs aller Lernobjektzugriffsschichten, auf denen ein Lernobjekt, gegeben durch dessen Identifikator, abgelegt ist. Ist kein Lernobjekt mit dem entsprechenden Identifikator bekannt, so ist die Menge der Lernobjekte, die den Anfragern (Clients) als Ergebnis geliefert wird, leer.

Grundlage für die Durchführung dieser Dienste ist eine Tabelle (objID, URL) in einer Datenbank, die der Lernobjekt-Service nutzt. Der Dienst register() befüllt diese Tabelle, der Dienst getServers() führt Abfragen auf dieser Tabelle aus.

7.4.2.3 Vermittlung von Lernobjekten mit bestimmten Eigenschaften

Aufgabe dieses Dienstes ist es, Lernobjekte mit bestimmten Eigenschaften zu suchen. Dies wird durch die Dienste getObjIDs() und getDescr() (vgl. Tabelle 6-3) realisiert. Basis der Suche sind dabei die Parameter, die für jedes Lernobjekt gespeichert sind (vgl. Dienst register()).

Die für diese Dienste benötigten Daten werden in der Tabelle (objID, Attribut, Wert) abgelegt. Mit register() werden Einträge in dieser Tabelle vorgenommen, mit getObjIDs() werden Suchen auf dieser Tabelle durchgeführt.

7.4.2.4 Speicherung der Strukturinformationen von Kursen und Modulen

Die Strukturinformationen von Kursen und Modulen wird durch Kompositionsbeziehungen zwischen Modulen bzw. Kursen und ihren Subelementen sowie durch Assoziationen zwischen Lernobjekten realisiert. Jede Beziehungsart – egal ob Komposition oder Assoziation – wird als Tabelle (conID, objID1, objID2) von Identifikatoren implementiert. Je nach Art der Beziehung haben objID1 und objID2 unterschiedliche Bedeutung: Bei Kompositionen ist objID1 das übergeordnete Modul bzw. der Kurs und objID2 das Subelement. Bei Assoziationen sind objID1 und objID2 einfach die Identifikatoren zweier Lernobjekte.

Die Dienste addSubconnection(), und addAssociation() fügen Elemente zu dieser Tabelle hinzu, die Dienste removeSubconnection() und removeAssociation() entfernen Elemente aus der Tabelle. Die Inhalte der Tabelle können mit den Diensten getSubconnections() sowie getAssociations() ausgelesen werden (alle Dienste vgl. Tabelle 6-3).

7.4.3 Umsetzung des Annotations-Repositories

7.4.3.1 Überblick über die betrachteten Aspekte

Das Annotations-Repository ist die Komponente in der Annotationsinfrastruktur, die die Annotationen speichert, verwaltet und mit den annotierten Lernobjekten bzw. Annotationen mittels Assoziationen verbindet. Das Annotations-Repository ist damit die zentrale Komponente in der Annotationsinfrastruktur.

In den folgenden Abschnitten wird beschrieben, aus welchen Komponenten das Annotations-Repository aufgebaut ist und wie die Annotationen sowie die zu ihnen gehörenden Assoziationen gespeichert werden. Die Dienste, die das Annotations-Repository hierzu bereitstellt, wurden in Tabelle 6-4 bereits vorgestellt und werden als Web Services nach außen angeboten. Um die Funktionsweise des Annotations-Repositories zu illustrieren, wird des weiteren erläutert, welche Schritte im Detail beim Anlegen oder Löschen von Annotationen oder beim Zugriff auf Annotationen durchgeführt werden.

7.4.3.2 Architektur des Annotations-Repositories

Als Ergebnis von 6.3.3.3 wurde als Speicherort der Annotationen die Rechner identifiziert, auf denen bereits der Lernobjekt-Service läuft. Auf diesen Rechnern wird ein Repository aufgebaut, in dem alle Annotationen gespeichert werden: das Annotations-Repository. Das Annotations-Repository realisiert dabei folgende Funktionalität:

- Funktionalität zur Speicherung der Annotationen sowie der Assoziationen, die die Verbindung von Annotation und annotiertem Lernobjekt bzw. annotierter Annotation realisieren.
- Suche nach allen Annotationen zu einem Lernobjekt mit gegebener Lernobjekt-ID.
- Suche nach allen Annotationen zu einer Annotation mit gegebener Annotations-ID.
- Suche nach allen Annotationen mit bestimmten, durch Parameter beschriebenen Eigenschaften.

Das Annotations-Repository ist in seiner Funktionalität somit deutlich umfangreicher als ein Datenspeicher, der lediglich Annotationsdaten aufnimmt und verwaltet.

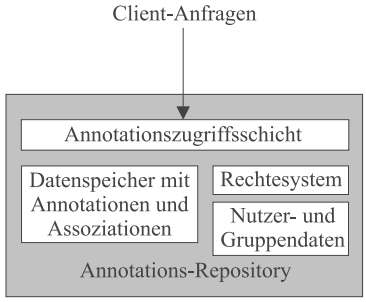


Abbildung 7-6: Aufbau des Annotations-Repositories

Das Annotations-Repository enthält eine Komponente zur Zugriffssteuerung, die regelt, welcher Anforderer auf welche Annotation zugreifen darf. Diese Komponente verwaltet auch die in der Annotationsinfrastruktur bekannten Kennungen von Nutzern und Designern sowie die definierten Gruppen. Des weiteren enthält das Annotations-Repository eine Schicht, die den Datenspeicher des Annotations-Repositories sowie die Komponente zur Zugriffssteuerung kapselt. Diese Schicht ist die in 6.3.3.4 genannte Annotationszugriffsschicht. Sie bietet die in Tabelle 6-5 genannten Dienste als Web Services an, integriert die Zugriffsfunktionalität für den Datenspeicher mit dem Rechtesystem und realisiert so die Schnittstelle des Anno-

tations-Repositories nach außen. Abbildung 7-6 zeigt die Architektur des Annotations-Repositories im Überblick.

Als Datenspeicher für die Annotationen (die Atome, die die Annotationen kapseln, und die Annotationstexte) wird im Prototypen derzeit nur das Dateisystem unterstützt. Eine Erweiterung, um weitere Datenspeicher nutzen zu können, ist problemlos mittels des bereits in 7.4.1.2 vorgestellten Plugin-Mechanismus möglich. Die gleiche Einschränkung gilt derzeit für das im Annotations-Repository eingesetzte Rechtesystem: Es wird bislang ausschließlich das Rechtesystem des Dateisystems (Dateieigentümer, Gruppe, sonstige Zugreifende) unterstützt. Auf dieses Rechtesystem werden derzeit die Zugriffsfunktionen, die das Annotations-Repository nutzt, abgebildet.

Die Assoziationen, die die Verbindungen zwischen Lernobjekten und Annotationen realisieren, werden in einer Datenbank abgelegt. Die hierbei zu Grunde liegende Tabelle hat die Gestalt (ID, ownerID, anchorID, anchorArea, annotationID). OwnerID ist hierbei der Identifikator des Nutzers, der die Annotation und damit die Assoziation angelegt hat. AnchorID ist der Identifikator des annotierten Lernobjekts, Kurses bzw. Annotation, anchorArea ist eine (evtl. leere) Zeichenkette, in der der Anker der Annotation festgehalten ist (für Details zur Adressierung von Annotationen vgl. 7.2.1.3). Das Format dieser Zeichenkette ist durch die Gestalt des Ankers (Textausschnitt, rechteckiger Ausschnitt eines Bildes, etc.) festgelegt. AnnotationID schließlich ist der Identifikator der Annotation. Die Parameter der Assoziation werden wieder in einer Tabelle (ID, Attribut, Wert) abgelegt.

7.4.3.3 Abläufe bei der Verwaltung von Annotationen

In den folgenden Abschnitten wird skizziert, welche Abläufe im Annotations-Repository stattfinden, wenn bestimmte Aktionen durchgeführt werden sollen. Besonderer Wert wird hierbei auf das Zusammenspiel der einzelnen Komponenten des Annotations-Repositories – Datenspeicher für die Annotationsdaten, Datenbank für die Assoziationen, Rechte- und Zugriffssystem – gelegt.

Neue Annotation erstellen

Annotationen werden von Designern und Nutzern mittels Annotationseditoren erstellt. Es findet zwar eine Authentifizierung der Nutzer statt, um der Annotation einen Autor zuordnen zu können, aber da jeder registrierte Designer oder Nutzer eine Annotation erstellen darf, wird keine weitere Überprüfung der Nutzerrechte vorgenommen.

Um eine neue Annotation mit eigenem Text zu erstellen, wird vom Annotationseditor der Dienst putAnnotationText(objID, objArea, text, courseID, userID) (vgl. Tabelle 6-4) aufgerufen, wobei text die Anmerkung ist, die der Nutzer userID im Annotationseditor im Zusammenhang mit Kurs courseID (evtl. Null) angegeben hat. Im folgenden wird beschrieben, welche Schritte hierbei durchgeführt werden:

1. Der Text text wird in einer Datei gespeichert. Der Name der Datei ist zufällig gewählt, muss aber einzigartig sein, um eine eindeutige Identifikation der Datei zu ermöglichen. Hierzu bietet sich beispielsweise der Hash-Code ihres Inhalts an.
2. Ein Atom mit dem Medientyp „Text“ wird im Datenspeicher des Annotations-Repositories angelegt. Im Atom wird ein Verweis auf die im letzten Schritt angelegte Datei mittels eines Parameters content-file gesetzt. Der Identifikator des Atoms wird entsprechend den in 7.2.1.1 genannten Richtlinien gewählt.
3. Ein Modul, das das Atom aus Schritt 2 kapselt, wird im Datenspeicher des Annotations-Repositories angelegt. Der Identifikator anID des Moduls wird analog zu Schritt 2 wieder entsprechend den in 7.2.1.1 genannten Richtlinien gesetzt. Zusätzli-

che Beschreibungen der Annotation, die die Designer oder Nutzer im Annotationseditor eingegeben haben oder die der Annotationseditor zu Verwaltungszwecken mit der Annotation speichern möchte, werden als Parameter dieses Moduls gesetzt.

4. In der Datenbank, die die Assoziationen zwischen den annotierten Lernobjekten und den Annotationen verwaltet, wird eine neue Assoziation vom Typ „annotiert“ angelegt. Diese hat das angegebene Lernobjekt objID mit dem Bereich objArea als annotiertes Element (Setzen der Parameter anchor-id=objID und anchor-area=objArea), den Kurs courseID als Kontext, in dem die Annotation erstellt wurde, und den Identifikator des in Schritt 3 neu angelegten Moduls als Annotation gesetzt (Parameter annotation-id=anID).

Soll ein bereits bestehendes Lernobjekt (Atom oder Modul) oder ein bereits bestehender Kurs als Annotation dienen, so kann die Methode putAnnotation(objID, objArea, anID, courseID, userID) genutzt werden, vgl. Tabelle 6-4. Hierbei ist nur folgender Schritt zu tun:

In der Datenbank, die die Assoziationen zwischen den annotierten Elementen und den Annotationen verwaltet, wird eine neue Assoziation vom Typ „annotiert“ angelegt. Diese hat das angegebene Lernobjekt objID im Bereich objArea als annotiertes Element (Parameter anchor-id=objID und anchor-area=objArea) und den Identifikator anID als Annotation gesetzt (Parameter annotation-id=anID). Die Werte courseID und userID werden aus Verwaltungsgründen wieder in der Assoziation vermerkt.

Abbildung 7-7 fasst diese Abläufe zusammen.

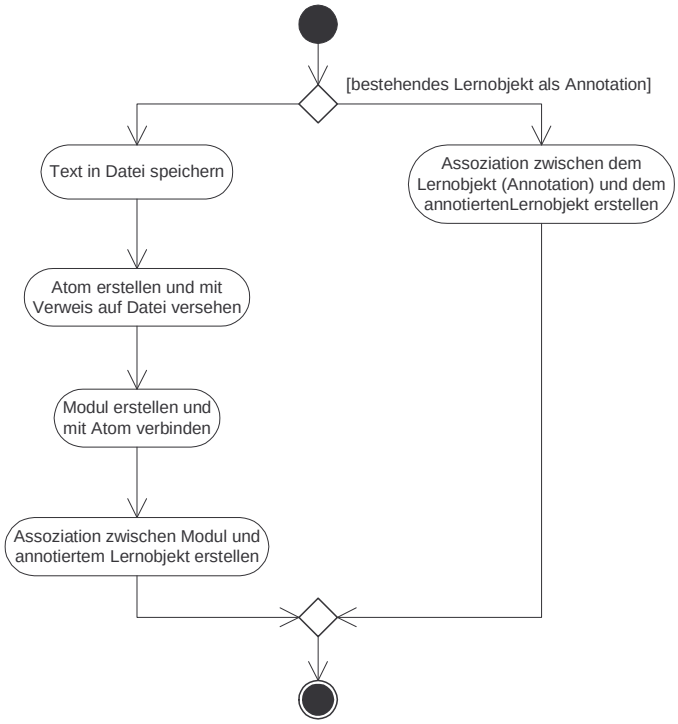


Abbildung 7-7: Schritte zur Erstellung einer neuen Annotation

Zugriff auf eine bestehende Annotation

Zugriff auf eine Annotation mit dem Identifikator anID ist mittels des Dienstes getAnnotation(anID, userID) möglich. UserID ist die Kennung des Nutzers, der auf die Annotation zugreifen will.

Die Rolle von userID ist vielfältig: Will ein einzelner Nutzer über den Annotationseditor auf die Annotation zugreifen, ist userID der Identifikator dieses Nutzers. Will hingegen der Kurs-Presenter einen Kurs für die Präsentation vorbereiten, wobei auch Annotationen in das endgültige Kursformat integriert werden sollen, so ist userID nicht der Identifikator des Kurs-Presenters, sondern ein allgemeiner Identifikator für alle Nutzer, der von dem Tool, das den präsentationsfertigen Kurs erzeugt, verwendet wird. Die Nutzung eines allgemeinen Identifikators ist notwendig, damit nicht private Annotationen des Kurs-Presenters oder Gruppenannotationen, die ja beide nicht öffentlich sichtbar sein dürfen, in den endgültigen Kurs integriert werden.

Bei Ausführung des Dienstes getAnnotation() werden folgende Schritte durchgeführt:

1. Es wird geprüft, ob der Nutzer mit dem Identifikator userID die Berechtigung hat, die Annotation abzurufen: Hierzu wird zuerst die Assoziation, die das annotierte Element mit der Annotation verbindet, von der Datenbank angefordert. Suchkriterium ist hierbei anID. In der Assoziation ist – falls es sich bei der Annotation nicht um eine öffentliche Annotation handelt – der Identifikator ownerID des Nutzers hinterlegt, der die Annotation und damit die Assoziation angelegt hat. Es kann nun einfach überprüft werden, ob userID und ownerID identisch oder in der selben Gruppe (falls die Annotation eine Gruppenannotation ist) sind. Hat userID nicht die erforderlichen Rechte, wird eine Fehlermeldung als Ergebnis zurückgeliefert.
2. Das Modul mit dem Identifikator anID wird vom Datenspeicher des Annotations-Repositories gelesen. Ist dieses Modul vorhanden, so handelt es sich um eine Annotation, die Text enthält und nicht auf ein anderes Lernobjekt verweist[107].

   Handelt es sich um eine Annotation, die Text enthält, werden folgende Schritte durchgeführt:

   a) Aus dem Modul wird der Identifikator des Atoms, das den Annotationstext enthält, ermittelt. Dies wird über die Komposition zwischen dem Atom und dem das Atom umfassende Modul realisiert.

   b) Das Atom wird vom Datenspeicher des Annotations-Repositories angefordert.

   c) Aus dem Atom wird der Name der Datei extrahiert, die den Text enthält. Der Dateiname ist im Parameter content-file des Atoms enthalten und entspricht im allgemeinen dem Identifikator des Atoms.

   d) Das Atom sowie der Inhalt der Datei, die den Text enthält, werden in eine Wrapper-Klasse (vgl. 7.3.2) gepackt; diese wird serialisiert und als Ergebnis an den Anfrager zurückgeliefert.

   Handelt es sich hingegen um eine Annotation, die auf ein anderes Lernobjekt verweist, werden folgende Schritte durchgeführt:

   a) Es wird eine Fehlermeldung, in der darauf hingewiesen wird, dass anID der Identifikator eines Lernobjekts ist, als Ergebnis zurückgeliefert. Das aufrufende Programm weiß dann, dass es den Speicherort des Lernobjekts anID über den Lernobjekt-Service erfragen und über die jeweilige Lernobjektzugriffsschicht abrufen kann.

[107] Annotationen, die auf andere Lernobjekte verweisen, werden nicht vom Annotations-Repository verwaltet. Daher kann davon ausgegangen werden, dass eine Annotation Text enthält, wenn sie im Annotations-Repository verfügbar ist.

b) Das aufrufende Programm kann auf diese Fehlermeldung auf verschiedene Arten reagieren: Es kann den Nutzer informieren, dass die Annotation ein bereits bestehendes Lernobjekt ist. Oder das Programm lädt kommentarlos das Lernobjekt (über Anfragen an den Lernobjekt-Service und die jeweilige Lernobjektzugriffsschicht) aus dem Lernobjekt-Repository und präsentiert sie dem Nutzer.

Die für den Zugriff auf eine Annotation nötigen Schritte fasst Abbildung 7-8 schematisch zusammen.

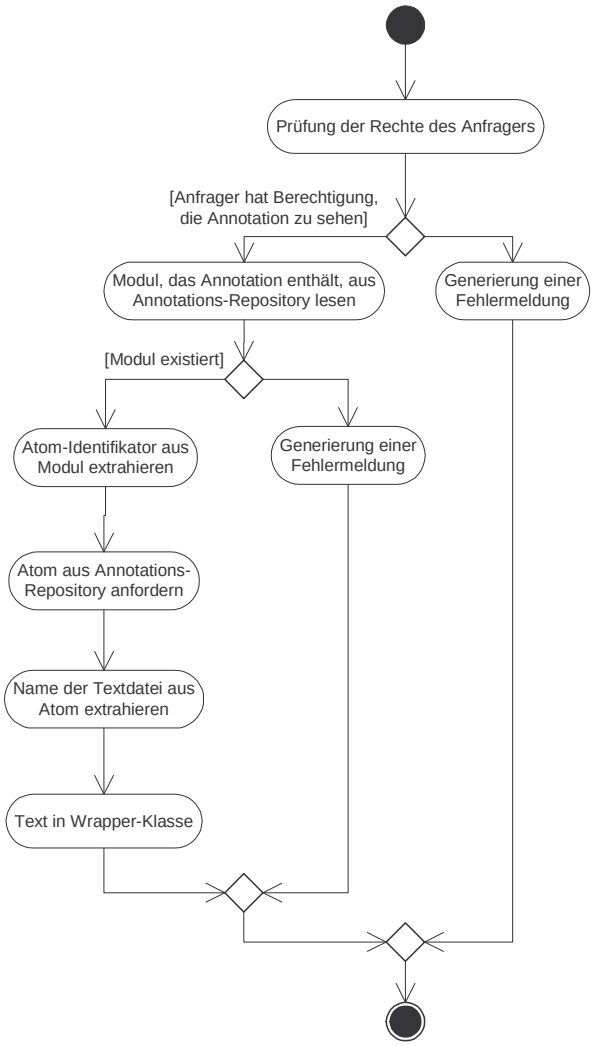


Abbildung 7-8: Schritte zum Zugriff auf eine Annotation

Annotation löschen

Eine Annotation anID kann mittels des Dienstes deleteAnnotation(anID, userID) (vgl. Tabelle 6-4) gelöscht werden. Dabei muss wieder unterschieden werden, ob die Annotation auf ein Lernobjekt verweist oder eine textuelle Anmerkung (Text, den Text kapselndes Atom und das Atom kapselndes Modul) ist. Hierbei werden folgende Schritte durchgeführt:

1. Es wird geprüft, ob der Nutzer mit dem Identifikator userID die Berechtigung hat, die Annotation zu löschen: Hierzu wird wieder zuerst die Assoziation, die das annotierte Element mit der Annotation verbindet, von der Datenbank des Annotations-

Repositories angefordert. Suchkriterium ist hierbei wieder der Identifikator der Annotation, anID. In der Assoziation ist der Identifikator ownerID des Nutzers hinterlegt, der die Annotation und damit die Assoziation angelegt hat. Es kann nun einfach überprüft werden, ob userID und ownerID identisch oder in der selben Gruppe sind. Hat userID nicht die erforderlichen Rechte, wird eine Fehlermeldung als Ergebnis zurückgeliefert.

