UNIVERSITÄT DORTMUND

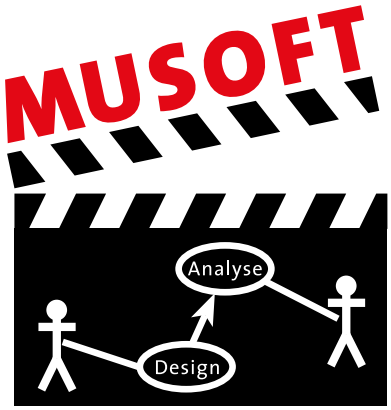
Fachbereich Informatik
Lehrstuhl für Software-Technologie

M E M O Nr. 148

MuSofT Bericht Nr. 5

Abschlussbericht des Projektes „MuSofT - Multimedia in der SoftwareTechnik“

Ernst-Erich Doberkat Gregor Engels Corina Kopka (Hrsg.)



April 2004

Internes Memorandum des
Lehrstuhls für Software-Technologie
Prof. Dr. Ernst-Erich Doberkat
Fachbereich Informatik
Universität Dortmund
Baroper Straße 301

D-44227 Dortmund

ISSN 0933-7725

Abschlussbericht des Projektes “MuSofT– Multimedia in der SoftwareTechnik”

Ernst-Erich Doberkat Gregor Engels (Hrsg.) Corina Kopka

April 2004

Vorwort

Das Projekt *MuSofT – Multimedia in der Softwaretechnik* wurde im Rahmen des Programmes Neue Medien in der Bildung seit Anfang März 2001 vom Bundesministerium für Bildung und Wissenschaft gefördert und endete im Dezember 2003. Das Projekt hat sich zum Ziel gesetzt, die Ausbildung im Bereich der Softwaretechnik in den Hochschulen durch den Einsatz Neuer Medien insbesondere in der Präsenzlehre zu unterstützen und zu verbessern. Mit diesem Abschlussbericht wollen wir die Projektergebnisse dokumentieren.

Für die Belange von MuSofT, die über die Verantwortlichkeit einzelner Teilprojekte hinausgehen, haben wir von Projektbeginn an Koordinationsteams geplant und eingesetzt. Die Aufgaben dieser über die einzelnen Standorte hinweg tätigen Koordinationsteams mit den Namen KT1 bis KT6 lassen sich wie folgt charakterisieren:

KT1 erarbeitete einheitliche Richtlinien für die didaktische Konzeption von Lerneinheiten.

KT2 befasste sich mit der inhaltlichen und stilistischen Abstimmung von Lerneinheiten.

KT3 hatte die Aufgabe, die Einsetzbarkeit der erstellten Materialien in anderen Studiengängen als der Kerninformatik, insbesondere auch in Ingenieurstudiengängen, zu untersuchen.

KT4 ist für die Bereitstellung eines Internetportals zuständig gewesen, über welches die erstellten Lerneinheiten und die dazugehörigen Werkzeuge angeboten werden.

KT5 hatte Medienproduktionen koordiniert, die über Teilprojektgrenzen hinweg möglich sind.

KT6 hatte sich mit den Grundlagen für die Nachhaltigkeit von MuSofT beschäftigt.

KT5 und KT6 haben wir im Laufe des Jahres 2002 zusätzlich zu KT1 bis KT4 gebildet, um neu aufgekommene Fragestellungen zu bearbeiten. Von den sechs Koordinationsteams hat KT3 seine Arbeit bereits abgeschlossen. Die übrigen noch aktiven fünf Teams stellen ihre Ergebnisse in dieser Sammlung vor.

Eine weitere teilprojektübergreifende Fragestellung beschäftigt sich mit Gender Mainstreaming für MuSofT. Diese umfassenden Aufgaben haben wir an externe Expertinnen vom Hochschuldidaktischen Zentrum der Universität Dortmund unter der Leitung von Frau Prof. Dr. Sigrid Metz-Göckel vergeben.

Wir möchten nun abschließend allen beteiligten Kolleginnen und Kollegen dafür danken, dass sie die Beiträge für diesen Projektbericht geschrieben und begutachtet haben. Dr. Klaus Alfert hat das Projekt MuSofT bis zum 31. Dezember 2003 als Projektmanager begleitet. Ihm sei für seine Arbeit an dieser Stelle herzlich gedankt.

Dortmund und Paderborn,
im April 2004

Corina Kopka
Prof. Dr. Ernst-Erich Doberkat
Prof. Dr. Gregor Engels

Inhaltsverzeichnis

Autorinnen und Autoren

Prof. Dr. Gregor Engels
Dipl.-Inform. Jan Hendrik Hausmann
Dipl.-Inform. Marc Lohmann
Dr. Annika Wagner Arbeitsgruppe Informationssysteme
Fachbereich Mathematik/Informatik
Universität Paderborn

Dipl.-Inform. Dirk Jesko
Institut für Technische und
Betriebliche Informationsysteme
Otto-von-Guericke-Universität Magdeburg

Prof. Dr. Johannes Magenheim
Olaf Scheel
Arbeitsgruppe Didaktik der Informatik
Fachbereich Mathematik/Informatik
Universität Paderborn

Dr. Klaus Alfert
Prof. Dr. Ernst-Erich Doberkat
Dipl.-Inform. Corina Kopka
Dipl.-Inform. Jörg Pleumann
Dipl.-Inform. Jens Schröder
Lehrstuhl für Software-Technologie
Fachbereich Informatik
Universität Dortmund

Dipl.-Psych. Dalinda Irmscher-Lecon
Prof. Dr. Silke Seehusen
Fachbereich Elektrotechnik
Fachhochschule Lübeck

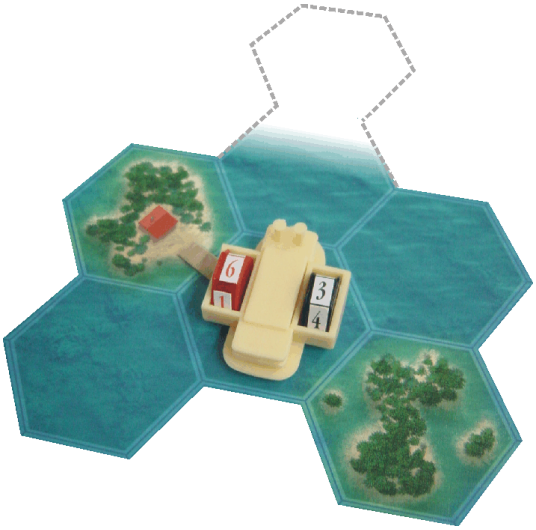
Dipl.-Inform. Peter Aschenbrenner
Prof. Dr. Andy Schürr
FG Echtzeitsysteme, FB 18
TU Darmstadt

Anita Böhm
Alexander Lurk
Andreas Piater
Prof. Dr.-Ing. Fritz Schmidt
Darko Sucic
Institut für Kernenergetik und Energiesysteme
Universität Stuttgart

Prof. Dr. Udo Kelter
Praktische Informatik / Softwaretechnik
Fachbereich Elektrotechnik und Informatik
Universität-Gesamthochschule Siegen

Der MuSofT-Abschlussbericht von TP 1.1

Gregor Engels, Jan Hendrik Hausmann, Marc Lohmann



Im Rahmen des Projekts MuSofT (Multimedia in der Softwaretechnik) war es die Aufgabe des Teilprojekts 1.1. Lehreinheiten zum Thema Videogestützte Anforderungsdefinition zu erzeugen. Hierzu wurden Lehreinheiten geplant, realisiert und im universitären Kontext eingesetzt. Besonderes Element dieser Lehreinheiten ist die Verwendung von Videos zum Dokumentieren einer Problemdomäne.

Inhaltsverzeichnis

1 Einleitung

Dieser Bericht beschreibt die Arbeit des MuSofT-Teilprojekts 1.1, dessen Aufgabe die Erstellung von Lehreinheiten zum Thema Videogestützte Anforderungsdefinition war. Die vorangegangenen Jahresberichte [DE02, ADE03] zeigten dabei die konzeptionellen Grundlagen der Einheit auf und stellten erste Realisierungserfolge sowie den ersten Einsatz der Materialien vor. Im zweiten Jahresbericht [ADE03] wurde insbesondere dargestellt, welche Erkenntnisse aus dem initialen Einsatz der Materialien gewonnen werden konnten und wie diese in einer Ergänzung und Verbesserung der Lehreinheit resultierten. Die Aufgaben des dritten Jahres, über das hier schwerpunktmäßig berichtet werden soll, konzentrierten sich auf die Aufbereitung der Lehreinheit in einer zur Weitergabe geeigneten Weise. Da diese Weitergabe das erklärte Projektziel von MuSofT ist, ist es unserer Ansicht nach unumgänglich, hier Vorkehrungen zu treffen, die anderen Lehrenden das Einsetzen unserer Materialien erleichtern bzw. erst ermöglichen. Dazu gehören insbesondere die Klärung der Nutzungsrechte, die technischen Aufbereitung und die Bereitstellung von begleitender Dokumentation zur Vorbereitung der Lehrenden.

Im Rahmen des Teilprojekts wurde die Lehreinheit „Videogestützte Anforderungsdefinition“ für den Einsatz im Grundstudium und die Lehreinheit „Zielorientierte Anforderungsdefinition“ für den Einsatz im Hauptstudium konzipiert und realisiert.