2. Das Modul anID wird vom Datenspeicher des Annotations-Repositories gelesen und analysiert: Enthält das Modul genau ein Atom, das Text enthält, existiert also genau eine Kompositionsbeziehung zwischen dem Modul und einem Atom, so handelt es sich um eine Annotation, die Text enthält und nicht auf ein anderes Lernobjekt verweist.

   Handelt es sich um eine Annotation, die Text enthält, werden folgende Schritte durchgeführt:

   a) Die Assoziation, die die Verbindung zwischen dem annotierten Element und der Annotation realisiert, wird aus der Datenbank gelöscht.

   b) Anschließend werden die Datei, die den Text enthält, das Atom, das den Text kapselt, sowie das das Atom kapselnde Modul gelöscht.

   Handelt es sich hingegen um eine Annotation, die auf ein anderes Lernobjekt verweist, wird folgender Schritt durchgeführt:

   Die Assoziation, die die Verbindung zwischen dem annotierten Element und der Annotation realisiert, wird aus der Datenbank gelöscht.

   Wichtig ist hier, dass das Lernobjekt, das als Annotation dient, nicht gelöscht werden darf, da es (zukünftig) für weitere Module und Kurse zur Verfügung stehen muss.

Diese Schritte werden von Abbildung 7-9 illustriert.

7.4.4 Nutzung von Annotationen auf der Client-Seite

Wie bereits in 6.3.4 besprochen, haben die Designer von Lernobjekten und Lehrmaterialien eine grundsätzlich andere Sicht auf Lehrmaterialien als deren Nutzer. Die Designer sehen Lehrmaterialien als Bäume von Lernobjekten, während die Nutzer mit den präsentationsfertigen Kursen arbeiten, die bereits in bestimmten Formaten (PDF, HTML, etc.) vorliegen. Die Tools, die Designer und Nutzer verwenden, sind daher auch grundverschieden – und damit auch die Art und Weise, wie diese Tools die Annotationsinfrastruktur nutzen (vgl. auch die Sichten in 4.2.3.2). Dies wird im folgenden näher betrachtet.

7.4.4.1 Präsentation und Nutzung von Annotationen durch die Designer

Bei der Erstellung von Atomen, Modulen und Kursen haben die Designer eine Elementsicht auf die zu erstellenden Lernobjekte und Kurse, das heißt, sie sehen Atome, Module und Kurse als Elemente, die noch nicht im endgültigen Präsentationsformat (PDF, HTML, etc.) vorliegen. Es ist den Tools, die die Designer nutzen, damit möglich, zu einem gegebenen Lernobjekt die jeweils zugeordneten Annotationen unmittelbar mittels des Dienstes getAnnotation() des Annotations-Repositories zu ermitteln und anzuzeigen. Beim Abrufen von Lernobjekten können diese Tools zugleich alle Annotationen zum jeweiligen Lernobjekt ermitteln und dem Designer in einer eigenen Benutzeroberfläche präsentieren, vgl. Abbildung 7-10.

Die Erstellung von Annotationen zu Lernobjekten erfolgt analog: Da bereits das Lernobjekt, das annotiert werden soll, im Tool bearbeitet wird, hat das Tool bereits alle Informationen

zum Lernobjekt (insbesondere dessen Identifikator) und kann so leicht Annotationen mittels den Diensten putAnnotation() und putAnnotationText() an das Annotations-Repository übergeben.

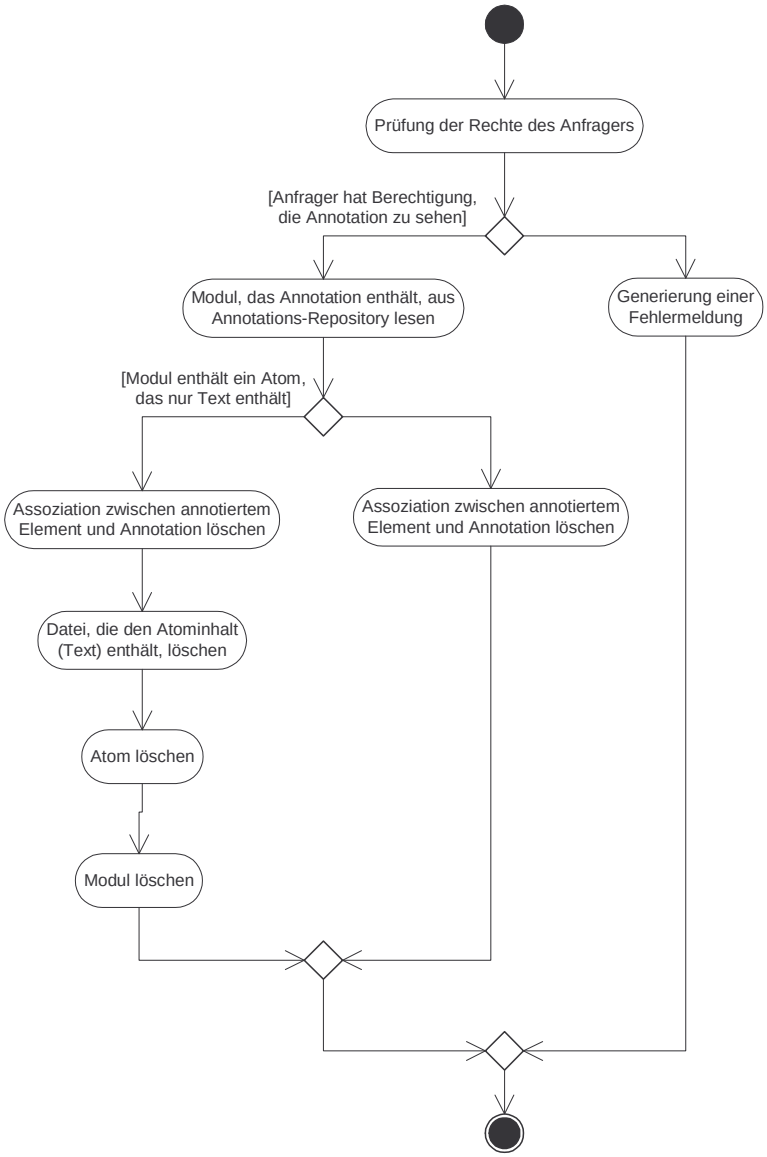


Abbildung 7-9: Schritte zum Löschen einer Annotation

7.4.4.2 Präsentation und Nutzung von Annotationen durch die Nutzer

Die Nutzer von Kursen sehen Annotationen als Teil des Kurses, denn die Annotationen zu einem Kurs werden bereits bei der Umsetzung des Kurses in das endgültige Präsentationsformat integriert, vgl. 6.3.4.2 Ob und auf welche Weise die Annotationen dargestellt werden (Fußnoten, Hyperlinks, etc.), hängt dabei vom Präsentationsformat sowie den jeweiligen Rechten der Zielgruppe, für die der Kurs erstellt wurde, ab. Die Annotationen sind somit integraler Bestandteil eines Kurses, sobald dieser in seiner endgültigen Präsentationsform vorliegt, und können mit den üblichen Viewern – je nach Präsentationsformat Web-Browser, PDF-Viewer, etc. – betrachtet werden.

Die Kurs-Presenter, Kurstutoren und Lerner sehen Kurse üblicherweise in dem Format, in das sie zur Präsentation konvertiert wurden. Das Präsentationsformat kann beispielsweise HTML oder Microsoft Powerpoint für Live-Präsentationen oder PDF oder Postscript für lehrveranstaltungsbegleitende Skripte sein. Die Atome, Module und Kursteile, aus denen der Kurs aufgebaut ist, sehen die Nutzer nicht, sondern sie sehen ein Dokument; sie können die Grenzen zwischen den einzelnen Elementen allenfalls bei ungünstigen Medienbrüchen erahnen.

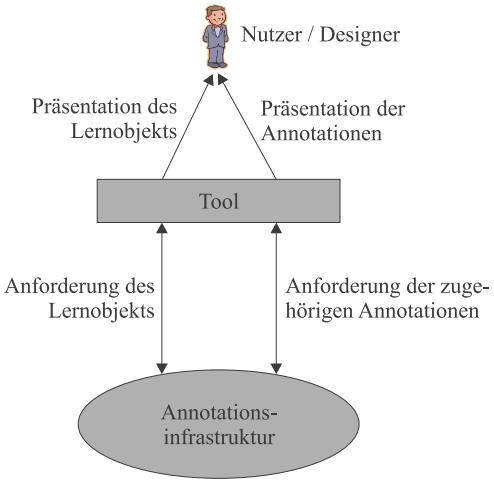


Abbildung 7-10: Integration von Lehrmaterial- und Annotationsbearbeitung[108]

Damit stellt sich das Problem, wie die Zuordnung von Annotationen zu Lernobjekten gestaltet werden kann. Letztlich muss eine Identifikation eines Tupels (anchor-id, anchor-area) stattfinden – falls mit einer Annotation mehrere Lernobjekte überspannend annotiert wurden, auch mehrere solcher Tupel.

Für die Identifikation des Lernobjekts, auf das sich eine Annotation bezieht, müssen zwei Fälle unterschieden werden: Lernobjekte können direkt einzelnen Präsentationselementen zugeordnet werden (beispielsweise wenn ein Lernobjekt als eine HTML-Seite dargestellt wird) oder Lernobjekte werden als Sequenz aneinandergehängt, so dass die einzelnen Lernobjekte in den Lehrmaterialien aufgehen (wie beispielsweise in einem vollständigen PDF-Skript, das vorlesungsbegleitend angeboten wird) und nicht mehr auf triviale Weise (ein Lernobjekt entspricht einer bestimmten Datei) identifiziert werden können. Beide Fälle werden nun im Detail betrachtet.

Fall 1: Jedes Lernobjekt liegt als eigene Datei vor

Dieser Fall liegt beispielsweise dann vor, wenn ein Kurs mit HTML als Präsentationsformat erstellt wird. Im folgenden wird davon ausgegangen, dass ein Kurs als Sammlung von HTML-Seiten vorliegt. Es wird eine HTML-Seite in einem Web-Browser angezeigt und die Stelle der HTML-Seite, die annotiert werden soll, wurde mit der Maus selektiert.

Um für diesem Fall eine Identifikation der zu annotierenden Lernobjekte zu erleichtern, wird das in 6.4 beschriebene Vorgehen zur Kurserstellung für den Fall, dass ein Lernobjekt in eine eigene Datei abgebildet wird, so erweitert, dass bei der Benennung einer HTML-Seite, in die ein Lernobjekt abgebildet wird, der Identifikator dieses Lernobjekts gleich im Dateina-

[108] Zur Erinnerung: Hier sind keine Komponenten und Interaktionen aufgeführt, die andere Nutzungen einer Lernplattform, beispielsweise die Verwendung bereits fertig erstellter Lehrmaterialien, die keine Annotationen enthalten, oder die Erstellung von Lernobjekten ohne die Nutzung von Annotationen, unterstützen.

men enthalten ist. So kann dann das Annotations-Tool anhand der URL der im Web-Browser angezeigten HTML-Seite den Identifikator des zu annotierenden Lernobjekts ermitteln. Der Parameter anchor-id der Assoziation, die die Annotation mit dem Lernobjekt verbindet, ist damit schon bekannt.

Für die vollständige Adressierung des annotierten Bereichs eines Lernobjekts muss neben dem Parameter anchor-id auch der Parameter anchor-area korrekt gesetzt sein. Im Fall von HTML-Seiten wird hierzu die Position des mit der Maus selektierten Bereichs in der HTML-Seite verwendet[109].

Der annotierte Bereich (anchor-id, anchor-area) wird so eindeutig über die URL der betrachteten HTML-Seite und die Position des mit der Maus selektierten Bereichs bestimmt.

Fall 2: Die Lernobjekte sind Teil einer größeren Datei

In diesem Fall sind mehrere Lernobjekte in einer größeren Datei aggregiert, wobei die Grenzen zwischen den einzelnen Lernobjekten nicht mehr notwendigerweise klar erkennbar sind. Im folgenden wir davon ausgegangen, dass mehrere Lernobjekte durch das in 6.4 beschriebene Verfahren zu einer größeren Postscript-Datei zusammengefasst wurden und dass die Stelle dieser Datei, die annotiert werden soll, mit der Maus selektiert wurde.

Um auch hier die Identifikation des zu annotierenden Lernobjekts zu erleichtern, wird das Verfahren aus 6.4 für den Fall, dass mehrere Lernobjekte zu einer größeren Datei aggregiert werden, so erweitert, dass zu Beginn und am Ende eines Lernobjekts dessen Identifikator in einem Kommentar in den erzeugten, präsentationsfertigen Lehrmaterialien vermerkt wird. So kann der Identifikator des zu annotierenden Lernobjekts ermittelt werden. Im folgenden wird ein Verfahren beschrieben, das es ermöglicht, den selektierten Teil von aus Lernobjekten aggregierten Lehrmaterialien auf Elemente des Kursbaums, aus dem die Lehrmaterialien erzeugt wurden, abzubilden. Die teilweise oder vollständig selektierten Lernobjekte werden im Kursbaum als selektiert markiert. Damit das Verfahren anwendbar ist, müssen folgende Voraussetzungen erfüllt sein:

- Die Lehrmaterialien sind direkt zugänglich. Liegt beispielsweise ein Kurs als Postscript-Datei vor, so muss der Zugriff auf den Quellcode der Lehrmaterialien in Postscript möglich sein. Nur so können die Grenzen zwischen den einzelnen Lernobjekten, die ja als Kommentare im Quellcode vorliegen, sicher erkannt werden. Dies ist ohne Probleme bei Lehrmaterialien, die als große (aus mehreren Lernobjekten aggregierte) HTML-Dateien oder Postscript-Dateien vorliegen, möglich.
- Der Kurs-Viewer muss so gestaltet sein, dass in ihn Plugins oder sonstige Erweiterungen, die auf den Quellcode der Lehrmaterialien zugreifen können, eingehängt werden können. Web-Browser beispielsweise bieten entsprechende Schnittstellen an, ebenso manche PDF-Viewer. Ghostview, ein Open-Source Postscript-Viewer, wiederum liegt im Quellcode vor und kann geeignet erweitert werden. Im folgenden wird davon ausgegangen, dass die von den Nutzern verwendeten Viewer mit einem entsprechenden Plugin ausgestattet sind.

Das Verfahren umfasst drei Schritte: Die Erkennung des selektierten, zu annotierenden Kursausschnitts, die Abbildung dieses Kursausschnitts auf den Kursbaum des Kurses und schließlich die Ablage der Annotation im Annotations-Repository.

---

[109] Aktuelle Web-Browser oder Dokument-Viewer (beispielsweise PDF- oder Postscript-Viewer) bieten Schnittstellen an, über die Plugins diese Informationen auslesen können.

Schritt 1: Erkennung des zu annotierenden Kursausschnitts

Die Selektion des zu annotierenden Kursausschnitts erfolgt über die Auswahl des Ausschnitts mittels Cursor oder Mauszeiger. Dabei wird üblicherweise der selektierte Bereich invertiert dargestellt, um den Nutzern eine Orientierung zu bieten. Dieses Vorgehen wird von vielen Anwendungen genutzt und ist den Nutzern geläufig; eine Einarbeitungszeit entfällt.

Bei vielen Medien ist diese Art der Auswahl problemlos möglich; Microsoft Word-Dokumente, PDF-Dokumente[110], HTML-Texte, etc., aber auch unstrukturierte Texte und Datenblöcke sind hier zu nennen.

Nach erfolgter Selektion können Anfang und Ende der Selektion im Quelltext der Lehrmaterialien ermittelt werden. Dies ist Grundlage für Schritt 2.

Schritt 2: Abbildung des zu annotierenden Ausschnitts auf einen Teilbaum des Kursbaums

Ausgehend von dem identifizierten, zu annotierenden Ausschnitt des Kurses kann das im Lehrmaterial-Viewer installierte Plugin nun eine Abbildung auf die Lernobjekte des Kurses vornehmen, um die Annotationen richtig zuordnen zu können. Hierzu wird der Anfang und das Ende des zu annotierenden Bereichs im Kursbaum wie folgt ermittelt:

1) Ermittlung des Anfangs des zu annotierenden Bereichs

Ausgehend vom Beginn der Selektion geht das Plugin im Lehrmaterial-Viewer so weit im Lehrmaterialquellcode zurück, bis es auf einen Kommentar stößt, in dem der Beginn eines Lernobjekts markiert wird. (Dies funktioniert unabhängig davon, ob die Selektion mit Text oder einem Multimediaelement wie einem Bild, etc. beginnt.) In diesem Kommentar ist auch der Identifikator des Lernobjekts vermerkt. Das Plugin ermittelt diesen Identifikator und speichert ihn in der Variablen selection-begin-id.

Als nächstes wird die Position der Selektion im gegenwärtigen Lernobjekt ermittelt. Hier muss unterschieden werden, ob die Selektion mit Text oder mit einem Multimediaelement beginnt:

Falls die Selektion mit Text beginnt, wird die Position der Selektion im Text ermittelt. Dies geschieht über die Anzahl der Zeichen. Diese Position speichert das Plugin in der Variablen selection-begin-position.

Falls die Selektion jedoch mit einem Multimediaelement beginnt, so wird davon ausgegangen, dass das gesamte Multimediaelement und damit das gesamte Lernobjekt selektiert ist; die Variable selection-begin-position wird mit 0 belegt.

2) Ermittlung des Endes des zu annotierenden Bereichs

Das Vorgehen zur Ermittlung des Endes der Selektion ist analog: Das Plugin geht ausgehend vom Ende des Selektion so lange im Lehrmaterialquellcode zurück, bis wieder ein Kommentar gefunden wird, der den Beginn des Lernobjekts markiert. Den in diesem Kommentar abgelegte Identifikator des Lernobjekts speichert das Plugin in der Variablen selection-end-id.

Als nächstes wird wieder die Position des Endes der Selektion im gegenwärtigen Lernobjekt ermittelt. Hier muss wieder unterschieden werden, ob die Selektion mit Text oder mit einem Multimediaelement endet:

[110] PDF erlaubt es, die Selektion von Dokumentausschnitten zu unterbinden. Hier wird davon ausgegangen, dass die Ersteller der PDF-Datei die Selektion von Dokumentausschnitten erlaubt haben.

Falls die Selektion mit Text endet, wird die Position der Selektion im Text ermittelt. Dies geschieht über die Anzahl der Zeichen. Diese Position speichert das Plugin in der Variablen selection-end-position.

Falls die Selektion mit einem Multimediaelement endet, so wird wieder davon ausgegangen, dass das gesamte Multimediaelement und damit das ganze Lernobjekt selektiert ist; das Plugin belegt die Variable selection-end-position mit 0.

3) Ermittlung des Teilbaums im Kursbaum, der der Selektion entspricht

Der einfachste Fall liegt vor, wenn selection-begin-id und selection-end-id gleich sind, wenn also die Selektion nur den Inhalt eines Lernobjekts umfasst. In diesem Fall muss kein Teilbaum des Kursbaums ermittelt werden, sondern die Parameter anchor-id und anchor-area, die ja den Anker einer Annotation markieren, können direkt gesetzt werden:

- anchor-id = selection-begin-id
- anchor-area = (selection-begin-position, selection-end-position)

In allen anderen Fällen (selection-begin-id ≠ selection-end-id) wird folgendermaßen vorgegangen: Die beiden Lernobjekte mit den Identifikatoren selection-begin-id und selection-end-id sind Blätter im Kursbaum. Alle Blätter (Atome) sowie alle inneren Knoten (Module), die zwischen diesen Blättern im Kursbaum liegen, sind mit der Selektion ebenfalls markiert, vgl. Abbildung 7-11.

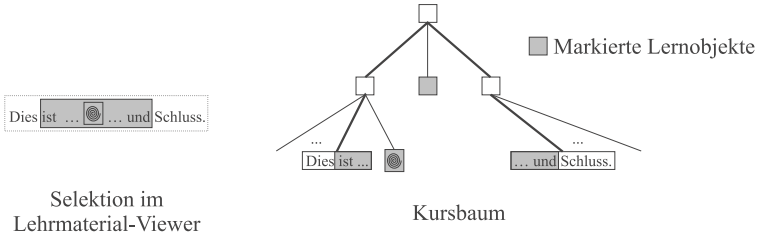


Abbildung 7-11: Beispiel für einen markierten Teilbaum

Für jeden dieser Knoten wird in Schritt 3 (vgl. unten) nun explizit eine Annotation eingetragen, wobei anchor-id mit dem Identifikator des jeweiligen Knotens belegt ist und anchor-area leer bleibt[111]. Außerdem wird für den Knoten, in dem der Beginn der Selektion liegt, eine Annotation mit

- anchor-id = selection-begin-id
- anchor-area = (selection-begin-position, 0)

gesetzt; für den Knoten, in dem das Ende der Selektion liegt, wird in der Annotation analog

- anchor-id = selection-end-id
- anchor-area = (1, selection-end-position)

gesetzt.