2 Vorstellung der Lehreinheiten

Die Lehreinheiten zur Anforderungsdefinition sollen Probleme aufzeigen, die typischerweise beim Erfassen, Strukturieren und Spezifizieren der Anforderungen von Benutzern an ein neues Softwaresystem entstehen. Sie sollen darüber hinaus den Studierenden die Fähigkeiten vermitteln, diesen Problemen in einem praktischen Kontext zu begegnen.

Die Lehreinheiten richten sich an zwei unterschiedliche Zielgruppen. Einerseits sollen Studierende ohne Vorkenntnisse im Bereich des Software Engineering an die Problematik der Anforderungsdefinition herangeführt werden und Grundkenntnisse erwerben, andererseits sollen aber auch Vertiefungen für Studierende mit eben diesen Grundkenntnissen angeboten werden.

Für diese Ausrichtung der Lehreinheiten auf zwei Zielgruppen mussten wir definieren, was unter Grundkenntnissen der Anforderungsdefinition zu verstehen ist. Dafür haben wir eine Gewichtung der in den vorangegangenen Jahresberichten dargestellten inhaltlichen Lernziele vorgenommen. Danach ist im Rahmen der Vermittlung von Grundkenntnissen der Anforderungsdefinition folgendes zu leisten:

- Den Studierenden soll die *Notwendigkeit einer Domänenanalyse* bewusst gemacht werden. Grundlegende Techniken zur Beschreibung des Ergebnisses der Domänenanalyse sollen vermittelt werden. Techniken der Informationsbeschaffung bleiben unberücksichtigt.

- *Methoden zur Spezifikation* von Anforderungen (insbesondere unter Einsatz von Diagrammen der Unified Modeling Language UML) sollen vermittelt werden.

- Durch eine explizite Anleitung sollen die Studierenden an die *Dokumentation* ihrer Entscheidungen herangeführt werden. Eine eigene Abschätzung wie viel und welche Art von Dokumentation an einer Stelle sinnvoll ist, wird nicht erwartet.

- Den Studierenden soll die *Notwendigkeit einer Spezifikationsanalyse*, also die (kritische) Reflexion der formal spezifizierten Modelle, bewusst gemacht werden. Inhaltlich werden im Rahmen dieser Analyse eher typische studentische Fehler diskutiert als die Probleme einer realen Spezifikationsanalyse. Aufwands- und Risikobewertungen bleiben daher vollständig unbehandelt.

Diese Grundkenntnisse sollen in der Lehreinheit „Videogestützte Anforderungsdefinition“ vermittelt werden, auf deren Entwicklung der Schwerpunkt unserer Arbeit lag. Diese Lehreinheit umfasst dabei Vorlesungsfolien sowie Übungsmaterialien (Haus- und Präsenzarbeitsaufgaben), wobei in beiden Lehrformen multimediale Elemente (Videos/Animationen) die zu spezifizierende Domäne darstellen. Zur Unterstützung der Lehrenden werden daneben Musterlösungen zu allen Übungsaufgaben sowie eine Mittschnitt eines beispielhaften Vortrages dieser Folien bereitgestellt.

Das Thema der aufbauenden Lehreinheit lautet „Zielorientierte Anforderungsdefinition“. Dieses Thema ist als aufbauendes Thema geeignet, da es ein grundsätzliches Verständnis für die Problematik der Anforderungsdefinition voraussetzt, darüber hinaus jedoch mit neuen Ideen und Aspekten eine intensivere Auseinandersetzung mit der Problematik erlaubt. Das Thema ist ein aktuelles Forschungsthema, welches sowohl im Bereich des Requirements Engineering, aber auch in allgemeinen Softwaretechnik-Communities zunehmend Interesse und Verbreitung findet. Wir gehen daher davon aus, dass eine Lehreinheit zu diesem Thema in Zukunft für verschiedenen Lehrenden von Interesse sein wird. In Bezug auf die im Jahresbericht 2001 [DE02] identifizierten Lernziele unterstützt diese Einheit besonders:

- **Zieldefinition.** Dies ist ein zentrales Konzept der zielorientierten Anforderungsdefinition. Abstrakte Ziele müssen aufgestellt und alternative Realisierungsmöglichkeiten dieser Ziele verglichen werden.

- **Spezifikation.** In der Lehreinheit zielorientierte Anforderungsdefinition sollen Spezifikationstechniken aus dem Bereich der temporalen Logik eingesetzt werden. Die hierfür benötigten Grundkenntnisse werden in der Einheit vermittelt.

- **Spezifikationsanalyse.** Es existieren umfangreiche Techniken, um zielorientierte Anforderungsdefinitionen auf Konflikte, Abhängigkeiten und Lücken hin zu untersuchen. Die Studierenden sollen lernen diese Techniken zu nutzen und die dahinter liegenden Konzepte zu verstehen.