[111] Zur Erinnerung: Bleibt anchor-area leer, so betrifft die Annotation das gesamte Lernobjekt. Details hierzu sind in 7.2.1.3 zu finden.

Schritt 3: Ablage der Annotation im Annotations-Repository

Je nach Gestalt der Annotation werden nun verschiedene Aktionen durchgeführt: Falls der Nutzer eine Anmerkung in Freitext verfasst hat, nutzt das Plugin im Lehrmaterial-Viewer den Service putAnnotationText() des Annotations-Repositories (vgl. 6.3.3.4). Dieser Service wird für jedes Lernobjekt, das in Schritt 2 markiert wurde, aufgerufen, so dass alle Lernobjekte, die selektiert wurden, mit entsprechenden Annotationen versehen werden.

Falls der Nutzer einen Verweis auf ein bereits existierendes Atom oder Modul als Annotation definiert hat, nutzt das Plugin für jedes in Schritt 2 markierte Lernobjekt den Service putAnnotation() des Annotations-Repositories.

Die hier aufgezeigten Schritte werden in einer Bibliothek zusammengefasst und stehen den Entwicklern von Plugins, Annotationseditoren und anderen Tools zur Nutzung bereit. So müssen diese Schritte nicht immer aufs Neue implementiert werden, sondern stehen „out of the box“ zur Verfügung.

7.4.5 Umsetzung des Lernobjekt-Scanners

7.4.5.1 Überblick über die betrachteten Aspekte

Durch viele Arbeiten mit Lernobjekten und Annotationen können Annotationen bzw. die Assoziationen zwischen annotierten Elementen (Lernobjekte, Kurse oder selbst wieder Annotationen) und Annotationen inkonsistent werden. Beispiele hierfür sind Annotationen, deren Lernobjekte, auf die sie sich beziehen, gelöscht wurden, oder Lernobjekte, die geändert wurden, nachdem sie annotiert wurden (vgl. 6.3.5.2 für Details). Um solche Inkonsistenzen erkennen und beheben zu können, müssen Lernobjekte und Annotationen sowie die Assoziationen zwischen den annotierten Elementen und den Annotationen regelmäßig überprüft werden. Dies ist Aufgabe des Lernobjekt-Scanners.

Im folgenden wird beschrieben, wie die Konsistenzprüfung technisch abläuft. Die Problemfälle, auf die der Lernobjekt-Scanner achten muss, wurden bereits in 6.3.5.2 ausführlich dargelegt.

7.4.5.2 Durchführung der Konsistenzprüfung

Beschreibung des Algorithmus

Grundlage für die Prüfung, ob die Annotation *A* zu einem Lernobjekt *L* im Rahmen eines Kurses *K* konsistent ist, ist das Prädikat consistent(*A, L, K*) über den Parametern von *A* und *L*. Dieses Prädikat wurde detailliert in 6.3.5.2 vorgestellt.

Um ein Lernobjekt *L* und eine Annotation *A* im Rahmen eines Kurses *K* auf Konsistenz zu prüfen, ist consistent(*A, L, K*) zu berechnen. Dieser Test muss paarweise für alle Annotationen und alle Lernobjekte durchgeführt werden, die mittels Assoziationen vom Typ „annotiert“ verbunden sind. Bei *n* Lernobjekten, die mit *m* Annotationen verbunden sind, hat dieses Vorgehen die sehr ungünstige Komplexität *O*(*nm*).

Hier können Hash-Tabellen und Bags helfen:

Hash-Tabelle *Annotations*(*objID*, {*anID*}) mit *objID* als Schlüssel ,
Bag *Status*(*Changed)* mit einer Menge *Changed* von Identifikatoren.

ObjID ist der Identifikator eines Lernobjekts bzw. einer Annotation, {*anID*} ist die Menge der Identifikatoren aller Annotationen, die zum Element *objID* verfasst wurden, und *mark* ist ein Flag, das markiert, ob das Lernobjekt bzw. die Annotation *objID* geändert wurde.

Jedes mal wenn das Lernobjekt bzw. die Annotation mit dem Identifikator *objID* geändert wird, werden die Identifikatoren der Annotationen, die bereits zu *objID* verfasst wurden, aus *Annotations* ausgelesen. Dies ist mit $O(1)$ möglich. Man erhält eine Menge {*anID*} von Identifikatoren. Anschließend werden alle IDs in {*anID*} zu *Status*(*Changed*) hinzugenommen. Bei geeigneter Implementierung des Bags ist dies mit $O(|\{anID\}|)$ möglich, wobei im allgemeinen $|\{anID\}| << n+m$ sein wird.

Status(*Changed*) enthält zu einem Zeitpunkt *t* nun die Identifikatoren aller Annotationen, die potenziell inkonsistent sind. Ein Abrufen der Identifikatoren aus dem Bag hat $O(1)$. Eine Auswertung dieser Identifikatoren – beispielsweise eine Benachrichtigung der Verfasser der Annotationen – ist mit linearem Aufwand $O(n)$ möglich, so dass für das gesamte Verfahren eine Komplexität von $O(2n+m)$ erreicht wird.

Das Verfahren besteht aus einer Funktion, die bei Änderungen aufgerufen wird, und einer Funktion, die die Behandlung potenziell inkonsistenter Annotationen anstößt:

Beim Ändern oder Löschen eines Lernobjekts oder einer Annotation mit dem Identifikator *objID* wird folgende Funktion ausgeführt:

```
void elementChanged(objID)
{
    {Annotations} := Annotations.get(objID);
    foreach ID in {Annotations} do
    {
        Status(Changed).put(ID);
    }
}
```

Diese Funktion des Lernobjekt-Scanners wird von der Lernobjektzugriffsschicht oder dem Annotations-Repository, in dem Element objID abgelegt ist, aufgerufen.

Eine Auswertung aller potenziell inkonsistenter Annotationen im Lernobjekt-Scanner sieht folgendermaßen aus:

```
void processInconsistentAnnotations(void)
{
    {InconsistentAnnotationIDs} := Status(Changed).getAll();
    foreach ID in {InconsistentAnnotationIDs} do
    {
        processAnnotation(ID);
    }
}
```

ProcessAnnotation() ist hierbei eine Funktion, die eine potenziell inkonsistente Annotation mit dem Identifikator *ID* bearbeitet. ProcessAnnotation() könnte beispielsweise den Verfasser der Annotation benachrichtigen.

Eine Veränderung der Datenstrukturen *Annotations* und *Status*(*Changed*) im Rahmen der Auswertung der potenziell inkonsistenten Annotationen in processInconsistentAnnotations() – beispielsweise indem bereits ausgewertete Annotationen aus *Annotations* entfernt werden oder für diese Annotationen *Status(Changed)* geändert wird – ist nicht sinnvoll, da allein durch die Bearbeitung der in diesen Datenstrukturen abgelegten Annotationen deren möglicherweise inkonsistenter Zustand nicht geändert wird. Der Zustand der Annotationen wird erst durch später erfolgende Nutzeraktionen – möglicherweise – geändert, nicht jedoch allein durch die Ausführung von processAnnotation().

Dieses Verfahren teilt die Behandlung potenziell inkonsistenter Annotationen in zwei Phasen, die zeitlich getrennt ablaufen: Die erste Phase findet beim Update oder beim Löschen von Lernobjekten oder Annotationen statt. In dieser Phase findet eine Interaktion mit Lernobjektzugriffsschichten und / oder dem Annotations-Repository statt. Hierbei werden die Datenstrukturen *Annotations* und *Status*(*Changed*) aufgebaut und befüllt. In der zweiten Phase werden alle potenziell inkonsistenten Annotationen bearbeitet. Die konkrete Realisierung findet hierbei in einer Callback-Funktion processAnnotation(*ID*) im Lernobjekt-Scanner statt. Diese Funktion kann mittels des Identifikators *ID* die fragliche Annotation vom Annotations-Repository anfragen und dann weiter bearbeiten, beispielsweise indem der Autor der Annotation ermittelt und benachrichtigt wird.

Zeitpunkt der Konsistenzprüfung

Die erste Phase des gerade beschriebenen Verfahrens läuft parallel zur Entwicklung der Lernobjekte; eine zeitliche Taktung ist hier nicht sinnvoll. Die zweite Phase, die Verarbeitung der potenziell inkonsistenten Annotationen, ist vollständig von der ersten Phase entkoppelt und kann regelmäßig in bestimmten Zeitabständen oder anhand bestimmter Last- oder Mengenkriterien durchgeführt werden. Im folgenden werden verschiedene Modelle kurz vorgestellt:

Alternative 1: Bearbeitung potenziell inkonsistenter Annotationen in regelmäßigen Zeitabständen
Bei diesem Modell wird die Bearbeitung der potenziell inkonsistenten Annotationen in regelmäßigen Zeitabständen durchgeführt, beispielsweise jede Nacht zu einem festgesetzten Zeitpunkt. Dieses Vorgehen hat den Vorteil, dass die fraglichen Annotationen zu garantierten Zeitpunkten bearbeitet werden und nicht von äußeren Einflüssen wie der CPU-Last oder der Änderungshäufigkeit von Lehrmaterialkomponenten abhängen. Es kann allerdings sein, dass bei einer hohen Änderungshäufigkeit eine große Zahl von Annotationen bearbeitet werden muss; der Aufwand ist von Durchlauf zu Durchlauf unterschiedlich.

Alternative 2: Bearbeitung potenziell inkonsistenter Annotationen entsprechend der CPU-Last
Der Lernobjekt-Scanner läuft auf den Rechnern, auf denen auch der Lernobjekt-Service ausgeführt wird. Bei diesem Modell wird die Bearbeitung der fraglichen Annotationen durchgeführt, wenn die CPU-Last dieser Rechner unter einen bestimmten Schwellenwert sinkt. So können die Rechner optimal genutzt werden; Leerlauf wird weitgehend vermieden. Allerdings ist bei einer hohen Beanspruchung der Rechner durch den Lernobjekt-Service eine regelmäßige Abarbeitung der zu prüfenden Annotationen nicht gewährleistet.

Alternative 3: Bearbeitung potenziell inkonsistenter Annotationen entsprechend ihrer Anzahl
Eine weitere Möglichkeit besteht darin, die potenziell inkonsistenten Annotationen erst dann zu bearbeiten, wenn von ihnen eine bestimmte Mindestanzahl vorliegt. So kann zwar vermieden werden, dass eventuell aufwändige Bearbeitungen für nur wenige Annotationen angestoßen werden, aber andererseits ist die Gefahr einer Aushungerung gegeben, wenn die Änderungsrate der Lernobjekte sehr gering ist.

Ergebnis
Es ist am geschicktesten, mehrere Ansätze zu kombinieren. Alternativen 1 und 3 lassen sich wie folgt kombinieren: Die potenziell inkonsistenten Annotationen werden erst dann bearbeitet, wenn eine bestimmte, konfigurierbare Mindestanzahl vorliegt, spätestens jedoch zu einem festgelegten Zeitpunkt einmal am Tag. So kann vermieden werden, dass durch eine hohe Anzahl von zu bearbeitenden Annotationen die Bearbeitungszeit über Gebühr lang wird

(Nachteil von Alternative 1), und andererseits ist eine maximale Zeit bis zur Bearbeitung sichergestellt (keine Gefahr einer Aushungerung im Gegensatz zu Alternative 3).

7.4.6 Awareness-Funktionalität für Annotationen

7.4.6.1 Überblick über die betrachteten Aspekte

Beim Verfassen und bei der Bearbeitung von Annotationen sowie bei der Arbeit mit Lernobjekten treten immer wieder Situationen ein, in denen bestimmte Nutzer und Designer über bestimmte Umstände informiert werden müssen. Ein Beispiel hierfür sind Inkonsistenzen zwischen annotierten Elementen (Lernobjekte, Kurse und Annotationen) und ihren Annotationen.

Grundsätzlich sind verschiedene Arten der Benachrichtigung möglich: In 6.3.6.2 wurden aktive und passive Benachrichtigungen genannt. Bei den aktiven Benachrichtigungen ist weiterhin zu berücksichtigen, wann und auf welche Weise ein Nutzer oder Designer benachrichtigt werden will.

All dies sind Aufgaben des Notifikations-Service. Der Notifikations-Service nimmt Ereignisdaten von verschiedenen Komponenten entgegen und informiert die Nutzer und Designer entsprechend ihren Wünschen und Vorgaben im Nutzerprofil. Der Notifikations-Service sorgt damit für die nötige Awareness bei den Nutzern und den Designern, wenn neue Annotationen angelegt werden, sich Unstimmigkeiten bei den Annotationen ergeben haben könnten oder sonstige Ereignisse auftreten.

7.4.6.2 Modellierung der Benachrichtigung

Ziel des Notifikations-Service ist es, die Verfasser von Lernobjekten und Annotationen sowie andere Interessenten über Änderungen in den von ihnen genutzten Lernobjekten und Annotationen zu informieren. Der Notifikations-Service nimmt Ereignisdaten von verschiedenen Komponenten der Annotationsinfrastruktur entgegen. Unter diesen Ereignisdaten sind auch viele, die mit Annotationen selbst nichts zu tun haben, beispielsweise wenn ein neues Lernobjekt erzeugt wird. Diese Ereignisdaten sollen hier nicht weiter betrachtet werden; hier soll der Schwerpunkt auf den Ereignisdaten liegen, die die Nutzer und Designer auf Ereignisse, die im Zusammenhang mit Annotationen stehen, hinweist. In diesem Sinne ist für die Nutzer somit vor allem von Interesse, auf welche ihrer Annotationen (und damit auf welches Lernobjekt oder welche Annotation) sich die Ereignisdaten beziehen und welche Aktion zu welchem Zeitpunkt durchgeführt wurde.

Für jedes Ereignis – insbesondere für jede potenzielle Inkonsistenz, die der Lernobjekt-Scanner aufgedeckt hat – erhält der Notifikations-Service folgenden Datenblock:

```
DATA
{
  action = <Aktion, über die informiert werden soll>
  time = <Zeitpunkt dieser Aktion>
  obj-id = <Identifikator des von der Aktion betroffenen
    Lernobjekts>
  annotation-id = <Identifikator der betroffenen Annotation>
  author-id = <Identifikator des Verfassers der Annotation>
  user-ids = <Menge von Identifikatoren der zu informierenden
    Nutzer>[112]
}
```

[112] Welche Nutzer bei Inkonsistenzen genau benachrichtigt werden, ist bei der Besprechung des Lernobjekt-Scanners in 6.3.5.1 erläutert.

Diese Datenblöcke werden an den Notifikations-Service durch Nutzung des Dienstes notify() (vgl. Tabelle 6.6) übergeben.

7.4.6.3 Architektur des Notifikations-Service

Der Notifikations-Service besteht aus vier Komponenten: Ein Nachrichtenpuffer, der die von anderen Komponenten übermittelten Ereignisdaten speichert, ein Nutzerprofil-Repository, das Angaben darüber speichert, wie die einzelnen Nutzer über Ereignisse informiert werden wollen, ein Report-Generator, der die Informationen für die Nutzer entsprechend ihren Wünschen zusammenstellt, und einen Nachrichten-Publisher, der die Verteilung dieser Informationen an die Nutzer übernimmt. Im folgenden werden diese Komponenten sowie ihr Zusammenspiel im Detail vorgestellt.

Nutzerprofil-Repository

Das Nutzerprofil-Repository hält Einstellungen der Nutzer vor, wie sie über Änderungen informiert werden wollen. Ein Nutzerprofil, wie es für diese Arbeit benötigt wird, enthält folgende Daten:

- Aktive Benachrichtigung:

  Diese Information gibt an, ob der Nutzer auch passiv benachrichtigt werden will. Wünscht der Nutzer eine aktive Benachrichtigung, so ist dieser Wert gesetzt. Wünscht der Nutzer hingegen eine passive Benachrichtigung, so ist dieser Wert nicht gesetzt.

- Aktive Benachrichtigung: Zeitpunkt der Benachrichtigung je Nachrichtenkanal:

  Hier ist der Zeitpunkt abgelegt, zu denen der Nutzer über einen bestimmten Nachrichtenkanal (E-Mail, Instant Message, etc.) über Ereignisse informiert werden will, wenn er aktive Benachrichtigung gewählt hat. Generell sind hier zwei Möglichkeiten denkbar: Eine unmittelbare Information, sobald Ereignisdaten beim Notifikations-Service eingehen, oder die Information in regelmäßigen Abständen zu einem bestimmten Zeitpunkt, beispielsweise jeden Tag um 10.00 Uhr. Wünscht der Nutzer über einen Nachrichtenkanal eine unmittelbare Benachrichtigung, so ist dieser Wert nicht gesetzt.

  Hier beschränkt sich diese Arbeit auf zwei Modelle: Die unmittelbare Information, sobald Ereignisdaten verfügbar sind, sowie eine Benachrichtigung einmal am Tag.

- Aktive Benachrichtigung: Kontaktinformationen des Nutzers je Nachrichtenkanal:

  Für eine aktive Benachrichtigung muss bekannt sein, wie der Nutzer über einen bestimmten Nachrichtenkanal zu erreichen ist. Je nach unterstützter Benachrichtigung müssen hier verschiedene Informationen angegeben sein, beispielsweise Mobiltelefonnummern für Benachrichtigungen mittels SMS, verschiedene Instant Messenger-IDs (beispielsweise für Yahoo! Messenger, MSN Messenger, Lotus Sametime, etc.) für kurze Instant Messages bei Änderungen, etc. Mindestens wird jedoch eine E-Mailadresse gefordert.

Ein Nutzerprofil für einen Nutzer wird somit durch folgendem Datenblock realisiert:

```
DATA
{
   active-notification = [true | false]
   SET
   {
     notification-time = <Zeitpunkt>
```

```
      contact-data = STRING
    }
}
```

Der Datenblock contact-data enthält Informationen, wie der Nutzer über einen bestimmten Nachrichtenkanal zu erreichen ist. Dies wird durch eine Zeichenkette festgelegt, die eine URL mit Kontaktdaten, zu verwendenden Protokollen, etc. enthält. Diese URL könnte beispielsweise folgende Gestalt haben:

- Kontakt per E-Mail: email://zhuang@cs.tum.edu
- Instant Message über den Yahoo! Messenger: yahoo://weilunzhuang
- SMS an ein Mobiltelefon: sms://017212345

Wird für einen Nachrichtenkanal eine unmittelbare Benachrichtigung gewünscht, so wird für diesen Nachrichtenkanal notification-time nicht gesetzt.

Die Nutzerprofile sind in einer Tabelle mit folgendem Schema abgelegt:

```
(user-ID, active-Notification, notification-Time, contact-Data)
```

Dieses Schema bildet den oben genannten Datenblock nahezu unmittelbar ab. User-ID ist dabei die Kennung des Nutzers. Active-Notification ist ein Flag, das für den Fall, dass der Nutzer eine aktive Benachrichtigung wünscht, auf true gesetzt ist; andernfalls ist es auf false gesetzt. (Ist active-Notification auf false gesetzt, impliziert dies eine passive Benachrichtigung.) Falls der Nutzer keine sofortige Benachrichtigung bei einem Ereignis wünscht, wird die Spalte notification-Time auf den Zeitpunkt gesetzt, zu dem der Nutzer benachrichtigt werden möchte. Wünscht der Nutzer eine sofortige Benachrichtigung, bleibt der Wert dieser Spalte undefiniert (NULL). Contact-Data schließlich enthält die Adresse, an die die Benachrichtigung geschickt werden soll, sowie den Nachrichtenkanal, der hierzu verwendet werden soll. Der Wert dieser Spalte wird von den Plugins ausgewertet, die die Benachrichtigungen verschicken. Werte in dieser Spalte haben immer folgende Gestalt:

```
<Kommunikationskanal>//<Adresse>
```

Soll eine Benachrichtigung beispielsweise mittels E-Mail an zhuang@cs.tum.edu verschickt werden, muss die Kontaktadresse folgendermaßen aussehen:

```
email://zhuang@cs.tum.edu
```

Das Nutzerprofil-Repository stellt nun einen Datenspeicher für Nutzerprofile sowie Dienste für Zugriff und Auswertung der Nutzerprofile bereit. Diese Dienste werden allerdings nur intern für die Komponenten des Notifikations-Service bereitgestellt; eine Bereitstellung nach außen (für Programme außerhalb der Annotationsinfrastrukur) findet nicht statt. Um den Nutzern eine Pflege ihrer Nutzerprofile zu ermöglichen, wird eine Web-basierte Nutzerschnittstelle angeboten, in der die jeweiligen Benachrichtigungseinstellungen überprüft und geändert werden können. In Abbildung 7-12 sind die Benachrichtigungseinstellungen für den Nutzer „Weilun“ zu sehen: Es sind Benachrichtigungen mittels E-Mail und über den Yahoo! Messenger definiert: Nutzer „Weilun“ bekommt einmal am Tag eine E-Mail, in der alle Ereignisse hinsichtlich Lernobjekten und Annotationen zusammengefasst sind. Des weiteren wird „Weilun“ jedes Mal, wenn ein auf Annotationen bezogenes Ereignis eintritt, sofort mit einer Instant Message über den Yahoo! Messenger informiert.

Welche Benachrichtigungsoptionen den Nutzern angeboten werden, hängt davon ab, welche Plugins zur Anbindung von Kommunikationsmitteln (E-Mail, verschiedene Instant Messenger, etc.) gegenwärtig in den Notifikations-Service eingebunden sind. So ist sichergestellt, dass die Nutzer keine Optionen angeboten bekommen, die nicht erfüllt werden können.

Es ist auch möglich, bereits existierende externe Nutzerprofildatenbanken anzubinden. Dies geschieht analog der Anbindung verschiedener Rechtesysteme im Lernobjekt-Repository (vgl. 7.4.1.4) wieder über einen Plugin-Mechanismus. Ein Plugin für eine bestehende Nutzerprofildatenbank muss dabei sowohl die technische Anbindung leisten (Bereitstellung einer einheitlichen Schnittstelle für das Plugin nutzende Komponenten, Verbindungsaufbau und Datenaustausch mit der Nutzerprofildatenbank), als auch eine Konvertierung der Nutzerprofildaten in das Datenformat, das der Notifikations-Service verarbeiten kann. Dies soll in dieser Arbeit jedoch nicht weiter verfolgt werden, da die Problematik der Anbindung von Nutzerprofildatenbanken hier nur am Rande von Interesse ist.

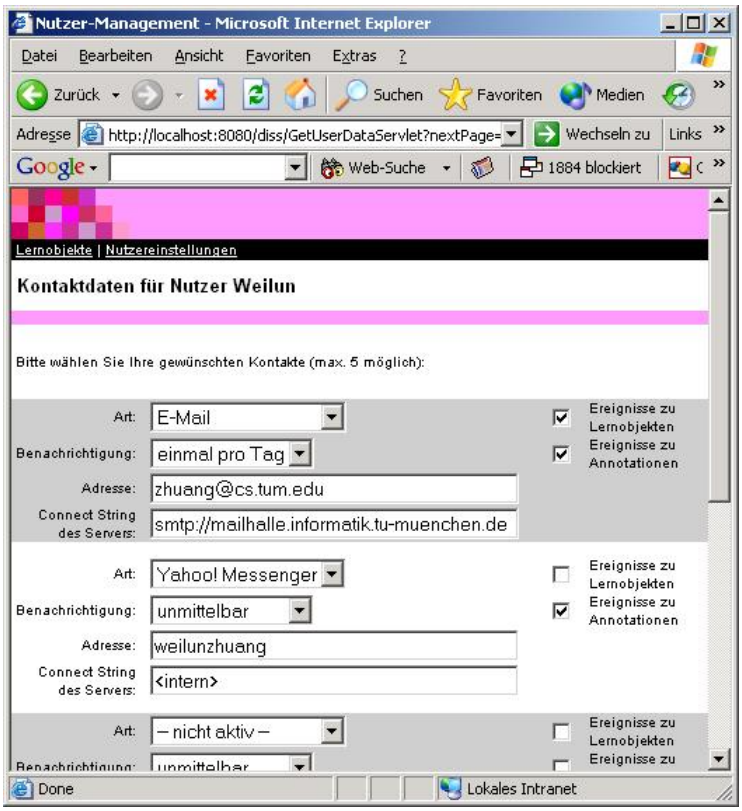


Abbildung 7-12: Benachrichtigungsoptionen für Nutzer "Weilun" (Screenshot)

Nachrichtenpuffer

Der Nachrichtenpuffer nimmt Ereignisdaten von anderen Komponenten (im Zusammenhang mit Annotationen ist hier vor allem der Lernobjekt-Scanner interessant) entgegen. Für jedes Ereignis sind eigene Ereignisdaten nötig. Hierzu stellt er die in Tabelle 6-6 genannten Dienste als Web Services zur Verfügung und bildet so die Schnittstelle des Notifikations-Service nach außen.

Im ersten Schritt prüft der Nachrichtenpuffer, ob der fragliche Nutzer, den ein Ereignis betrifft bzw. der an einer Unterrichtung bezüglich eines Ereignisses interessiert ist, unmittelbar über das Ereignis informiert werden will oder ob eine spätere Benachrichtigung gewünscht wird. Hierzu entnimmt der Nachrichtenpuffer der Nachricht den Identifikator der Nutzer der potenziell inkonsistenten Annotation bzw. des Lernobjekts und prüft im Nutzerprofil, welcher Zeitpunkt der Benachrichtigung gewünscht wird. Falls eine sofortige Information gewünscht wird, wird die Nachricht unmittelbar an den Report-Generator übergeben; andern-

falls werden die Ereignisdaten für die spätere Weiterleitung in einer internen Datenbank zwischengespeichert. Abbildung 7-13 fasst den Ablauf zusammen.

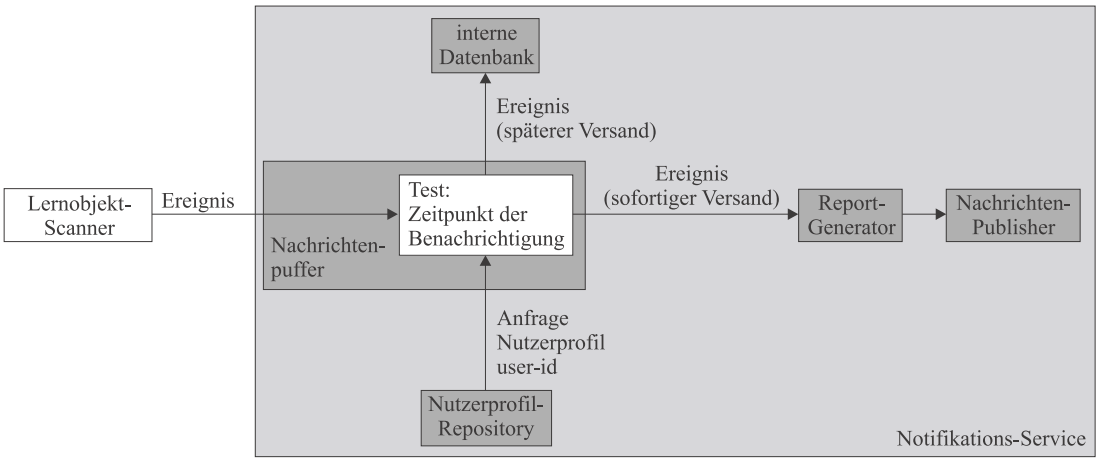


Abbildung 7-13: Funktionsweise des Nachrichtenpuffers

Report-Generator

Der Report-Generator hat die Aufgabe, die Ereignisdaten, die von anderen Komponenten geliefert und vom Nachrichtenpuffer entgegengenommen werden, zu aggregieren und so aufzubereiten, dass sie für die Nutzer gut lesbar sind. Dies wird durch einen Template-Mechanismus realisiert.

Es gibt zwei Templates: Ein Template für die Weiterleitung eines einzelnen Ereignisses sowie ein Template für die Weiterleitung von mehreren Ereignissen in Form einer Liste. Jedes dieser Templates enthält Platzhalter, in die die konkreten Werte aus den Ereignissen eingesetzt werden. Wichtig ist hier zu beachten, dass die Templates keine konkreten Layout-Informationen wie Schriftarten, Farben, etc. beinhalten. Da das Layout bestimmte Rahmenbedingungen, die durch die Versandart der Nachricht definiert werden, einhalten muss, ist dies im Report-Generator noch nicht möglich. Das exakte Layout der Nachrichten erfolgt erst im Nachrichten-Publisher.

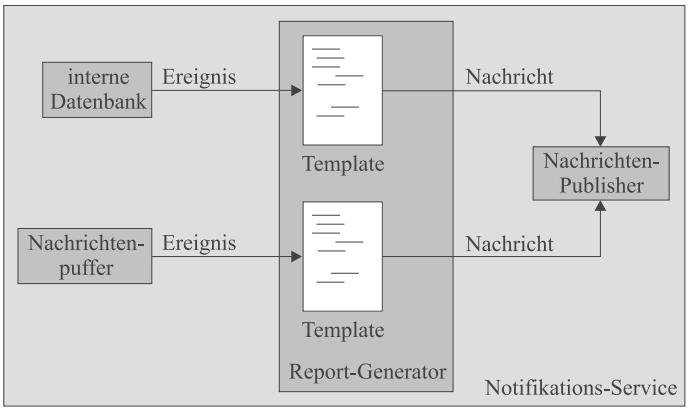


Abbildung 7-14: Funktionsweise des Report-Generators

Der Report-Generator nimmt Ereignisdaten von zwei verschiedenen Quellen entgegen: Ereignisdaten, für die sofort ein Report erstellt werden soll, werden vom Nachrichtenpuffer geliefert. Diese Ereignisdaten werden dann unmittelbar in das Template für die Versendung einer einzelnen Nachricht eingesetzt und an den Nachrichten-Publisher weitergeleitet, so

dass eine sofortige Information der Nutzer eingeleitet werden kann. Daneben werden einmal am Tag auch alle Ereignisse, die in der internen Datenbank für einen späteren Versand zwischengespeichert wurden, an den Report-Generator als Batch übergeben. Die im Report-Generator aus den Ereignisdaten erstellten Nachrichten werden dann im Template für mehrere Nachrichten zusammengefasst und an den Nachrichten-Publisher weitergeleitet. Diese beiden Arten der Nachrichtenverarbeitung sind in Abbildung 7-14 illustriert.

Nachrichten-Publisher

Der Nachrichten-Publisher hat die Aufgabe, Benachrichtigungen an die fraglichen Nutzer und Designer weiterzuleiten. Die Daten kommen hierbei immer vom Report-Generator.

Es kann sein, dass zu bestimmten Zeiten Nachrichten von der internen Datenbank als Batch geliefert werden und gleichzeitig andere Nachrichten vom Nachrichtenpuffer zum sofortigen Versand übergeben werden. Letztere Nachrichten haben eine höhere Priorität als Nachrichten, die als Batch ausgeliefert werden, so dass die unmittelbare Auslieferung der Informationen an die Nutzer, die dies wünschen, sichergestellt ist.

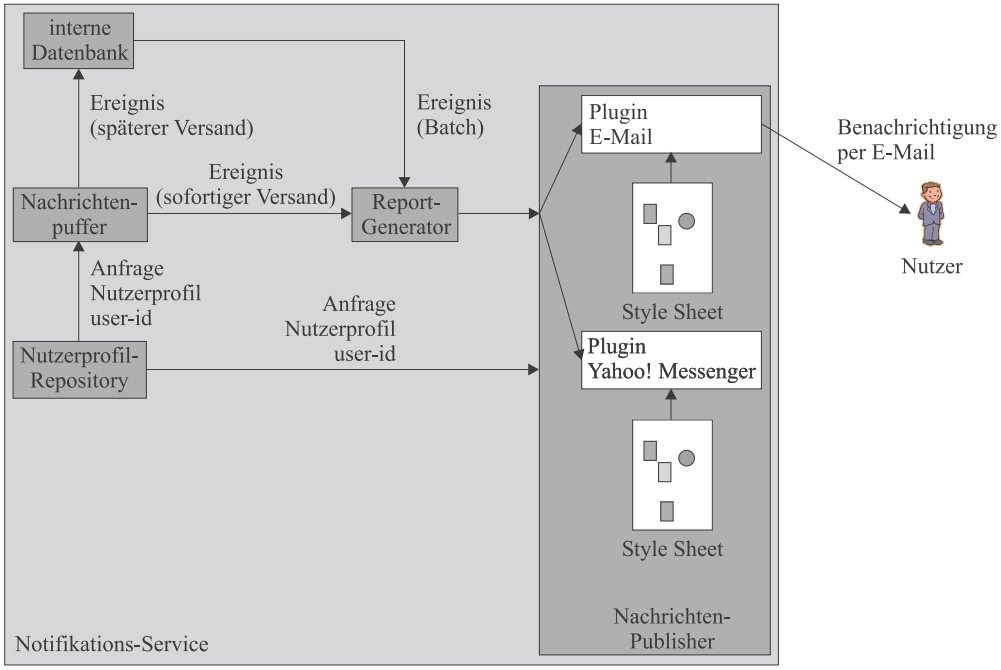


Abbildung 7-15: Funktionsweise des Nachrichten-Publishers

Der Nachrichten-Publisher erlaubt mehrere Arten der Benachrichtigung, beispielsweise mittels E-Mail, Instant Messages, etc. Für jede Art der Benachrichtigung ist im Nachrichten-Publisher ein Plugin installiert, das die Nachrichten, die vom Report-Generator bereits mit Templates vorbereitet wurden, anhand von Style Sheets in das endgültige Präsentationsformat überführt und dann über das jeweilige Protokoll an den Nutzer sendet (vgl. Abbildung 7-15). Soll der Nutzer beispielsweise per E-Mail benachrichtigt werden, so werden die Nachrichten vom Report-Generator an das E-Mail-Plugin übergeben und von diesem, konvertiert in ein passendes Format mit ansprechendem Layout, als E-Mail an den Nutzer geschickt. Soll der Nutzer per Instant Message informiert werden, so werden die Nachrichten vom Report-Generator an das Plugin des entsprechenden Instant Messengers, beispielsweise das Plugin für den MSN Messenger oder den Yahoo! Messenger, übergeben, dort entsprechend den Rahmenbedingungen des jeweiligen Instant Messengers mittels Style Sheets formatiert und dann versendet.

Ein Plugin zum Versand von Benachrichtigungen ist eine Java-Klasse, die folgendes Interface implementiert:

```
public interface NotificationSenderInterface
{
  public void init(Properties configData);
  public boolean notifyUser(String userID, Object notification);
}
```

Das Plugin wird mittels init() konfiguriert. Für ein Plugin, das den Versand mittels E-Mail realisiert, würden im Datenblock configData beispielsweise die IP-Adresse des SMTP-Servers sowie Nutzername und Passwort für die Authentifizierung beim SMTP-Server enthalten sein. Des weiteren müssen Style Sheets angegeben werden, die das Layout der Nachrichten, wie sie jeweils über die verschiedenen Kommunikationsmedien versandt werden sollen, beschreiben.

Die Formatierung der Nachricht sowie deren Versand erfolgt mittels notifyUser(). In dieser Methode wird das konkrete Style Sheet benannt, mit dem die Nachricht im endgültigen Präsentationsformat formatiert werden soll, denn die Wahl des Style Sheets ist abhängig vom Kommunikationsmedium, über das die Nachricht versandt werden soll: Eine Instant Message oder eine SMS haben völlig andere Anforderungen an Layout und Formatierung als beispielsweise eine E-Mail. Im Prototypen werden Benachrichtigungen nur per E-Mail versandt. Ist die Nachricht schließlich ordnungsgemäß formatiert[113], so wird sie an den Nutzer versandt. Die hierzu nötigen Informationen wurden dem Plugin bei dessen Initialisierung mittels init() mitgeteilt (vgl. oben).

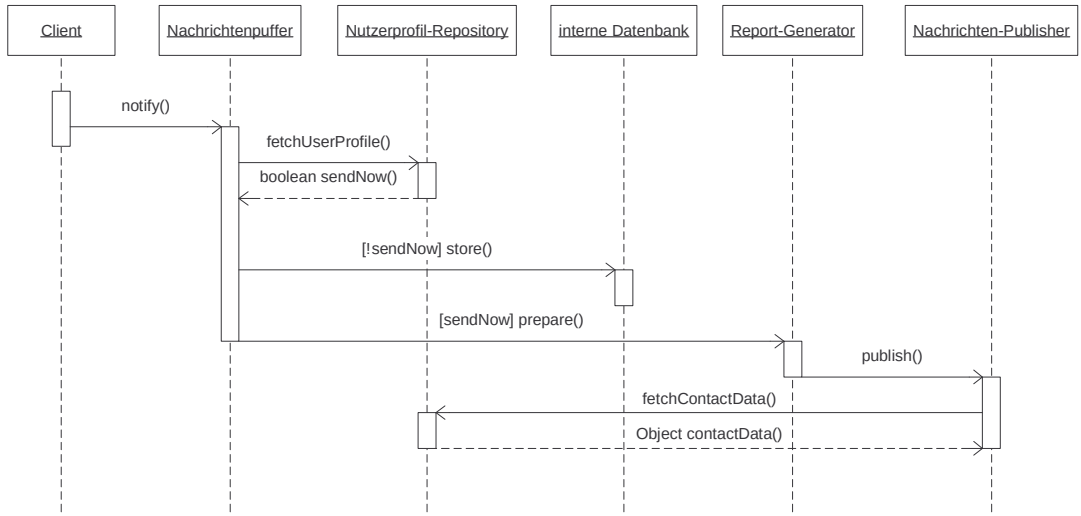


Abbildung 7-16: Zusammenspiel der Komponenten des Notifikations-Service

Die Auswahl, über welchen Nachrichtenkanal der Nutzer informiert werden soll, wird entsprechend seines Nutzerprofils vorgenommen; ebenso werden auch die jeweiligen Kontaktdaten des Nutzers aus seinem Nutzerprofil ermittelt. Hierzu führt der Nachrichten-Publisher eine Anfrage an das Nutzerprofil-Repository durch.

[113] Dies kann beispielsweise mittels Apache Xalan [XALAN] geschehen.

Zusammenfassung: Funktionsweise des Notifikations-Service

In den letzten Abschnitten wurden die einzelnen Komponenten des Notifikations-Service vorgestellt. Abbildung 7-16 zeigt das Zusammenspiel dieser Komponenten beim Versenden einer Nachricht mittels notify().

7.5 Zusammenfassung

In diesem Kapitel wurde die prototypische Realisierung der Annotationsinfrastruktur, die konzeptionell bereits in Kapitel 6 besprochen wurde, beschrieben. Wesentlicher Einflussfaktor war dabei die Umgebung, in der die einzelnen Komponenten ablaufen und interagieren müssen. Neben Überlegungen der technischen Umsetzung mussten hierbei immer auch organisatorische Aspekte berücksichtigt werden. Dies war notwendig, da nicht nur nicht-kommerziell arbeitende Organisationen, wie beispielsweise Universitäten, Lehrmaterialien erstellen, sondern auch Unternehmen an der Erstellung der Lehrmaterialien beteiligt werden sollen, um eine möglichst breite Basis an verfügbaren Lernobjekten zu erreichen. Gerade im kommerziellen Umfeld spielen Aspekte des Zugriffsschutzes und Lizenzierungsmodelle eine essentielle Rolle.

Zuerst wurden die Grundlagen der Realisierung von Lernobjekten und Annotationen beleuchtet. Dies betraf die Gestaltung der Identifikatoren für Lernobjekte und Annotationen so, dass sie weltweit eindeutig sind. Des weiteren wurden Parameter von Lernobjekten, Annotationen und Assoziationen vorgestellt, die vor allem für das korrekte Funktionieren der Annotationsinfrastruktur von Bedeutung sind. Und schließlich wurde gezeigt, wie Ausschnitte von Lernobjekten mittels des Parametertupels (anchor-id, anchor-area) eindeutig adressiert werden und so als Bezugspunkt für Annotationen dienen.

Anschließend wurde beschrieben, basierend auf welcher Kommunikationstechnologie die einzelnen Komponenten der Annotationsinfrastruktur interagieren. Da von einer losen Kopplung der einzelnen Komponenten ausgegangen wurde und eine gewisse Flexibilität in der Realisierung (Programmiersprache, Ablaufumgebung wie Betriebssystem, Hardware, etc.) sichergestellt werden sollte, wurden Web Services als Kommunikationsmittel ausgewählt. Der einheitliche Transport von Lernobjekten und Annotationen unabhängig von ihrer konkreten Gestalt geschieht mittels Wrappern.