Die Materialien zum Durchführen dieser Lehreinheit umfassen Vorlesungsfolien sowie erste Übungsaufgaben. Auch in dieser Lehreinheit werden Videos zur Dokumentation der Problemdomäne eingesetzt. Es handelt sich hierbei um Videos, die in Kooperation mit der Firma arvato (Bertelsmann Konzern) produziert wurden, und die Problemstellungen aus dem Bereich der Logistik illustrieren.

3 Technische Realisierung

Die technische Realisierung der Lehreinheit lässt sich in verschiedene Punkte untergliedern. Wir stellen daher in den folgenden Abschnitten dar, welche Konzepte den Lehreinheiten zu Grunde liegen, welche Medien zur Realisierung dieser Konzepte dienen, wie sich die Medien in den Lehr-/Lernkontext einbetten und welche Maßnahmen wir getroffen haben, um die nachhaltige Nutzung der Materialien zu ermöglichen.

3.1 Konzeption

Zwei wesentliche didaktische Konzepte bilden das konzeptionelle Fundament der Lehreinheit Videogestützte Anforderungsdefinition: Der *Einsatz von Videos* in der Lehre und das *dokumentenzentrierte Vorgehen*.

Der Einsatz von Videos in der Anforderungsdefinition geht auf die Erkenntnis zurück, dass zur Definition von Anforderungen das Erkennen dieser Anforderungen in einer komplexen Situation gehört. Das Problem ist, dass man in einer realistischen Situation mit einer komplexen betrieblichen Realität konfrontiert wird, einer Reihe unterschiedlicher Gesprächspartner gegenübersteht, die jeweils ihren eigene Sichtweise auf ein zu entwickelndes System wiedergeben und diese Sichtweisen sich dann auch noch häufig widersprechen. Man kann im Allgemeinen nicht davon ausgehen, dass notwendige Informationen eindeutig und vollständig aufbereitet vorliegen. Es ist also die Aufgabe des Softwareingenieurs in dieser Situation Informationen zu ordnen, aufzubereiten und zu analysieren und sich so ein möglichst präzises und konsistentes Bild der Situation zu machen. Dies kann er dann mit geeigneten Modellierungsnotationen festhalten. Klassischerweise wird in der Lehre jedoch vor allem dieser zweite Schritt betont, d.h. es werden Beispiele gewählt, bei denen die Beschreibung klar, eindeutig und vollständig ist und anhand dieser Beispiele wird dann das reine Spezifizieren gelehrt. Wesentliche Probleme der Anforderungserfassung werden dabei ausgeblendet.

Grundidee der Lehreinheiten zur videogestützten Anforderungsdefinition ist es, an dieser Stelle einzugreifen und den Aspekt der Informationsgewinnung und -aufbereitung stärker in der Vordergrund zu stellen. Der Einsatz von Videos unterstützt dieses Ziel in mehrfacher Weise:

- Videos sind multimedial. Sie verknüpfen visuelle und auditive Elemente und können sehr komplexe und detaillierte Informationen parallel präsentieren. Im Vergleich zu einem beschreibenden Text stellen sie viel höhere Anforderungen an die Wahrnehmungs- und Verarbeitungsfähigkeiten der Studierenden. Auch wenn dies noch nicht die volle Komplexität der betrieblichen Informationserfassung darstellen kann, so ist es durch den Einsatz von Videos möglich, eine vielschichtigere Darstellung zu erreichen.

- Durch ihre höhere Informationsdichte ist es möglich, in Videos wesentlich komplexere Beispiele abzubilden, als dies typischerweise in beschreibendem Text möglich ist. Es ist dadurch möglich, auch enger an die Realität angelehnte Beispiele in der Vorlesung einzusetzen.

- Die Reproduktion von Videos schaltet den Lehrenden und seine Vermittlungsfunktion aus. Häufig befinden sich Lehrende beim Präsentieren von Beispielen zur Anforderungsdefinitionen in einem Zwiespalt: typischerweise ist es ihre Aufgabe, die zu vermittelnden Informationen sinnerschliessend und strukturiert vorzutragen. An dieser Stelle läuft dies jedoch dem Lernziel zuwider, da die Studierenden lernen sollen, auch mit komplexen und nicht vorstrukturierten Informationen zu arbeiten. Der Einsatz von Videos befreit den Lehrenden aus diesem Dilemma. Die Videos repräsentieren alle Informationen, die zum Verständnis des Problembereichs notwendig sind.

- Setzt man weiterhin Videos ein, die nicht spezifisch für diesen Lehrzweck gedreht worden sind, so schließt man sogar aus, dass bei der Erstellung der Videos eine entsprechende Vorstrukturierung stattgefunden hat. Wir haben diesem Gedankengang Rechnung getragen, indem wir existierende Videos eingesetzt haben, die technische Logistiksysteme zu Werbezwecken vorstellen. Diese Videos stehen nicht im Verdacht, irgendeine für eine Anforderungsdefinition geeignete Struktur aufzuweisen. Vielmehr verdeutlicht ihr Einsatz, dass häufig zwar Informationen über den Problembereich vorliegen, diese aber mit einer vollständig anderen Zielrichtung erstellt wurden.

Auf diese Weise unterstützt der Einsatz von Videos die Lehre der Anforderungsdefinition, da er es uns erlaubt, realistischere Beispiele in komplexeren Darstellungen und ohne Vorstrukturierung für unser Lehrgebiet zur Demonstration unserer Techniken zu verwenden.

Als zweites didaktische Grundkonzept der Lehreinheit Videogestützte Anforderungsdefinition haben wir den Ansatz der *Dokumentenzentrierung* gewählt. Aus den Evaluationen früherer Veranstaltungen konnten wir bei den Studierenden eine Desorientierung ob der vielen in der Softwaretechnik eingesetzten Notationen diagnostizieren. Wir haben daher strukturierte Dokumentenvorlagen geschaffen (so genannte Templates), die sowohl als Gliederung der Vorlesungen dienen, als auch den Studierenden bei den selbst durchzuführenden Übungsarbeiten Anleitung und Orientierung bieten. Unser didaktisches Konzept stellt also zu erstellende Dokumente in den Mittelpunkt. Wir bezeichnen es daher auch als dokumentenzentrierten Ansatz für die Ausbildung. Dem Anfänger, der über keinerlei Erfahrung in größeren Softwareentwicklungsprojekten verfügt, bieten sich so wesentlich mehr Orientierungsmöglichkeiten um sich schrittweise einer Lösung anzunähern.

Im Mittelpunkt der Lehreinheit steht das Dokument Pflichtenheft. Das Pflichtenheft umfasst die Ergebnisse der Domänenanalyse ebenso wie die Spezifikation der Anforderungen. Die Studierenden werden dazu angeleitet, ein Pflichtenheft basierend auf einem Template zu erstellen. Dieses Template gibt eine Gliederung für das Pflichtenheft vor und beschreibt zu jedem Abschnitt die Aufgabe, die er zu erfüllen hat. Dadurch wird dem Studierenden ermöglicht nachzuvollziehen, mit welchem *Ziel* er etwas tut. Außerdem enthält es konkrete Arbeitsanweisungen und bietet auf diese Art und Weise dem Lernenden Hilfen an, *wie* bestimmte Aufgaben zu erfüllen sind.

Neben dem Template zur Erstellung eines Pflichtenheftes wird dem Lernenden ein Beispielpflichtenheft zur Verfügung gestellt, das mit Hilfe des Templates für ein komplexeres Projekt erstellt wurde. Dieses Beispielpflichtenheft kann als Illustration der abstrakten Arbeitsanweisungen im Template angesehen werden.

Nach dem gleichen Konzept verfahren wir bei der Vermittlung der Spezifikationsanalyse. Hierfür ist ein eigener Dokumententyp vorgesehen, der Review des Pflichtenheftes. Auch hier wird wieder über ein Template eine praktische Anleitung bereitgestellt, eine solche Analyse durchzuführen, und durch ein Beispieldokument illustriert.

Die Lehreinheit *zielorientierte Anforderungsdefinition* wendet ebenfalls das Konzept der videogestützten Anforderungsdefinition an. Auch hier werden im Rahmen einer komplexen betrieblichen Realität Probleme visualisiert, die mit den Methoden der Zielorientierung bearbeitet werden können. Da die Zielgruppe dieser Vorlesung bereits eine bessere Orientierung im Themengebiet besitzt, ist das dokumentenzentrierte Vorgehen in dieser Lehreinheit nicht notwendig.

3.2 Fallbeispiele

Gemäß den Ausführungen zur videogestützen Anforderungsdefinition im vorherigen Abschnitt kommt der Rolle der gewählten Fallbeispiele im Rahmen unserer Lehreinheiten eine besondere Rolle zu. Optimalerweise sollten hier komplexe realitätsnahe Fallbeispiele ohne Ausrichtung auf die Anforderungsdefinition präsentiert werden. Andererseits ist jedoch auch zu berücksichtigen, dass der eigentliche Lehrinhalt Techniken zur Anforderungsdefinition sind. Die Komplexität der Videos darf daher nicht so hoch sein, dass die Studierenden den Bezug zum eigentlichen Lehrthema nicht mehr erkennen, gleichzeitig sollten die Beispiele jedoch auch genug natürliche Ansatzstellen zur Demonstration der unterschiedlichen Techniken bieten. Wir haben deshalb unterschiedliche Fallbeispiele gewählt, um den verschiedenen Anforderungen entsprechen zu können:

In der Lehreinheit videogestützte Anforderungsdefinition wird in der Vorlesung die Entwicklung eines Steuersystems für ein automatisiertes Lager entwickelt.Die Logistikbranche schien uns besonders geeignet, da hier die ablaufenden Prozesse häufig sehr plastisch dargestellt werden können. Dieses Beispiel ist am stärksten an eine betrieblichen Realität angelehnt, da es beim Einsatz in der Vorlesung durch den Lehrenden vorgestellt und interpretiert wird. Dabei können Unklarheiten geklärt und die Aufmerksamkeit der Studierenden auf bestimmte Aspekte gerichtet werden. Das Beispiel ist komplex genug, um alle Techniken der Anforderungsdefinition daran zu demonstrieren, allein das dazu von uns erstellte Beispielpflichtenheft umfasst fast 50 Seiten Spezifikation.

Diese Komplexität ist jedoch in den Übungen, in denen die Studierenden die selbstständige Erarbeitung eines Pflichtenheftes erlernen sollen, nicht angebracht. Wir haben daher für die Übungen ein Gesellschaftsspiel als Fallbeispiel dokumentiert. Gesellschaftsspiele haben den Vorteil, dass der Problembereich eine klare Begrenzung besitzt (im Gegensatz zu vielen betrieblichen Problemen), dass es gut dokumentiert ist (Spielanleitung) und dass eine leichte Einarbeitung in die Domäne möglich ist. Weiterhin hoffen wir, dass der Einsatz eines Spieles motivierender ist als der von technischen Beispielen. Für die Übungen haben wir daher Animationen erstellt, die das Spiel *Mississippi Queen* darstellen. Aufbauend auf diesem Spiel sollen

die Studierenden Pflichtenhefte für die Erstellung einer elektronischen Version von Mississippi Queen schreiben.

In der Lehreinheit Zielorientierung wird als Fallbeispiel in der Vorlesung der London Ambulance Service verwandt. Es handelt sich dabei um eine reale Softwareentwicklung für die Koordination der Krankenwageneinsätze im Großraum London, die Anfang der Neunziger Jahre durchgeführt wurde. Das Beispiel ist gut dokumentiert, da es sich um ein spektakulär gescheitertes Projekt handelt, das eine öffentliche Untersuchung nach sich zog [FD96]. Aufgrund der dadurch vorliegenden Unterlagen wurde das Fallbeispiel des London Ambulance Service zu einem klassischen Fallbeispiel im Requirements Engineering, an dem viele formale Techniken demonstriert wurden.

In den Übungen zur Zielorientierung sollen die Studierenden lernen, die Techniken der zielorientierten Anforderungsdefinition selbständig auf Beispiele anzuwenden. Als Fallbeispiel haben wir uns hier für Logistikprobleme der arvato AG entschieden. Als industrieller Kooperationspartner des Projekts MuSofT hat uns die Firma arvato dabei Einblick in die internen Betriebsabläufe gewährt. Arvato ist im Bertelsmann-Konzern unter anderem für die Logistik zuständig und betreibt am Standort Gütersloh eins der größten Hochregallager Europas. Von hier werden hauptsächlich Produkte der Medienbranche (also Bücher, CDs, DVDs etc.) sowohl zur Auslieferung an die Filialen des Bertelsmann Clubs als auch an private Besteller vorbereitet. Durch diese Kooperation war es uns möglich, Beispiele zu finden, die sich wiederum in der Logistikbranche bewegen, die wir aber auch gemäß unseren speziellen Anforderungen zuspitzen können.

3.3 Medienerstellung

Zur Verwirklichung des dokumentenzentrierten Ansatzes wurden Vorlesungsmaterialien geschaffen werden, die diesen Ansatz unterstützen. Templates und Beispieldokumente müssen den Studierenden ebenso zur Verfügung gestellt werden wie Aufgabenblätter, die dieses Vorgehen unterstützen. Weiterhin sollte sowohl in der Vorlesung als auch in den Übungen das Vorgehen der Videogestützen Anforderungsdefinition konsistent durchgeführt werden. Im Folgenden werden die Arbeiten zu diesen Themen detaillierter beschrieben:

Für eine Lehrveranstaltung zum Thema Videogestütze Anforderungsdefinition wurde ein *Foliensatz* mit Microsoft Powerpoint entwickelt. Aufgabe dieses Foliensatzes ist es, zur Erstellung der Dokumente anzuleiten und ihren Aufbau zu erläutern. Als illustrierendes Beispiel werden innerhalb des Foliensatzes Ausschnitte aus dem Beispielpflichtenheft verwendet. Der Foliensatz umfasst Folien für fünf 90-minütige Veranstaltungen. Folgende Themen werden behandelt:

- Überblick über das Pflichtenheft
- Modell des Problembereichs (inkl. Einführung UML Objekt- und Klassendiagramme)
- Geschäftsprozessmodellierung (inkl. Einführung UML Use Case- und Aktivitätendiagramme)
- Produktfunktionen
- Qualität des Pflichtenheftes

Abbildung 1: Orientierungsicons

Dabei erläutern die ersten vier Vorlesungen das vorgehen beim Ausfüllen des Pflichtenhefts (bzw. des entsprechenden Templates), während die letzte Vorlesung in den Review des Pflichtenheftes gemäß dem Review-Template einführt.

Ein wesentlicher Punkt bei der Erstellung der Folien war die Etablierung eines einheitlichen, auf MuSofT zugeschnittenen *Layouts*. Dabei sollte auch die Kritik der Studierenden aus einem früheren Einsatz der Folien berücksichtigt werden, wo aufgrund eines dunklen Hintergrunds die Folienprojektion nicht gut lesbar war. Für die Folien wurde ein weißer Hintergrund mit schwarzer Schrift gewählt wobei rote und blaue Elemente als Hervorhebungen eingesetzt werden. Um eine Orientierung in den Folien zu erleichtern und ein einheitliches Aussehen zu erreichen wurde weiterhin eine einheitliche Symbolik etabliert, für die grafische Elemente gestaltet wurden (siehe Abb. 1). Auf diese Weise können Verweise auf Übungen oder externe Literaturquellen schnell in den Folien gefunden werden. Eine Beispielfolie ist in Abb. 2 dargestellt. Das gewählte Layout macht weiterhin den Bezug zum Projekt MuSoft deutlich, indem es das MuSofT Logo als festen Bestandteil in jede Folie einbettet und die Farbwahl sich an den Farben des MuSoft Logos orientiert.

Sowohl Templates als auch Beispieldokumente wurden als Word-Dokumente realisiert. *Templates* arbeiten mit Feldern, die ein leichtes Umsetzen der konkreten Arbeitsanweisungen ermöglichen sollen. Die Beispieldokumente sind durch studentische Mitarbeiter des Projekts erstellt worden, die die entsprechende Lehrveranstaltung direkt vorher gehört hatten. Ihr Wissensstand ist also von dem der Teilnehmer der Lehrveranstaltung noch nicht weit entfernt. Auf diese Art und Weise konnten bei der Erstellung auftretende typische studentische Fehler ermittelt werden. Diese Fehler bilden die Grundlage des Templates für den Review des Pflichtenheftes. In diesem Template werden die Studierenden angeleitet in dem von ihnen angefertigten Pflichtenheft nach typischen Fehlern zu suchen.

Zu den oben beschriebenen Fallbeispiele wurden Videos produziert. Dabei kam eine Reihe unterschiedlicher Vorgehensweisen zum Einsatz. Für die Lehreinheit Videogestützte Anforderungsdefinition sichteten wir einen Bestand an industriellen Werbe- und Anschauungsfilmen, der am Lehrstuhl für Logistik der Universität Magdeburg vorliegt. Unter den gesichteten Videos schien uns ein Video über ein horizontales Karussellager der Firma Mannesmann besonders geeignet. In diesem Film wird der Einsatz eines so genannten Demag Horizontal-Karusselllagers im Kreiskrankenhaus Heidenheim gezeigt. Der eigentlich zu Werbezwecken erstellte Film beschreibt dabei sowohl das Anwendungsgebiet (Arzneimittelverwaltung) als