Im Hauptteil dieses Kapitels wurden schließlich die Komponenten im Detail vorgestellt, die die Annotationsinfrastruktur ausmachen: Die Speicherung der Lernobjekte findet in einem Lernobjekt-Repository statt. Das Lernobjekt-Repository besteht aus Datenspeichern, die die Lernobjekte verwalten, Lernobjektzugriffsschichten, die einen einheitlichen Zugriff auf die potenziell unterschiedlich realisierten Datenspeicher ermöglichen, und einem zentral organisierten Lernobjekt-Service, der einerseits die Strukturinformationen von Kursen und Modulen (also Assoziationen und Kompositionen) speichert und andererseits Suchdienste zum Auffinden von Lernobjekten anbietet. Dem Lernobjekt-Repository steht das Annotations-Repository anbei, das als zentraler Dienst die Annotationen verwaltet. Das Annotations-Repository besteht ähnlich dem Lernobjekt-Repository aus einem Datenspeicher, der die Annotationen selbst beinhaltet, und einer Annotationszugriffsschicht, die den Zugriff auf diesen Datenspeicher regelt. Auf beide Komponenten – Lernobjekt-Repository und Annotations-Repository – greifen die Lehrmaterial-Viewer, Annotationseditoren und Authoring-Systeme zu, wenn Annotationen gelesen oder neu erstellt werden sollen. Bei der Erstellung und beim Lesen von Annotationen erzeugen das Lernobjekt-Repository und das Annotations-Repository Ereignisse, die potenziell auch für andere Nutzer und Designer interessant sein könnten. Diese Ereignisse werden an den Notifikations-Service weitergeleitet, der alle Ereignisse sammelt, gegebenenfalls zwischenspeichert, zu Listen aufbereitet und entsprechend den Wün-

schen der Nutzer als E-Mail, SMS, Instant Message, etc. verschickt. Neben dem Lernobjekt-Repository und dem Annotations-Repository verschickt auch der Lernobjekt-Scanner, der für die Integrität von Lernobjekten und den ihnen zugeordneten Annotationen verantwortlich ist, Ereignisse an den Notifikations-Service.

Alle diese Komponenten bilden in ihrer Gemeinsamkeit die Annotationsinfrastruktur und erlauben einen transparenten Austausch von Annotationen zu Lernobjekten. Da Annotationen selbst wieder annotiert werden können, sind auch Diskussionen über das Medium „Annotation“ möglich. Für die nötige Awareness, dass neue Annotationen – und damit möglicherweise auch neue Diskussionsbeiträge oder die Qualität der Lernobjekte steigernde Anmerkungen – verfügbar sind, wird durch das enge Zusammenspiel von Datenspeichern (Lernobjekt-Repository und Annotations-Repository) und Benachrichtigungskomponenten (Notifikations-Service) gesorgt.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung und Ergebnisse

Ausgangspunkt dieser Arbeit war die Beobachtung, dass Lehrmaterialien derzeit auch trotz umfangreicher Unterstützung durch verschiedene Lernplattformen immer noch selten wiederverwendet werden. Werden für ein Skript oder für Vorlesungsunterlagen Lehrmaterialien mit bestimmten Inhalten benötigt, werden diese meist neu erstellt, statt dass bereits vorhandene Materialien den Anforderungen entsprechend angepasst und weiterverwendet werden.

Die Ursachen hierfür liegen nur zum Teil an der Unterstützung der Lehrmaterialautoren und –nutzer durch entsprechende Software-Systeme. Ein wesentlicher Punkt ist auch die Art und Weise, wie die Lehrmaterialien bislang erstellt werden: Die Lehrmaterialien werden isoliert, ohne Blick auf bereits vorhandene Lehrmaterialien, erstellt. Oder aber, falls doch eine Zusammenarbeit mit anderen Lehrmaterialautoren besteht, ist nur eine sehr lose, kaum organisierte Abstimmung zwischen den Lehrmaterialautoren zu beobachten. Eine Qualitätskontrolle der entstehenden Lehrmaterialien oder ein gezieltes Einarbeiten von Feedback der Lerner findet im allgemeinen nicht statt.

In dieser Arbeit wurde ein Ansatz vorgeschlagen, der diese Defizite behebt: Die Wiederverwendung der Lehrmaterialien wird erhöht, indem die am Lehrmaterialerstellungsprozess Beteiligten enger zusammengeführt werden und so eine bessere Abstimmung und Information bei der Lehrmaterialerstellung erfolgt. Dies beschleunigt die Lehrmaterialerstellung und die Information über bereits verfügbare Lehrmaterialien wird deutlich verbessert. Außerdem werden die Nutzer von Lehrmaterialien – Lerner, Dozenten und Tutoren gleichermaßen – in die Lage versetzt, Anmerkungen, Korrekturvorschläge und allgemeines Feedback an die Lehrmaterialautoren zurückzuliefern, die dann das Feedback in die Lehrmaterialien einbauen und so die Qualität der Lehrmaterialien und damit auch deren Wiederverwendbarkeit nachhaltig verbessern.

Als Mittel, um dies zu erreichen, werden in dieser Arbeit Annotationen vorgeschlagen. Annotationen ermöglichen einen informellen und nichtformalen Informationsfluss zwischen den verschiedenen, an der Lehrmaterialerstellung und –nutzung beteiligten Personen. Sie ermöglichen es, formal schwer greifbare Aspekte von Lehrmaterialien zu beschreiben, so dass über Annotationen in Kombination mit Metadaten eine umfassende Beschreibung bereits bestehender Lehrmaterialien erreicht werden kann. So können Lehrmaterialien leichter katalogisiert und gefunden und damit auch leichter in neuen Lehrmaterialien wiederverwendet werden. Außerdem ermöglichen es Annotationen, gezieltes Feedback zu (Teilen von) Lehr-

materialien abzugeben, das dann von den Lehrmaterialautoren gelesen und so in die Entwicklung neuer Lehrmaterialien oder in die Überarbeitung bereits bestehender Lehrmaterialien einfließen kann. Dies erhöht letztlich die Qualität und damit die breitere Einsetzbarkeit von Lehrmaterialien.

Um zu diesem Ergebnis zu gelangen, waren mehrere Schritte nötig. Zuerst wurde das Umfeld betrachtet, in dem sich diese Arbeit bewegt. Dieses Umfeld ist zweigeteilt: Zum einen werden zentrale Begriffe dieser Arbeit vorgestellt, zum anderen werden verschiedene Aspekte von Wiederverwendbarkeit – nicht nur im Zusammenhang mit Lehrmaterialien – beleuchtet, analysiert und hinsichtlich ihrer konzeptionellen Stärken und Schwächen und hinsichtlich ihrer Nutzbarkeit für die Erhöhung der Wiederverwendbarkeit von Lehrmaterialien bewertet. Dies wurde in Kapitel 2 abgehandelt; dieses Kapitel legt den begrifflichen und konzeptionellen Rahmen fest, in dem diese Arbeit angesiedelt ist.

In den nächsten drei Kapiteln wurden die drei Säulen vorgestellt, auf denen eine Verbesserung der Wiederverwendung der Lehrmaterialien ruht: Eine Vereinheitlichung der bei der Lehrmaterialerstellung benötigten Datenbasis (Kapitel 3), eine Darstellung, Formalisierung und kritische Betrachtung der zur Lehrmaterialerstellung nötigen Abläufe (Kapitel 4) und die Zusammenführung und Optimierung dieser beiden Aspekte mit Hilfe von Annotationen (Kapitel 5). Damit bei diesen Schritten die Annotationen auch die beabsichtigte Wirkung entfalten können, ist eine Infrastruktur nötig, die bestimmte Dienste bereitstellt. Diese Infrastruktur muss einerseits einen einheitlichen Zugriff auf alle Daten und Lehrmaterialien bereitstellen – und zwar unabhängig von der tatsächlichen Speicherung der Daten und Lehrmaterialien – und sie muss andererseits die Annotation beliebiger Teile der Lehrmaterialien ermöglichen und zugleich für die Information von Interessenten (Lehrmaterialautoren, Lerner, Dozenten, etc.) sorgen. Zugleich muss sie sich leicht in verschiedenste, bereits existierende Lernplattformen integrieren lassen und eine Nutzung bereits bestehender Lehrmaterialien ermöglichen. Eine Infrastruktur, die dies leistet, wurde in Kapitel 6 konzeptionell vorgestellt. Ihre prototypische Implementierung wurde in Kapitel 7 beschrieben.

8.2 Ausblick und abschließende Bemerkungen

Der in dieser Arbeit vorgestellte Ansatz trägt zur Verbesserung der Wiederverwendbarkeit von Lehrmaterialien und der Erhöhung ihrer Qualität bei. Das gemeinsame Datenmodell ermöglicht es, die große Menge der bereits verfügbaren Lehrmaterialien in einer einheitlichen Hülle (den Atomen) zu kapseln und diese in allen Prozessschritten auf transparente Weise zu nutzen und neu (in Modulen und Kursen) zu kombinieren. Lehrmaterialien werden so einer großen Zahl von Lehrmaterialautoren und -nutzern verfügbar gemacht; die Lehrmaterialien können hierbei auf vielerlei Weise genutzt werden – auch auf Arten, für die sie ursprünglich vielleicht gar nicht vorgesehen waren. Zusätzlich kann durch die Nutzung eines gemeinsamen Datenmodells der ungeordnet und nicht zentral koordinierbar ablaufende Prozess der Lehrmaterialerstellung auf eine einheitliche, geordnete Basis gestellt werden. Die Nutzung von Annotationen als ein an Lehrmaterialien gebundenes Kommunikationsmittel zwischen den Prozessteilnehmern erlaubt überdies eine auf die jeweiligen Lehrmaterialien fokussierte Kommunikation zum Zwecke der Abstimmung der Prozessteilnehmer bei der Lehrmaterialerstellung (Prozessaspekt) und der Verbesserung der Qualität der Lehrmaterialien durch Feedback, Änderungswünsche und Korrekturen (Qualitätsaspekt).

Metriken zur Messung der Wiederverwendbarkeit und Qualität von Lehrmaterialien

In einem nächsten Schritt wäre durch geeignete eine Metrik zu quantifizieren, wie stark der in dieser Arbeit vorgestellte Ansatz zu einer Erhöhung der Wiederverwendbarkeit und einer Verbesserung der Qualität der Lehrmaterialien beiträgt. Hierzu ist eine Metrik zu definieren,

mit der man Wiederverwendbarkeit von Lehrmaterialien messen kann. Die Metrik muss sowohl den Grad der Wiederverwendbarkeit als auch die Qualität der Lehrmaterialien berücksichtigen. So eine Metrik erlaubt es, unterschiedliche Lehrmaterialien hinsichtlich ihrer Wiederverwendbarkeit zu vergleichen, und es wird möglich, unterschiedliche Prozesse zur Lehrmaterialerstellung anhand konkreter Zahlen zu vergleichen. Um hierbei zu repräsentativen Aussagen zu kommen, muss die Annotationsinfrastruktur von möglichst vielen Designern und Nutzern – idealerweise über mehrere, unterschiedlich arbeitende Organisationen und Unternehmen hinweg – für die Arbeit mit Lehrmaterialien eingesetzt werden.

Es bietet sich an, unter Verwendung der Metrik eine Studie durchzuführen mit dem Ziel, Vorgaben und Richtlinien für bislang wenig formalisierte Teile des Lehrmaterialerstellungsprozesses (beispielsweise die Anforderungsanalyse) herzuleiten, um den Designern Hilfen zur Entwicklung gut wiederverwendbarer Lehrmaterialien an die Hand zu geben. Hierzu müssen folgende Fragen geklärt werden:

- Wie können die Anforderungen hinsichtlich Wiederverwendbarkeit und Qualität in konkrete Vorgaben umgewandelt werden?
- Wie kann die Einhaltung dieser Vorgaben überprüft werden? Wo bzw. zu welchem Zeitpunkt im Prozess sind die Messpunkte zu setzen, an denen die Wiederverwendbarkeit der Lehrmaterialien und die Qualitätsvorgaben anhand der Metrik überprüft werden?
- Wie können bei den Messungen sowohl Aspekte der Lehrmaterialien als auch Aspekte des Prozesses bzw. der gerade aktiven Prozessteilnehmer mit berücksichtigt werden?

Während der Studie muss insbesondere das Kommunikationsverhalten zwischen den Teilnehmern beobachtet werden. Prozessschritte, in denen eine besonders aktive Kommunikation stattfindet, scheinen einer verstärkten Abstimmung der Prozessteilnehmer untereinander zu bedürfen. Diese Prozessschritte müssen später daraufhin untersucht werden, ob sie nicht durch Vorgaben, Richtlinien, Checklisten, etc. für die Prozessteilnehmer so gesteuert werden können, dass der Abstimmungsaufwand sinkt und den Prozessteilnehmern mehr Zeit bleibt, sich ihrer eigentlichen Aufgabe zu widmen.

Von Kommunikation und Koordination zu Kooperation

Durch Annnotationen wird es möglich, dass a priori nicht bekannte Personen miteinander kommunizieren, indem sie die Nachrichten (die Annotationen) an die gemeinsam genutzten Lernobjekte anheften. Zugleich helfen Annotationen bei der Koordination an den Schnittstellen zwischen den Teilprozessen des Lehrmaterialienerstellungs- und -nutzungsprozesses.

Im weiteren Fortgang kann die Intensität des Miteinander ausgehend von Zusammenarbeit und Kommunikation über Koordination bis hin zu einer engen Kooperation zunehmen. Die Frage nach dem gemeinsamen Ziel [BoSc00], also die Kooperation zwischen Designern, Designern und Nutzern, sowie Nutzern untereinander durch Unterstützung mit Annotationen ist in dieser Arbeit nur am Rand betrachtet. Es stellt sich die Frage, welcher Art die sozialen Beziehungen sind, die auf diese Weise entstehen. Bilden sich eher Teams, die ein gemeinsames Ziel finden und ihre persönlichen Ziele dem gemeinsamen Ziel unterordnen? Oder entstehen eher Strukturen ähnlich virtueller Communities, die durch ein gemeinsames Interesse – die möglichst schnelle und einfache Erstellung qualitativ hochwertiger Lehrmaterialien – zusammenarbeiten?

In engem Zusammenhang mit dieser Fragestellung steht das Problem des Übergangs von asynchroner zu synchroner Kommunikation. In dieser Arbeit wurden Annotationen als Kom-

munikationsmedium zwischen Prozessteilnehmern vorgeschlagen. Annotationen werden zur asynchronen Kommunikation genutzt und können – insbesondere wenn passive Benachrichtigungen genutzt werden – sehr diskret eingesetzt werden. Die Beobachtung des Kommunikationsverhaltens der Prozessteilnehmer kann nun Aufschluss darauf geben, wie sich Beziehungen im Virtuellen entwickeln. In diesem Zusammenhang sind vor allem folgende Fragestellungen interessant: Bis zu welcher Intensität an Zusammenarbeit sind Annotationen als asynchrones Kommunikationsmedium ausreichend und wann nehmen die Prozessteilnehmer synchrone Kommunikationsmedien hinzu? Hängt die Nähe der Prozessteilnehmer untereinander von der Verfügbarkeit bestimmter Kommunikationsmittel (Annotationen, E-Mail, Chat, etc.) ab? Falls ja, welche Einflussfaktoren bestimmen den Einsatz bestimmter Kommunikationsmittel?

Interkultureller Einsatz

Ein wesentliches Element dieser Arbeit war der Prozess der Lehrmaterialerstellung und –nutzung. Je nach Kultur kann sich dieser Prozess jedoch völlig unterschiedlich gestalten: Während sich der grobe Ablauf nicht wesentlich von Kultur zu Kultur ändern wird, werden doch große Unterschiede gerade im Kommunikations- und Kooperationsverhalten sichtbar werden. Annotationen mögen in westlichen Kulturen einen Beitrag zur Erhöhung der Wiederverwendung und zur Verbesserung der Qualität leisten, aber gilt diese Aussage gleichermaßen beispielsweise in Asien? Und – noch interessanter – wie kann die Wiederverwendbarkeit von Lehrmaterialien, die von Arbeitsgruppen aus unterschiedlichen Kulturen erstellt wurden, sichergestellt werden? Sind hierzu Annotationen ausreichend oder sind weitere Maßnahmen nötig? Diese Fragestellungen sind heute noch vielfach unbeantwortet, aber sie gewinnen gerade im Kontext der immer stärker spürbar werdenden Globalisierung enorm an Bedeutung.

Dies ist eine Auswahl von Fragen und Themen, denen man in weiteren Arbeiten nachgehen kann. Weitere Fragen – auch aus anderen Wissensgebieten als der Informatik – sind denkbar, beispielsweise Fragen nach der didaktischen Wirksamkeit von Annotationen von Nutzern der Lehrmaterialien. Diese Arbeit legt die Grundlagen für die Klärung dieser Fragen.

Trotz der vielen interessanten Fragestellungen, die die Ergebnisse dieser Arbeit aufwerfen, und der vielen, in der Folge noch durchzuführenden Maßnahmen: Das Ziel dieser Arbeit war es aufzuzeigen, wie die Wiederverwendung von Lehrmaterialien verbessert werden kann. Das Ergebnis dieser Arbeit liefert konkrete Maßnahmen, wie dieses Ziel erreicht wird. Statt mehrere Themen einzeln nebeneinander zu betrachten, werden in dieser Arbeit alle Aspekte, die Einfluss auf die Wiederverwendbarkeit von Lehrmaterialien haben, integriert: Aspekte der Prozessoptimierung, Aspekte der Kommunikation und Aspekte des Datenmodells werden so aufeinander abgestimmt, dass die vielfältige Nutzung von Lehrmaterialien möglichst einfach möglich wird und zugleich eine hohe Qualität der Lehrmaterialien sichergestellt wird. Die Ergebnisse dieser Arbeit zeigen so einen Weg hin zu vielfältig einsetzbaren, qualitativ hochwertigen Lehrmaterialien auf.

Anhang A: Modifikationen des LOM-Basisschema

Im folgenden werden die Änderungen am LOM-Basisschema vorgestellt. LOM-Parameter, die aufgrund der Vorgehensweise in dieser Arbeit überflüssig werden, sind durchgestrichen dargestellt, neu hinzukommende Parameter kursiv.

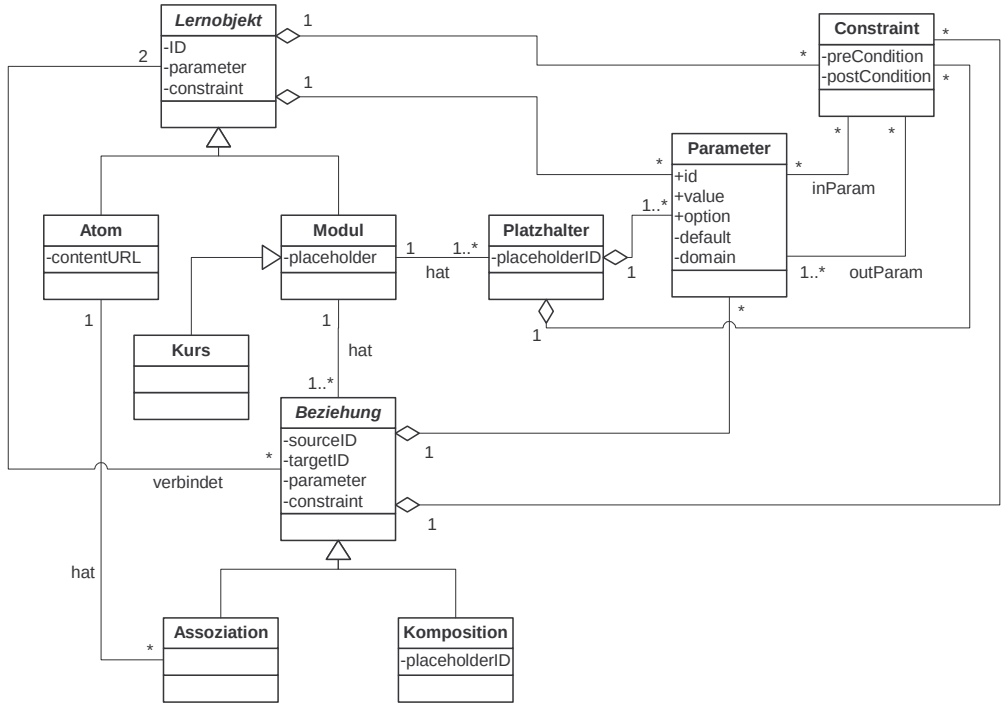
Kategorie und Attribute	**Beschreibung**
General	**Allgemeine Informationen**
Identifier: Identifier	ID
Title: LangString	Name des Lernobjekts
CatalogEntry*	Zuordnung zu einem Katalogsystem
Catalogue: String	ID, Name
Entry	Eintrag
Language: Locale	Sprache
Description: LangString	Beschreibung
Keywords: LangString	Stichwort
Coverage: LangStringy	Räumliche und zeitliche Angaben
~~Structure: Vocabulary~~	Struktur *In dieser Arbeit weggelassen.*
~~Aggeragation Level: Vocabulary~~	Typ des Lernobjekts: Atom etc. *In dieser Arbeit weggelassen.*
Life Cycle	**Bearbeitungsinformationen**
Version	Versionsnummer
Status	Bearbeitungsstatus
Contribute*	Prozessinformationen
Role: vocabulary	Rolle, die für die Erstellung des Lernobjekts beteiligt ist
Entry: String	Eintrag
Date: Date	Datumsangaben
Meta-Metadata	**Beschreibung der Metadatensätze** ***Neu: Erweiterung über Konzeptebene um Strukturierungsarten und Nutzungsklassen***
Identifier: reserved	ID
CatalogEntry*	Zuordnung der Metadaten zu einem Katalogsystem
Catalogue: String	ID, Name
Entry	Eintrag
Contribute*	Prozessinformationen
Role: vocabulary	Rolle, die für die Erstellung des Lernobjekts beteiligt ist
Entry: String	Eintrag
Date: Date	Datumsangaben
Metadata Scheme: String	Metadatenschema
Language: String	Sprache
Technical	**Technische Eigenschaften**
Format: Format	Mime Type
Size: Integer	Größe in Bytes
Location: Locsspec	URL
Requirements	Systemanforderungen

Type: Vocabulary	Systemtyp
Name: Vocabulary	Name des Systems
Minimumversion: String	Version
Maximumversion: String	Version
Installation Remarks: LangString	Hinweise zur Installation des Lernobjekts
Other Platform Requirements: LangString	Andere Systemanforderung
Duration: Date	Abspielzeit
Educational	**Einsatz von Lernobjekten in der Lehre** ***Neu: In dieser Arbeit aus Gründen der Übersichtlichkeit in „Educational“ und „Content“ aufgespalten***
Interactivity type: Vocabulary	Interaktionsmöglichkeiten
Interactivity Level: String	Grad der möglichen Interaktion
Intended End User Role: Vocabulary	Rolle des Endnutzers
Context: Vocabulary	Lehrkontext
Typical Age Range: LangString	Altergruppe
Typical Learning Time: Date	Benötigte Lernzeit
Description: String	Beschreibung des Lernobjekts
Language: String	Sprache der Zielgruppe
Form: Vocabulary	*Neu: Lehrform, beispielsweise Vorlesung, Tutorübung, etc.*
Target: Vocabulary	*Neu: Lehrziel*
Approach: Vocabulary	*Neu: Lehransatz, beispielsweise mehr Theorie, mehr Beispiele, etc.*
Knowlege: Vocabulary	*Neu: Vorhandenes Wissen*
Content	**Inhalt von Lehrmaterialien**
Semantic Density: String	Semantische Dichte
Learning Resource type: Vocabulary	Typ des Lernobjekts
Difficulty: String	Schwierigkeitsgrad
Abstraction: String	*Neu: Abstraktionsgrad*
Detail: String	*Neu: Detaillierungsgrad*
Subject: String	*Neu: Thema*
Rights	**Rechtliche Bestimmungen**
Für diese Arbeit nicht relevant.	Für diese Arbeit nicht relevant.
Relation	**Beziehung zu anderen Lernobjekten**
Kind: Vocabulary	Art der Beziehung
Resource	Bezeichnung des Ziel-Lernobjekts
Identifier: reserved	Identifikator
Description. LangString	Beschreibung
CatalogEntry*	Katalog-Eintrag
placeholderID: LangString	*Neu: Platzhalter-ID*
Annotation	**Kommentare zum Lernobjekt**
Person: String	Autor
Date: Date	Erstellungsdatum
Description: LangString	Inhalt
Classification	**Zuordnung zu bestimmten Klassifizierungssystemen**
Purpose: Vocabulary	Zweck der Klassifizierung
TaxonPath*	Klassifizierungssystem
Source: String	Name

Taxon+ ID: String Entry: LangString Description: LangString Keywords: LangString	Knoten innerhalb der Zuordnung ID Eintrag Beschreibung Stichworte
Presentation	***Neu: Darstellung und Navigation des Lernobjekts***
Navigation: Vocabulary *Layout** *Layout: Vocabulary* *Dataformat: Vocabulary*	*Neu: Navigationsform* *Neu: Layoutinformationen* *Neu: Präsentationsformat*

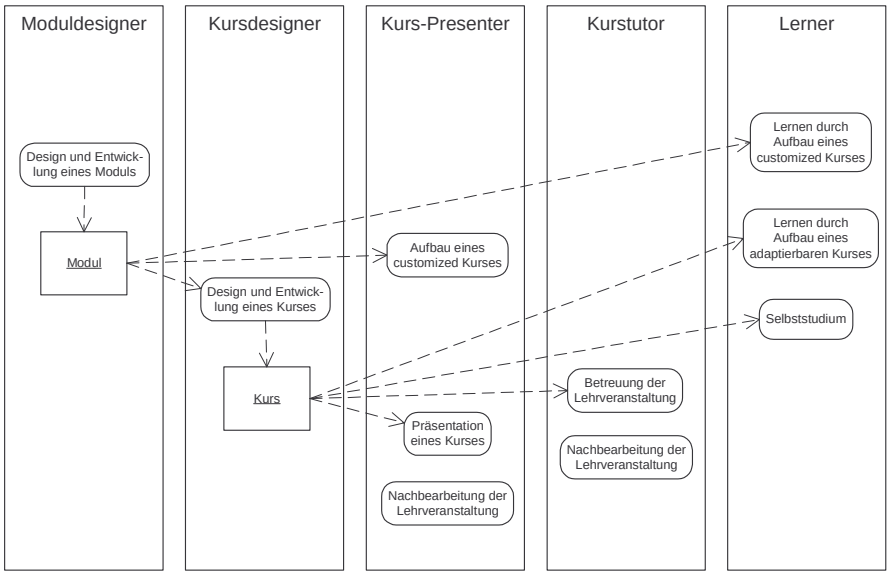
Anhang B: Klassendiagramm des Datenmodells

In Kapitel 3 wurde Schritt für Schritt das Datenmodell hergeleitet, das neben dem Lehrmaterialerstellungsprozess aus Kapitel 4 die Basis für das Konzept zur Verbesserung der Wiederverwendung von Lernobjekten bildet, das in dieser Arbeit vorgestellt wurde. In Kapitel 3 wurden immer nur die Ausschnitte dargestellt, die für das Verständnis des jeweiligen Abschnitts nötig waren. An dieser Stelle wird nun das vollständige Datenmodell dargestellt.

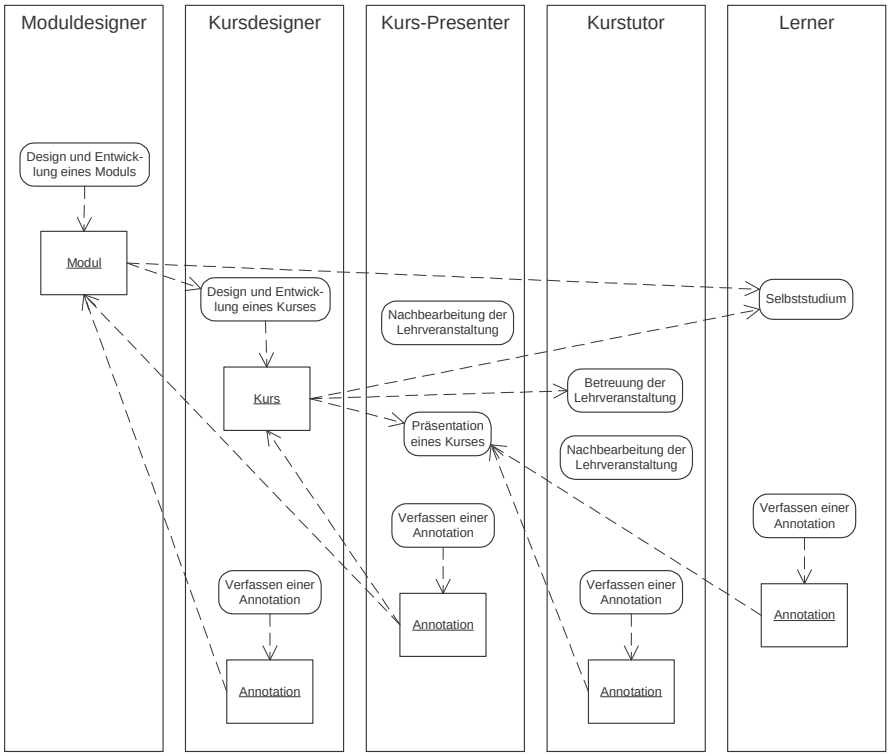


Anhang C: Prozessbeschreibungen

In Kapitel 4 wurde ein generischer Lehrmaterialerstellungsprozess vorgestellt. An dieser Stelle ist eine Beschreibung dieses Prozesses in UML zu finden.



Durch die Hinzunahme von Annotationen in den Lehrmaterialerstellungsprozess, wie dies in Kapitel 5 hergeleitet wurde, ändert sich auch die Prozessbeschreibung. In folgender Abbildung ist der Lehrmaterialerstellungsprozess, ergänzt durch Annotationen, aufgezeigt.



Anhang D: Exportierte Dienste der Komponenten der Annotationsinfrastruktur

Dienste des Lernobjekt-Repositories

Folgende Tabelle beschreibt die Dienste, die die Lernobjektzugriffsschicht (vgl. 6.3.2.3) anderen Komponenten der Annotationsinfrastruktur anbietet:

Signatur des Dienstes	Parameter und Ergebnis	Beschreibung
Object get(*ObjID, UserID*)	*ObjID*: Identifikator des gesuchten Lernobjekts *UserID*: Identifikator des Nutzers. *Object*: Das gesuchte Lernobjekt	Liefert das Lernobjekt mit dem Identifikator *ObjID*, falls der Nutzer *UserID* die Berechtigung hat, das Lernobjekt anzufordern. Gegebenenfalls muss vor der Anforderung mit get() eine Authentifizierung mittels authenticate() durchgeführt werden.
void put(*UserID, Object, Params*)	*UserID*: Identifikator des Nutzers. *Object*: Das im Datenspeicher abzulegende Lernobjekt *Params:* Menge von Parametern, die das Lernobjekt *Object* beschreibt.	Legt das Lernobjekt *Object*, beschrieben durch die Parameter *Params*, im Datenspeicher ab. *Params* enthält auch den Identifikator des Lernobjekts. Nutzer *UserID* muss die Berechtigung haben, das Lernobjekt anzufordern. Gegebenenfalls muss vor der Anforderung mit get() eine Authentifizierung mittels authenticate() durchgeführt werden.
hasPermission authenticate(*UserID, Credential*)	*UserID*: Eindeutiger Identifikator des Nutzers. *Credential:* Objekt zur Authentifizierung des Nutzers *UserID*, beispielsweise ein Passwort.. *hasPermission:* Boolesches Flag, das anzeigt, ob die Authentifizierung erfolgreich war	Führt die Authentifizierung der Nutzers *UserID* beim Datenspeicher durch. Die Authentifizierung erfolgt auf Grundlage des *Credential*-Objekts.

Lernobjekt-Service

Der Lernobjekt-Service (vgl. 6.3.2.4) exportiert folgende Dienste:

Signatur des Dienstes	Parameter und Ergebnis	Beschreibung
*URL*s getServers(*ObjID*)	*ObjID*: Identifikator des gesuchten Lernobjekts *URL*s: Menge der Server-URLs, auf denen das Lernobjekt *ObjID* abgelegt ist. Die URLs verweisen auf die jeweiligen Lernobjektzugriffsschichten.	Liefert die Menge der URLs der Server, auf denen Lernobjekt *ObjID* gespeichert ist. Da Lernobjekt *ObjID* auf mehreren Rechnern gespeichert sein kann, wird eine Menge als Ergebnis zurückgeliefert. Dieser Dienst wird von der Verzeichniskomponente des Lernobjekt-Service erbracht.
ObjIDs getObjIDs (*Params*)	*Params*: Parameter, die die gesuchten Lernobjekte erfüllen müssen *ObjIDs:* Menge der Identifikatoren passender Lernobjekte	Liefert die Menge der Identifikatoren von Lernobjekten, die zu den übergebenen Parametern *Params* passen. Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.
Params getDescr(*ObjID*)	*ObjID*: Identifikator des Lernobjekts, dessen Beschreibung angefordert wird *Params:* Menge von Parametern, die Lernobjekt *ObjID* beschreiben	Liefert die Beschreibung von Lernobjekt *ObjID* als Menge von Parametern. Dieser Dienst wird von der Katalogkomponente des Lernobjekt-Service erbracht.
void register(*ObjID*, *Params*, *URL*)	*ObjID*: Identifikator des zu registrierenden Lernobjekts *Params*: Parameter, die das Lernobjekt beschreiben *URL*: URL der Lernobjektzugriffsschicht, über die Lernobjekt *ObjID* abgerufen werden kann	Registriert Lernobjekt *ObjID* beim Lernobjekt-Service. Dieser Dienst wird von der Verzeichniskomponente des Lernobjekt-Service erbracht.
void addSubconnection (*ConID*, *ObjID*, *SubID*)	*ConID*: Identifikator der neu anzulegenden Kompositionsbeziehung. *ObjID*: Identifikator des Lernobjekts, dem ein Subelement hinzugefügt werden soll. *SubID*: Identifikator des hinzuzufügenden Subelements.	Definiert eine Kompositionsbeziehung mit Identifikator *ConID* zwischen Lernobjekt *ObjID* und einem Subelement *SubID*.

void removeSubconnection(*ConID*)	*ConID*: Identifikator der Kompositionsbeziehung, die aufgelöst werden soll.	Entfernt die Kompositionsbeziehung *ConID* zwischen zwei Lernobjekten.
SubIDs getSubconnections(*ObjID*)	*ObjID*: Identifikator des Lernobjekts, dessen Subelemente als Ergebnis geliefert werden sollen. *SubIDs*: Menge von Identifikatoren aller Kompositionen, die von *ObjID* ausgehen.	Liefert eine Menge *SubIDs* von Identifikatoren der Subelemente von Modul bzw. Kurs *ObjID*.
void addAssociation (*ConID, ObjID, TargetID*)	*ConID*: Identifikator der neu anzulegenden Assoziation. *ObjID*: Identifikator des Lernobjekts, von dem die Assoziation ausgeht. *TargetID*: Identifikator des assoziierten Lernobjekts.	Definiert eine Assoziation mit Identifikator *ConID* zwischen Lernobjekt *ObjID* und Lernobjekt *TargetID*.
void removeAssociation(*ConID*)	*ConID*: Identifikator der Assoziation, die aufgelöst werden soll.	Entfernt die Assoziation *ConID* zwischen zwei Lernobjekten.
IDs getAssociations (*ObjID*)	*ObjID*: Identifikator des Lernobjekts, von dem aus alle als Ergebnis gelieferten Assoziationen ausgehen. *SubIDs*: Menge von Identifikatoren aller Assoziationen, die von *ObjID* ausgehen.	Liefert eine Menge *IDs* von Identifikatoren der Subelemente von Modul bzw. Kurs *ObjID*.

Dienste des Annotations-Repositories

Die Annotationszugriffsschicht stellt zwei Gruppen von Diensten bereit: Dienste zum Zugriff auf die Annotationen und Dienste zur Verwaltung von Gruppen und Rechten bezüglich des Zugriffs auf Annotationen.

Signatur des Dienstes	Parameter und Ergebnis	Beschreibung
void putAnnotation(*ObjID, ObjArea, AnID, CourseID, UserID*)	*ObjID*: Identifikator des Lernobjekts, das mit der Annotation *AnID* verbunden werden soll (Parameter anchor-id) *ObjArea*: Beschreibung des Ausschnitts des Lernobjekts, der annotiert wird (Parameter anchor-area) *AnID*: Identifikator der zu setzenden Annotation (Parameter	Hängt die Annotation mit dem Identifikator *AnID* an Lernobjekt *ObjID*. Eine Prüfung der Zugriffsrechte ist hier nicht nötig, da jeder Annotationen zu beliebigen Lernobjekten verfassen kann. (Eine Authentifizierung von *UserID* wird jedoch durchgeführt.)

	annotation-id) *CourseID*: Identifikator des Kurses, in dem die Annotation gültig ist. Ist die Annotation nicht an einen Kurs gekoppelt, so ist *CourseID* Null. *UserID*: Kennung des die Annotation setzenden Nutzers	
void putAnnotationText (*ObjID*, *ObjArea*, *Text*, *CourseID*, *UserID*)	*ObjID*: Identifikator des Lernobjekts, das mit der Annotation verbunden werden soll *ObjArea*: Beschreibung des Ausschnitts des Lernobjekts, der annotiert wird *Text*: Anmerkung des Nutzers als Freitext *CourseID*: Identifikator des Kurses, in dem die Annotation gültig ist *UserID*: Kennung des die Annotation setzenden Nutzers	Erzeugt ein neues Modul und ein neues Atom, fügt den Freitext *Text* an das Atom an, fügt das Atom in das Modul ein und verbindet das Lernobjekt *ObjID* mit diesem Modul mittels einer Assoziation. Eine Prüfung der Zugriffsrechte ist hier nicht nötig, da jeder Annotationen zu beliebigen Lernobjekten verfassen kann. (Eine Authentifizierung von *UserID* wird jedoch durchgeführt.)
Object getAnnotation (*AnID*, *UserID*)	*AnID*: Identifikator der angeforderten Annotation *UserID*: Kennung des anfordernden Nutzers *Object*: Die angeforderte Annotation	Liefert die Annotation mit dem Identifikator *AnID*. Der Zugriff wird anhand der Kennung *UserID* geprüft.
*Annotation*s getAnnotations (*ObjID*, *CourseID*, *UserID*)	*ObjID*: Identifikator des Lernobjekts, dessen Annotationen geliefert werden sollen *CourseID*: Identifikator des Kurses, in dem die Annotation gültig ist *UserID*: Kennung des anfordernden Nutzers *Annotation*s: Menge von Annotationen	Liefert die Menge der Annotationen, die für Lernobjekt *ObjID* verfasst wurden und für Nutzer *UserID* sichtbar sind.
*AnID*s getAnnotationIDs (*ObjID*, *CourseID*, *UserID*)	*ObjID*: Identifikator des Lernobjekts, dessen Annotationen geliefert werden sollen *CourseID*: Identifikator des Kurses, in dem die Annotation	Liefert die Menge der Identifikatoren der Annotationen, die für Lernobjekt *ObjID* in Kurs *CourseID* verfasst wurden und für Nutzer *UserID* sichtbar sind.

	gültig ist *UserID*: Kennung des anfordernden Nutzers *AnID*s: Menge von Identifikatoren von Annotationen	
void deleteAnnotation (*AnID, UserID*)	*AnID*: Identifikator der zu löschenden Annotation *UserID*: Kennung des die Annotation löschenden Nutzers	Löscht Annotation *AnID*, indem die Assoziation zwischen Annotation *AnID* und dem annotierten Lernobjekt gelöscht wird, falls Nutzer *UserID* hierzu die Berechtigung hat.
void createGroup(G*roupID, UserID*)	*GroupID*: Identifikator (eine beliebige Zeichenkette) der zu erzeugenden Gruppe *UserID*: Kennung des Nutzers, der die Gruppe erzeugt	Erzeugt eine Gruppe *GroupID* mit Nutzer *UserID* als einzigem Mitglied. Jeder darf Gruppen erzeugen.
void deleteGroup(*GroupID, UserID*)	*GroupID*: Identifikator der zu löschenden Gruppe *UserID*: Kennung des Nutzers, der die Gruppe löschen will	Löscht Gruppe *GroupID*, falls *UserID* die Rechte hierzu hat.
void addToGroup(*GroupID, UserID*)	*GroupID*: Identifikator der Gruppe, zu der Nutzer *UserID* hinzugefügt werden soll *UserID*: Kennung des Nutzers, der zu Gruppe *GroupID* hinzugefügt werden soll	Fügt den Nutzer mit der Kennung *UserID* zu Gruppe *GroupID* hinzu.
void removeFromGroup (G*roupID, UserID*)	*GroupID*: Identifikator der Gruppe, aus der ein Nutzer entfernt werden soll *UserID*: Kennung des Nutzers, der aus Gruppe *GroupID* entfernt werden soll	Entfernt den Nutzer mit der Kennung *UserID* aus Gruppe *GroupID*.
visible isVisible(*AnID, UserID*)	*AnID*: Identifikator einer Annotation *UserID*: Kennung des Nutzers, für den Annotation *AnID* geprüft werden soll *visible*: Boolescher Wert	Prüft, ob die Annotation mit dem Identifikator *An-ID* für den Nutzer mit der Kennung *UserID* sichtbar ist.
void authenticate(*UserID, Password*)	*UserID*: Identifikator des zu authentifizierenden Nutzers. *Password:* Passwort von Nutzer *UserID*	Führt die Authentifizierung der Nutzers *UserID* bei der Annotationszugriffsschicht durch.

Dienste des Notifikations-Service

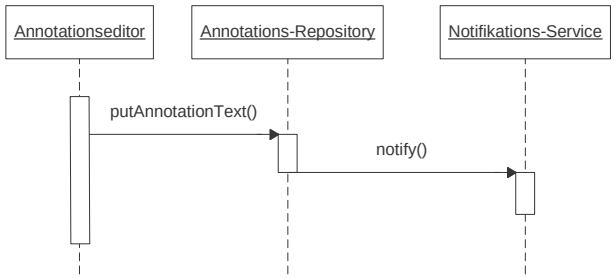
Aufgabe des Notifikations-Service (vgl. 6.3.6) ist es, Nachrichten gezielt zu verteilen. Hierzu stellt er folgende Dienste bereit:

Signatur des Dienstes	Parameter und Ergebnis	Beschreibung
void notify(*Nachricht)*	*Nachricht*: Eine Nachricht, die Informationen über ein Ereignis vermittelt	Meldet dem Notifikations-Service ein Ereignis in Form einer Nachricht.
*AnID*s getAnnotationIDs (*ObjID*, *UserID*)	*ObjID*: Identifikator des Lernobjekts, dessen Annotation potenziell inkonsistent ist *UserID*: Kennung des anfordernden Nutzers *AnIDs*: Menge von Identifikatoren von potenziell inkonsistenten Annotationen	Liefert die Menge der im Nachrichtenpuffer gespeicherten Identifikatoren der Annotationen von Lernobjekt *ObjID*, die neu oder potenziell inkonsistent und für Nutzer *UserID* sichtbar sind.

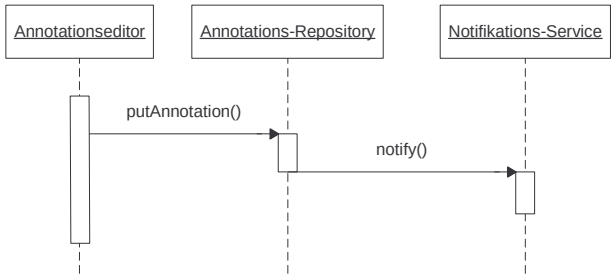
Anhang E: Sequenzdiagramme gängiger Use Cases

In 6.3 wurde das Zusammenspiel der einzelnen Komponenten der Annotationsinfrastruktur vorgestellt. Im folgenden werden Sequenzdiagramme für die vorgestellten Anwendungsszenarien geliefert.

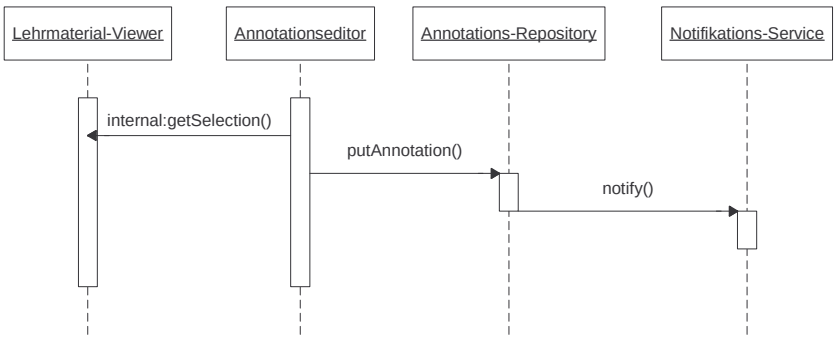
Verfassen einer Annotation zu einem Lernobjekt:



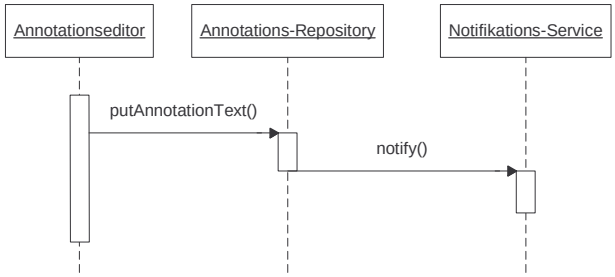
Verweis auf ein bestehendes Lernobjekt als Annotation zu einem anderen Lernobjekt:



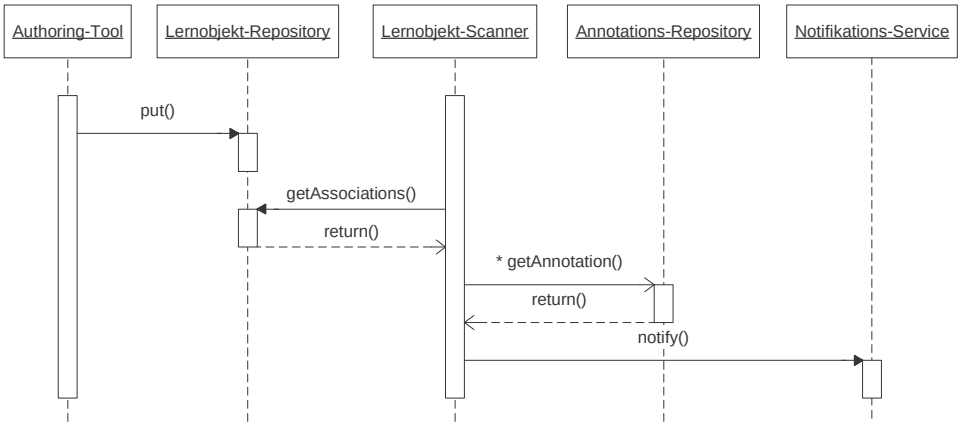
Verfassen einer Annotation zu präsentationsfertigen Lehrmaterialien:



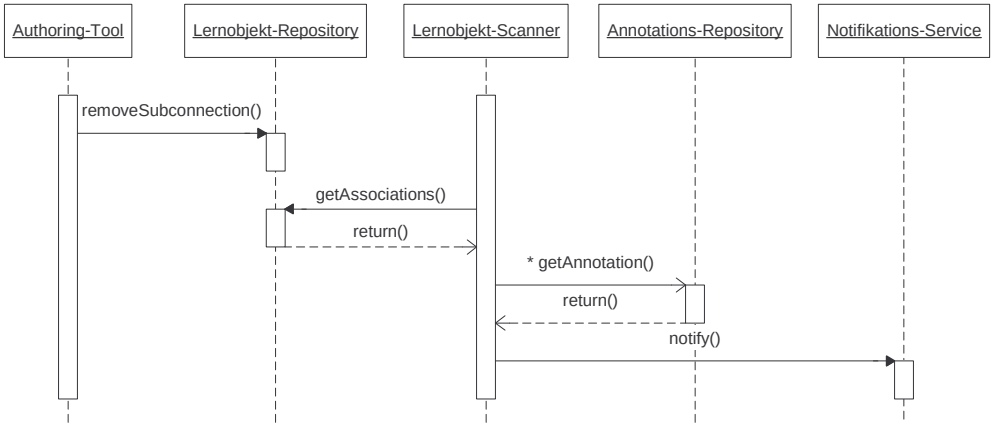
Verfassen von Annotationen zu einer bereits bestehenden Annotation:



Überarbeiten eines annotierten Lernobjekts:



Entfernen eines annotierten Lernobjekts aus einem Kurs:



Glossar

Adaptierbarer Kurs

Ein adaptierbarer Kurs wird durch den Kurs-Presenter anhand der vom Kursdesigner vorgesehenen Nutzungen passend parametrisiert.

Adaptierter Kurs

Ein adaptierter Kurs ist bereits vom Kursdesigner vollständig hinsichtlich der Nutzung spezifiziert. Der Kurs ist damit vom Kursdesigner von vorneherein für eine bestimmte Nutzung in der Lehre und vor allem auch im Hinblick auf eine bestimmte Nutzergruppe (als Kursvariante) parametrisiert.

Anker

Ein Anker bezeichnet einen Referenzpunkt. Er markiert den Teil eines Elements (Lernobjekt oder einer Annotation), auf den sich eine Annotation bezieht.

Annotation

Annotationen haben einen Inhalt und einen Bezug zu einer Entität (hier: ein Lernobjekt wie ein Modul oder Atom, oder aber ein Kurs) haben.

Assoziation

Eine Assoziation bezeichnet allgemein eine Beziehung zwischen zwei Lernobjekten, die nicht notwendigerweise in einer hierarchischen Beziehung stehen müssen.

Atom

Atome sind nicht weiter sinnvoll unterteilbar, ohne dass ihre Eigenständigkeit und Wiederverwendbarkeit verloren ginge. Beispiele für Atome sind Texte, Animationen, Bilder, etc.

Basiselement

Eine Annotation hat immer einen Bezug zu einer Entität, die sie annotiert. Die Entität, auf die sich eine Annotation bezieht, wird Basiselement genannt.

Content Management

Content Management betrachtet einen Teil des Wissensmanagements, der sich mit dem Handling einer großen Menge von Informationen befasst. Content Management ist damit das grundlegende Werkzeug zum Generieren, Einspielen, Verteilen und Nutzen von Informationen, worauf alle weiteren Methoden des Wissensmanagements aufsetzen.

Curriculum

Ein Curriculum stellt einen Rahmen, meist eine Zusammenstellung von Kursen, bereit, der zur Erreichung eines wohldefinierten akademischen Zieles dient.

Customized Kurs

Ein customized Kurs wird von den Nutzern – Lerner und Kurs-Presenter – aus bestehenden Modulen aufgebaut. In diesem Fall übernehmen die Nutzer sowohl Kursdesign als auch Kursentwicklung und entwerfen so Kurse entsprechend ihren Interessen.

Domain, Domäne

Der Begriff Domäne im Kontext dieser Arbeit gliedert sich in Inhaltsdomäne (das Themengebiet) und in Anwendungsdomäne (die Nutzungsklassen).

Domänendesign

Domänendesign umfasst in dieser Arbeit das Erfassen möglichst aller Umstände und Anforderungen, unter denen Lehrmaterialien später eingesetzt werden sollen.

Domänenspezifische Wiederverwendung

Domänenspezifische Wiederverwendung von Lehrmaterialien bedeutet in dieser Arbeit die Nutzung der Lehrmaterialien in verschiedenen Lehrveranstaltungen derselben Anwendungsdomäne, aber in unterschiedlichen didaktischen Funktionen.

Domänenübergreifende Wiederverwendung

Domänenübergreifende Wiederverwendung von Lehrmaterialien bedeutet hier die Nutzung der Lehrmaterialien in unterschiedlichen Inhalts- oder Anwendungsdomänen, also beispielsweise in Lehrveranstaltungen mit unterschiedlichen Themengebieten.

Inhaltsergänzende Annotation

Annotationen können genutzt werden, um Lehrmaterialien selbst – also nicht Metadaten, sondern konkrete Inhalte – zu ergänzen. Solche Annotationen werden im inhaltsergänzende Annotationen genannt. Ziel von inhaltsergänzenden Annotationen ist es, den Inhalt der betreffenden Lehrmaterialien mit weiteren Informationen anzureichern, ohne jedoch den Lesefluss der Lerner zu unterbrechen.

Inline-Annotation

Die Beziehung zwischen einer Annotation und ihrem Basiselement kann auf verschiedene Weise realisiert werden. Annotationen können in das Basiselement eingebettet sein. In diesem Fall spricht man von einer Inline-Annotation.

Komposition

Von Komposition wird gesprochen, wenn ein Lernobjekt aus mehreren Subelementen aufgebaut ist. Das Lernobjekt ist dann mit seinen Subelementen mittels Kompositionen verbunden.

Kurs

Ein Kurs ist eine abgeschlossene Lehrveranstaltung zur Vermittlung von komplexen Inhalten mit einer starken logischen Strukturierung und einem didaktischen Ziel. Kurse sind besondere Lernobjekte: Strukturell sind sie zwar identisch mit einem gewöhnlichen Modul, das ein oder mehrere Submodule bzw. Atome kapselt. Sie sind allerdings nicht nur inhaltlich abgeschlossen, sondern auch in ihrer Nutzung in der Lehre.

Kursrahmen

Ein Kursrahmen ist ein Kurs, bei dem die Parameter festgelegt sind und der mit komplett parametrisierten Platzhaltern versehen ist. Den Platzhaltern sind jedoch noch keine Module zugewiesen.

Lehrmaterialien

Der Begriff „Lehrmaterialien“ bezeichnet in dieser Arbeit allgemein konkrete Lehrmittel, die die Lerner zum Wissenserwerb nutzen.

Lernplattform

Eine Lernplattform integriert Authoring- und Lernsysteme, welche über Schnittstellen miteinander verknüpft werden können. Eine Lernplattform konzentriert sich dabei auf inhaltliche Aspekte (content), deren Bewertung (assessment), die beschreibenden Metadaten sowie entsprechende Tools.

Lerneinheit

Anderer Begriff für ein (einfaches) Modul. Siehe dort.

Lernobjekt

Lernobjekt ist ein Oberbegriff für Atome und Module. Wenn nicht zwischen Atomen und Modulen unterschieden werden soll, wird in dieser Arbeit der Begriff „Lernobjekt“ verwendet.

Lernpfad

Ein Lernpfad ist eine im allgemeinen (zielgruppen-)spezifische Struktur, bestehend aus Lernobjekten eines Kurses. Ein Lernpfad schlägt eine konkrete Reihenfolge vor, in der die Lernobjekte abgearbeitet werden sollten.

Modul

Module sind komplexe Objekte. Sie bestehen aus Atomen und / oder wiederum aus Modulen (strukturelle Rekursion). Sie sind thematisch in sich abgeschlossen und eigenständig wiederverwendbar.

Nutzungsklassen

Nutzungsklassen fassen äquivalente Nutzungsszenarien zusammen. Nutzungsklassen sind ein Strukturierungsmittel der Konzeptebene.

Parameter

Parameter sind Metadaten. Sie sind typisiert und werden Wertebereichen (Datentypen) zugewiesen.

Referenzierte Annotation

Annotationen können als eigenständige Entitäten vorliegen. In diesem Fall wird die Beziehung zwischen dem Basiselement und ihrer Annotation mittels einer Referenz realisiert. Die Annotation wird dann referenzierte Annotation genannt.

Sicht

Die Daten und Lernobjekte, die während der Durchführung eines Teilprozesses betrachtet und genutzt werden, werden Sicht auf den Teilprozess genannt. Die Sicht auf den Teilprozess umfasst damit mindestens die Vereinigung der Eingangs- mit der Ausgangssicht.

Variante

Varianten sind Lernobjekte, die zwar die selben Inhalte behandeln, die aber in verschiedenen Nutzungsklassen angewandt werden.

Wiederverwendung

Den Einsatz wiederverwendbarer Lehrmaterialien in verschiedenen Kursen oder Nutzungen nennt man Wiederverwendung. In dieser Arbeit wird die Wiederverwendung von Lehrmate-

rialien zwischen mehreren Beteiligten aus (potenziell) unterschiedlichen Organisationen und Unternehmen betrachtet.

Wissen, Wissensmanagement

Wissen bezeichnet eine „bedeutungsvolle Vernetzung von Informationen“. Generelle Aufgabe des Wissensmanagements ist es, die Erzeugung, die Bewahrung und die Nutzung von Wissen zu erleichtern und darauf aufbauend die Prozesse zur Wissensgenerierung und –nutzung effizienter zu gestalten.

Literaturverzeichnis

[ADL] Homepage von Advanced Distributed Learning: http://www.adlnet.org/

[AML00] K. Ateyeh, J. A. Mülle, P. C. Lockemann: Modulare Aufbreitung von multimedialen Lerninhalten für eine heterogene Lernumgebung. Tagung "Computergestütztes Kooperatives Lernen (D-CSCL 2000)", S. 267-268, Darmstadt, 2000

[Ann01] Homepage von Annotea: http://www.w3.org/2001/Annotea, W3C

[AQuA] Homepage des Projekts AQuA: http://learning.ipsi.fhg.de, Institut Integrierte Publikations- und Informationssysteme, Fraunhofer-Gesellschaft

[ARIADNE] Alliance of Remote Instructional Authoring & Distribution Networks for Europe (ARIADNE): http://adriadne.unil.ch/metadata

[Aß02] U. Aßmann: Das Web der zweiten Generation – Chancen, Schwächen und Gefahren aus Anwendersicht, Präsentation, Habilitationskolloquium, Universität Karlsruhe, 2002

[Bal98] H. Balzert: Lehrbuch der Software-Technik, Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung, Spektrum Akademischer Verlag GmbH, 2. Auflage, 1998, ISBN: 3827400422

[BHL01] T. Berners-Lee, J. Hendler and O. Lassila: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities, Scientific American, 2001

[BiPe89] T. Biggerstaff, A. Perlis: Software Reusability, Band I und II in der Frontier Series, ACM Press, 1989

[BJR00] G. Booch, I. Jacobson und J. Rumbaugh: UML Distilled Second Edition A Brief Guide to the standard Object Modelling Language, Addison Wesley Longman, 2000, ISBN: 020130998X

[BKW96] A. Brüggemann-Klein, R. Klein und S. Wohlfeil: Pagination Reconsidered, in Proceedings of Electronic Publishing 1996, Cambridge University Press, Cambridge, MA, 1996

[BMRS98] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad: Pattern-orientierte Software-Architektur, ADDISON-WESLEY, 1998, ISBN: 3-89864-142-2

[BoSc97] D. Boles, V. Schnuck: AKI – Ein Annotationssystem zur kooperativen Nutzung gefundener Information im WWW: http://www-is.informatik.uni-oldenburg.de/~dibo/paper/stja97/aki.html, Fachbereich Informatik, Universität Oldenburg,1997

[BoSc00] U.M. Borghoff, J. H. Schlichter: Computer-Supported Cooperative Work – Introduction to Distributed Applications, Springer, 2000, ISBN: 3540669841

[BrDu04] B. Brügge, A. H. Dutoit: Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java, Pearson Studium, 2004, ISBN: 3827370825

[Brü98] T. Brückner: Ein WWW-basiertes Lernsystem zum Thema „Internet“, Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe, 1998

[BSCW] Basic Support for Cooperative Work (BSCW)-Homepage: http://bscw.fit.fraunhofer.de/index.html

[Bür99] M. Bürger: Unterstützung von Awareness bei der Gruppenarbeit mit gemeinsamen Arbeitsbereichen, Dissertation, Lehrstuhl für Angewandte Informatik / Kooperative Systeme, TU München, 1999

[CaHo01] J. Caumanns, S. Hollfelder: Web-Basierte Repositories zur Speicherung, Verwaltung und Wiederverwendung multimedialer Lernfragmente, 23. DGI Online Tagung, Frankfurt, 2001

[CAL] Homepage des Projekts CAL – Call A Lecture: http://www11.in.tum.de/proj/cal/index.html, Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Technische Universität München

[Cok] Homepage der wissensmanagement/community-of-knowledge: http://www.c-o-k.de/

[CPR99] R. M. Carro, E. Pulido, P. Rodriguez: TANGOW: Task-based Adaptive learNer Guidance On the WWW: http://wwwis.win.tue.nl/asum99/carro/carro.html, Universidad Autónoma de Madrid, 1999

[CSCW] Homepage der Fachgruppe CSCW der Gesellschaft für Informatik: http://wwwfgcscw.in.tum.de/index.html

[CSS] Homepage der W3C, Cascading Style Sheets (CSS): http://www.w3.org/Style/CSS/

[CVS] Homepage CVS - Concurrent Versions System : http://www.nongnu.org/cvs/

[Dav96] J. Davis: Homepage von CoNote, http://dri.cornell.edu/Public/davis/Annotation.html, 1996

[DDLFH90] S. Deerwester, S. Dumais, T. Landauer, G. Furnas und R. Harshman: Indexing by Latent Semantic Analysis, Journal of the American Society for Information Science, 41(6), 1990

[Die03] A. Dieckmann: Generische E-Learning-Plattform für interaktive Lehrsimulationen zum Einsatz in Selbststudium und Präsenzlehre online und offline, Dissertation, Technische Fakultät, Universität Bielefeld, 2003

[DIN95] ISO 8042: Qualitätsmanagement – Begriffe. Beuth, Berlin, 1995

[DoBi02] P. Dolog, M. Bielikova: Towards Variability Modelling for Reuse in Hypermedia Engineering: http://www.fiit.stuba.sk/~bielik/publ/abstracts/2002/adbis2002.pdf, Department of Computer Science and Engineering, Slovak University of Technology, 2002

[Don01] M. Donoser: Annotationen in HTML-Dokumenten und On-the-fly Modifikationen von HTML-Dokumenten mit Hilfe eines Proxys, Seminar-/ Projektarbeit, Institute for Information Systems and Computer Media (IICM), Technischen Universität Graz, 2001

[DTD] Homepage der W3C, Document Type Definition (DTD): http://www.w3.org/TR/ REC-html40/ sgml/dtd.html

[Dub a] Open forum to develop the Dublin Core metadata standard: http://dublincore.org/

[Dub b] Dublin Core Metadata Element Set, Version 1.1: http://dublincore.org/documents/ dces/, Reference Description, 2003

[Fia02] Z. Fiala: Ein Komponentenbasierter Ansatz für adaptive, dynamische Web-Dokumente, GK-Workshop in Lohmen, 2002

[FiWe02] C. Fillies, F. Weichhardt: Warum wird das Semantic Web im Knowledge Management noch so wenig beachtet? http://www.semtalk.com/pub/Knowtech2002frauke.htm, Semtation GmbH, Beratung im Netz, 2002

[FM] Formales Modell, Homepage der Requirements Engineering: http://www.requirements-analysis.info/tmp/Uberblick/6-infmodell.pdf

[FMRSA03] D. Feuerhelm, O. Mehl, T. Rudin, J. Steinwand, S. Abeck: Eine dienstorientierte Architektur für den Internet-basierten Wissenstransfer Cooperation & Management: http://www.cm-tm.uka.de/publikationen/paper/kivs2003_dienstorientierte_architektur.pdf, Institut für Technik, Universität Karlsruhe

[FrSc03] D. Frohberg, G. Schwabe: Der mobile Mehrwert von Annotationen im mCSCL, im Rahmen des EU-Projekts MOBILEARN: www.mobilearn.org, 2003

[Für02] A. Fürbach: Entwicklung eines Systems zur Erstellung von wiederverwendbaren Lehr-/ Lerninhalten im Projekt SCORE, Studienarbeit, Institut für Programmstruktur und Datenorganisation, Universität Karlsruhe, 2002

[GHJV96] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Entwurfsmuster, Elemente wiederverwendbarer objektorientierter Software, Addison Wesley, 1996, ISBN 3-85409-179-6

[GaMü03] B. Gartaldeen, S. Münzer: Online-Feedback und Auswertung für E-Learning, Tagungsband der 1.E-Learning Fachtagung Informatik DeLFI, Garching bei München, 2003

[GKS99] A. Geyer-Schulz, S. Koch and G. Schneider: Virtual Notes: Annotations on the WWW for Learning Environments, Proceedings of the Fifth Americas Conference on Information Systems (AMCIS 1999), pp. 136-138, Milwaukee, WI , 1999

[Gri98] F. Griffel: Componentware, Konzept und Techniken eines Softwareparadigmas, dpunkt Verlag, 1. Auflage, 1998, ISBN: 3932588029

[Har00] A. Harrer: Adaption von Benutzerschnittstellen in verteilten intelligenten Lehrsystemen, 10. Arbeitstreffen der GI-Fachgruppe, „Intelligente Lehr-/Lernsysteme, Hamburg, 2000

[HaSt02] S. Handschuh, S. Staab: Authoring and Annotation of Web Pages in CREAM, in Proceedings International WWW Conference(11), Honolulu, Hawaii, USA, 2002

[HeHe00] J. Heflin, J. Hendler: Semantic Interoperability on the Web: http://www.cs.umd.edu/projects/plus/SHOE/pubs/extreme2000.pdf, University of Maryland, 2000

[Hel02] M. Helfert: Planung und Messung der Datenqualität in Data-Warehouse-Systemen, Dissertation, Universität St. Gallen, 2002

[Hod00] W. Hodgins: “Into the future”: http://www.learnativity.com/download/MP7.PDF, Online Book, 2000

[Hof02] H. Hoffmann: „Leitfaden zur Erstellung von Inhalten New Economy“. Im Rahmen des Projekts CeDiS New Economy; http://www.dialekt.cedis.fu-berlin.de/neweconomy/Dokumente/Leitfaden NewEconomy_020102.pdf, Berlin 2002

[Hof03] M. Hoffmann: Semantische Annotation von Lehrmaterialien, Bachelorarbeit, Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Institut für Informatik, TU München, 2003

[Hol01] A. Holzinger: Interoperabilität und Metadaten, Workshop am 2. Business Meeting „Forum Neue Medien“, Wien, 2001

[Hub] S. Huber: Die Theorie sozialer Systeme und das Internet, Kapitel 5 USENET als soziales System, Magisterarbeit im Fach Soziologie, Uni Augsburg

[Hyperwave] Homepage der Hyperwave AG: http://www.hyperwave.com

[HYTIME] HyTime Users’ Group Home Page: http://www.hytime.org/

[IM01] Metadaten für Content-Indizierung und Wissenssicherung , Teil 1: Homepage des Internet Management, http://www.internetmanagement.ch/index.cfm/fuseaction/shownews/newsid/351/, Online-Artikel, 2001

[IMS a] Homepage von IMS Global Learning Consortium: IMS Learning Resource Metadata Best Practices and Implementation Guide v1.1, http://www.imsproject.org/metadata/mdbestv1p1.html#LOM

[IMS b] IMS Learning Resource XML Binding Specification, Version 1.1 - Final Specification: http://www.imsproject.org/metadata/ mdbindv1p1.html, IMS Global, Learning Consortium 2000

[Inm96] W. H. Inmon: Building the Data Warehouse. John Wiley & Sons, New York, 2nd Edition, 1996

[JGJ97] I. Jacobson, M. Griss und P. Jonsson: SOFTWARE REUSE-Architecture, Process und Organization to Business Success. ACM Press/Addison-Wesley, 1997, ISBN: 0201924765

[JöSc99] T. Jörding, M. Schmidt: Unterstützung eines effektiven und attraktiven Designs bei der Generierung adaptiver Produktpräsentationen im World Wide Web: http://www-mmt.inf.tu-dresden.de/projekte/TELLIM/Members/Joerding/abis99/EndPaper/joerding_9/joerding_9.html, Fachgebiet Multimediatechnik, Institut Softwaretechnik II, Fakultät Informatik, Technische Universität Dresden, 1999

[Ker98] M. Kerres: Multimediale und telemediale Lernumgebungen, Konzeption und Entwicklung, , R. Oldenbourg Verlag München Wien, 1998, ISBN: 3-486-24539-2

[KiHe] A. Kienle, T. Herrmann: Kommunikationsunterstützung in kollaborativen Lernprozessen, Erfahrung mit der Lernumgebung KOLUMBUS: http://eldorado.uni-dortmund.de:8080/unistruktur/uebergreifendes/lehrenlernen/kienle/Kiehnle_Herrmannunt.pdf, Fachgebiet Informatik und Gesellschaft, Universität Dortmund

[KlRe97] J. Kleinz und K. Reichenberger: APALO – Ein Modell typografischer Gestaltung und seine Implementierung, in Technische Information in Elektronischen Medien, Fachtagung T:I:E:M, 1997

[Kla04] R. Klamma: Mutimedia Applikationen, Hypertext und Hypermedia, Vorlesungsskript, TU Chemnitz, 2004

[KLTWO04] L. Kornelsen, U. Lucke, D. Tavangarian, M. Waldhauer, N. Ossipova: Strategien und Werkzeuge zur Erstellung multimedialer Lehr- und Lernmaterialien auf Basis von XML, Tagungsband der 2.e-Learning Fachtagung Informatik DeLFI 2004, Paderborn, 2004

[KoKo97] M. Koch, J. Koch: Using Component Technology for Group Editors - The Iris Group Editor Environment. Proceedings of Workshop on Object Oriented Groupware Platforms, Lancaster, UK, G. H. ter Hofte, H. J. van der Lugt (eds.), 1997

[Kre02] K. Kreulich: Generische Bücher – ein graphentheoretisches Modell zur logischen Strukturierung von Büchern in On-Demand-Publikationsprozessen, Dissertation von der Fakultät für Maschinenbau und Verfahrenstechnik, Technische Universität Chemnitz, 2002

[KuWö02] M. Kuschke, L. Wölferl: Web Services kompakt, Spektrum Akademischer Verlag GmbH, 2002, ISBN 3-8274.1375-3

[LaL95] D. LaLiberte: Hypernews: http://union.ncsa.uiuc.edu/HyperNews/get/hypernews.html, 1995

[LHG02] A. Löser, M. Hoffman, C. Grune: Projekt “New Economy” Lernobjekte und ihre Metadaten, technischer Report, CIS TU Berlin und CeDiS FU-Berlin, 2002

[LLTZ04] G. Lortal, M. Lewkowicz, A. Todirascu, M. Zacklad : A web-based Application to support Collaboration through Annotation: http://www.ii.uam.es/~rcarro/AHCW04/Lewkowicz.pdf, Technology University of Troyes, Frankreich, 2004

[LMML05] Homepage von LMML: http://www.lmml.de/, Institut für Informationssysteme und Softwaretechnik, Universität Passau, 2005

[Lob01] H. Lobin: Informationsmodellierung in XML und SGML, Springer 2001, ISBN 3-540-65356-2

[LOM] LOM-Arbeitsgruppe des IEEE, Draft Standard for Learning Object Metadata: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf, 2002

[LTSC] Homepage des Learning Technology Standardization Committee des IEEE: http://ltsc.ieee.org/

[Mai04] R. Maier: Knowledge Management Systems Second Edition, Springer, Berlin, 2004, ISBN: 3-540-20547-0

[MBGLU] E. Melis, J. Büdenbender, G. Goguadze, P. Libbrecht, C. Ullrich: Knowledge Representation and Management in ACTIVEMATH: http://www.activemath.org/publications/Knowledge_Representation_in_ActiveMath.pdf, DFKI Saarbrücken

[McL97] T. McLellan: An Introduction to Usenet News: http://www.islandnet.com/~tmc/html/articles/usentnws.htm

[MITO] Homepage der MITO-Lernobjekte und Metamodell: http://teamschlichter.informatik.tu-muenchen.de/learningobjects/

[Mon04] C. Montandon: Standardisierung im e-Learning: Eine empirische Untersuchung an Schweizer Hochschulen, Institut für Wirtschaftsinformatik der Universität Bern Arbeitsbericht Nr. 161, 2004

[Moti] Motion Tracking Tutorial: http://www.wave-report.com/tutorials/MoTrak.htm

[Nac99] K. Naceur: Annotationssysteme für adaptive Hyperbooks, Diplomarbeit, Institut für Rechnergestütze Wissensverarbeitung, Universität Hannover, 1999

[NoTa95] I. Nonaka, H. Takeuchi: The knowledge creating company, Oxford University Press, 1995, ISBN: 0-19-509269-4

[OMG] Homepage der Object Management Group: http://www.omg.org/, OMG Unified Modeling Language Specification, Version 1.4

[OWL] Homepage der W3C, Web Ontology Language: http://www.w3.org/TR/2004/REC-owl-features-20040210/

[RaDo00] E. Rahm, H. H. Do: Data Cleaning: Problems and Current Approaches, University of Leipzig, IEEE Bulletin of the Technical Committee on Data Engineering, Vol 23 No. 4, Dezember 2000

[RDF] Homepage der W3C, Resource Description Framework: http://www.w3.org/RDF/

[Rie96] A. Riel: Object-Oriented Design Heuristics, Addison Wesley, April 1996. ISBN: 020163385X

[RLMKJ03] H. Reiterer, T. Limbach, F. Müller, P. Klein, Christian Jetter: Ein visueller Metadaten Browser für die explorative Erkündigung großer Datenmengen, Tagungsband Mensch und Computer, Stuttgart, 2003

[RMS00] P. Reimann, K. Müller, P. Starkloff: Kognitiv kompatibel Wissensmanagement: Brückenschlag zwischen Technik und Psyche, ct 4/2000, S.275

[RMW97] Martin Röscheisen, Christian Mogensen, Terry Winograd: Shared Web Annotations As A Platform for Third-Party Value-Added Information Providers: Architecture, Protocols, and Usage Examples, Technical Report CSDTR/DLTR , Computer Science Department, Stanford University, Stanford, CA 94305, U.S.A.

[RuFl04] M. Rust, G. Flach: Integration von Wissensmanagement und eLearning im Rahmen der WIESEL-Frameworkarchitektur, Tagungsband zum 16 GI-Workshop über Grundlagen von Datenbanken, Düsseldorf: Heinrich-Heine-Universität, S. 98-102, 2004

[SAB94] G. Salton, J. Allan und C. Buckley: Automatic structuring and retrieval of large text files, Communications of the ACM, 37(2), 1994

[San96] R.S. Sandhu: Role Hierarchies and Constraints for Lattice-Based Access Controls“, Proceedings of the European Symposium on Research in Security and Privacy, Rom, Italien, 1996

[Schi04] U. Schinz: Semantic Web Technologie und Einsatzgebiete: http://www.dbis.informatik.hu-berlin.de/dbisold/lehre/SS04/metadaten/t3_Schinz.pdf, Berlin, 2004

[Schö01] J. Schönfeld: LOM-IMS-Adriadne Metadaten für web-basierte Lernen: http://cis.cs.tu-berlin.de/Lehre/WS-0001/Sonstiges/wbl-pages/Material/ausarbeitungen/sem_ws00-01_rk_jc_wbt_metadaten.pdf, Technische Universität Berlin, 2001

[Schö00] U. Schöning: Logik für Informatiker, 5 Auflage, Spektrum Akademischer Verlag, ISBN: 3827410053

[Schu03] R. Schulmeister: Lernplattformen für das virtuelle Lernen, Evaluation und Didaktik, München, Wien, Oldenburg: 2003, ISBN 3-486-27250-0

[Schü02] F. Schütz: Annotations: Though Standard in conventional Learning, a stepchild of E-Learning, Proc. of ICTE2002 (Antonio Mendez Vilas, J.A. Mesa Gonzalez, eds.), Government of Extremadura, Badajoz, Spanien, 2002

[See00] S. Seehusen: InGel: Dynamische Informationsgewinnung für Lerneinheiten, Tagungsband „Computergestütztes Kooperatives Lernen (D-CSCL 2000), Darmstadt, 2000

[SemDok04] Das DFG Projekt “Semantik generischer Dokumentstrukturen“: http://www.uni-giessen.de/germanistik/ascl/dfg-projekt/werkstatt.html, Forschergruppe Texttechnologische Informationsmodellierung, Universität Giessen, 2004

[SHOE] Homepage von SHOE: http://www.cs.umd.edu/projects/plus/SHOE/, Parallel Understanding Systems Group Department of Computer Science, University of Maryland at College Park

[SICT] Strategie für die Standardisierung der Informations- und Kommunikationstechnik (ICT) – Deutsche Positionen, Version 1.0, DIN Deutsches Institut für Normung e.V. Strategieausschuss für die Standardisierung in der Informations- und Kommunikationstechnik

[Sim02] J. E. Simpson: XPath and Xpointer, First Edition, O’Reilly, 2002, ISBN: 0-596-00291-2

[SiteMinder] Homepage von Computer Associates: http://www.ca.com/

[Spe98] M. Specht: Adaptive Methode in Computerbasierten Lehr/Lernsystemen, Dissertation des Fachbereiches Psychologie der Universität Trier, 1998

[SSS01] L. Stojanovic, S. Staab, R. Studer: eLearning based on the SemanticWeb: http://www.aifb.uni-karlsruhe.de/WBS/Publ/2001/WebNet_lstsstrst_2001.pdf, Institut AIFB, Universität Karlsruhe, 2001

[SSN03] R. Studer, H.-P. Schnurr, A. Nierlich: TIME2Research Ein Wissensportal für den Unternehmensanalysen: http://www.aifb.uni-karlsruhe.de/WBS/cte/html/publications/pdf/Tempich_AIFB_KarlsruheWI2003.pdf, Institut AIFB, Universität Karlsruhe und Ontoprise GmbH, Karlsruhe, 2003

[Stei01] U. Steinmann: Anwendungsentwicklung für die Aus- und Weiterbildung auf Basis eines Hypermedia-Systems, Dissertation, Fachbereich Elektrotechnik und Informationstechnik der FernUniversität in Hagen, 2001

[SüFr99] C. Süß, B. Freitag: Wiederverwendung von Inhalten und Strukturen offener Lernsysteme, Tagungsband 9. Arbeitstreffen GI-FG Magdeburg, 1999

[SüFr01] C. Süß, B. Freitag: Learning Material Markup Language LMML: http://www.ifis.uni-passau.de/publikationen/reports/ifis200103.pdf, Lehrstuhl für Informationsmanagement, Universität Passau, 2001

[SW a] Homepage der W3C, Semantic Web: http://www.w3.org/2001/sw/

[SW b] Semantic Web Community Portal: http://semanticweb.org/

[TARGETEAM] Homepage des Projektes Targeteam: http://www11.in.tum.de/forschung/projekte/targeteam/index.html.de, Institut für Informatik, Technische Universität München

[TeBr02] G. Teege, P. Breitling: Targeteam: Adaptierbare Lehrinhalte auf Basis von XML und XSLT: http://inf3-www.informatik.unibw-muenchen.de/institut/personen/teege/extension/publications/Teege2002b.pdf, Universität der Bundeswehr München, Technische Universität München, 2002

[Tee04] G. Teege: Targeteam 0.5.7: http://inf3-www.informatik.unibw-muenchen.de/research/projects/targeteam/targeteam-0.5.7/doc/targeteam.pdf, Institut für Informatik, Technische Universität München und Institut für Informationstechnische Systeme Universität der Bundeswehr München, 2004

[Tho03] L. Thomas: Qualität in eLearning, Aktueller Stand und Perspektiven, Tagungsband der 1.e-Learning Fachtagung Informatik DeLFI 2003, München

[ToD] Homepage des Projekt Teachware On Demand: http://www.ipsi.fraunhofer.de/oasys/projects/teachware/index_e.html, Institut für integrierte und Informationssysteme, Fraunhofer-Gesellschaft

[Toz99] G. Tozer: Medatada Management for Information Control and Business Success, Artech House Computer Library, 1999, ISBN: 0-89006-280-3

[Tra00] S. Trahasch: Ariadne - Digitale Bibliothek für die (virtuelle) Hochschule Virtuelle Hochschule VIROR, Vorträge auf dem 2. BSZ-Kolloquium am 09.-10.10.2000 in Konstanz, 2000

[TSMB95] S. Teufel, C. Sauter, T. Mühlherr und K. Bauknecht: Computerunterstützung für die Gruppenarbeit, Addison-Wesley, 1995, ISBN: 3-89319-878-4

[URI] Homepage der W3C, Naming and Addressing: http://www.w3.org/Addressing/

[USENET] Homepage der Wikipedia: http://de.wikipedia.org/wiki/Usenet

[Ves04] F. Vester: Denken, Lernen, Vergessen, Was geht in unserem Kopf vor? Wie lernt das Gehirn? Wann lässt es uns im Stich?, dtv Taschenbuch, 30. Auflage, 2004, ISBN: 3-423-33045-7

[Völ02] M. Völkel: Extraktion von XML aus HTML-Seite Das WYSIWYG-Werkzeug d2c, Diplomarbeit, Fakultät für Informatik, Universität Karlsruhe, 2002

[Wall90] E. Wallmüller: Software-Qualitätssicherung in der Praxis. Hanser, München; Wien, 1990, ISBN: 3-446-15846-4

[WaSt96] R. Y. Wang, D. M. Strong: Beyond Accuracy: What Data Quality Means to Data Consumers. In: Journal of Management Information Systems 12 (1996) 4, S. 5-331, 1996

[Wiki] Linguistik-Portal der Computerlinguistik an der Universität Trier: http://www.uni-trier.de/uni/ fb2/ldv/ldv_wiki/index.php/Constraint

[Wiki a] E-Learning- Wikipedia: http://de.wikipedia.org/wiki/E-Learning

[XALAN] Homepage Apache XML-Projekte: http://xml.apache.org/

[XML] Homepage der W3C, Extensible Markup Language (XML): http://www.w3.org/ XML/

[XML Schema] Homepage der W3C, XML Schema: http://www.w3.org/XML/Schema

[XPATH] Homepage der W3C, XML Path Language (XPath): http://www.w3.org/TR/xpath, Version 1.0, 1999

[XSLT] Homepage der W3C, XSL Transformations (XSLT): http://www.w3.org/TR/xslt, Version 1.0, W3C Recommendation 16, November 1999

[Xu00] C. Xu: Interaction and Cooperation Mechanisms for Distributed Communities and Groups in Educational Settings, Dissertation, Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Institut für Informatik, Technische Universität München, 2000

Abbildungsverzeichnis

Tabellenverzeichnis

Index