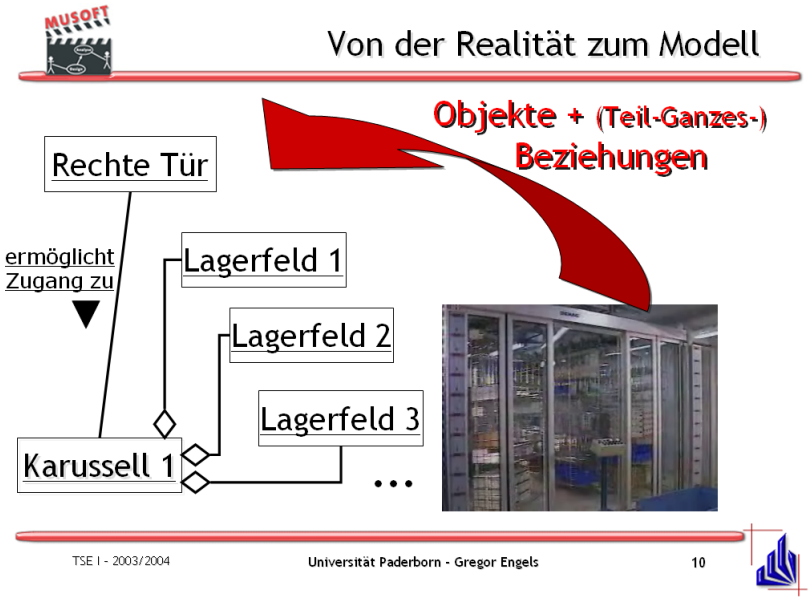


Abbildung 2: MuSofT-Folie aus der Vorlesung TSE I

auch Technik und Bedienung dieser Lageranlage in einigen Details. Die technische Konvertierung und der Schnitt des Videos wurde vom universitätseigenen Audio-Visuellen Medienzentrum (AVMZ) übernommen.

Für die Übungen sollte das Spiel Mississippi Queen durch eine Reihe von Animationsfilmen illustriert werden, die jeweils unterschiedliche Spielsituationen darstellen. Diese Animationen wurden im Rahmen einer Studienarbeit angefertigt. Dabei wurden die einzelnen Elemente der Spieles gescannt bzw. fotografiert und mit Hilfe des Programms Macromedia Director zu Animationsszenen kombiniert. Die so dargestellten Abläufe wurden dann durch einen gesprochenen Text kommentiert. Auf diese Weise sind insgesamt 9 Animationsfilme entstanden, die jeweils partielle Informationen über mögliche Spielverläufe geben (siehe Abb. 3).

Für die Übungen zur Lerneinheit zielorientierte Anforderungsdefinition wurden Videos über Abläufe in der Firma arvato produziert. Um in diesem Umfeld qualitativ hochwertige Aufnahmen zu erhalten, dabei den Betriebsablauf aber so wenig wie möglich zu stören, wurde ein externe Medienproduktionsfirma engagiert, die bereits Erfahrungen in der Arbeit mit und für arvato hat. Dabei konnten wir auch auf bereits vorhandenes Material zurückgreifen, so dass etwa eine Gesamtansicht des Hochregallagers vom Hubschrauber aus einem alten Werbevideo entnommen werden und in unsere Videos integriert werden konnte. In Spielszenen (siehe Abb. 4) werden über die normalen Betriebsabläufe hinaus spezielle Szenarien aufgebaut, die den Studierenden auf das Thema Zielorientierung zugespitzte Probleme präsentieren. Zur Erstellung dieser Spielszenen besuchten wir ein Seminar zum Thema Drehbucherstellung. Wir waren somit in der Lage, unsere Vorstellungen zu präzisieren und der Produktionsfirma

Abbildung 3: Animation des Spiel Mississippi Queen

Abbildung 4: Szene aus dem arvato Video

zu kommunizieren.

3.4 Integration und Nachhaltigkeit

Da das Ziel von MuSofT primär in der Weitergabe von Lehrmaterial an andere Hochschulen zu sehen ist, halten wir es für zentral, die produzierten Medienobjekte so aufzubereiten, dass diese Weitergabe auch praktisch handhabbar wird. Dies umfasst die inhaltliche Einbettung der multimedialen Elemente in eine Lehreinheit, die Dokumentation der Lehreinheit, eine technische Integration und Aufbereitung und die Klärung rechtlicher Fragen. Ohne diese flankierenden Maßnahmen kann Lehrmaterial nur „prinzipiell“ ausgetauscht werden, eine realistische Nutzung ist jedoch nicht möglich.

3.4.1 Inhaltliche Einbettung

Wie bereits im vorhergehenden Kapitel erläutert, ist ein Großteil der Materialien für das Basismodul inhaltlich in einer Weise integriert und miteinander verzahnt, dass er in verschiedenen Vorlesungen bereits Verwendung findet. Die Vorlesungen vermitteln das dokumentenorientier-

te Vorgehen, für das Aufgaben und vertiefende Aufgaben verfügbar sind. Darüber hinaus gibt es noch einige Medienobjekte, die Zusatzmaterial darstellen, um neue Aufgaben konstruieren zu können. Das Angebot wird abgerundet durch Klausurfragen, die das gelernte Wissen überprüfen sollen. Damit steht einem Lehrenden zu diesem Thema eine umfassende und aufeinander abgestimmte Sammlung von Materialien zur Verfügung, die in anderen Veranstaltungen Verwendung finden können.

3.4.2 Dokumentation

Jedem Softwaretechniker ist bewusst, dass neben dem eigentlichen Programmcode eine Dokumentation eines Programmes unerlässlich ist, um den Sinn eines Programms erfassen zu können. Eine ähnliche Situation ergibt sich für Lehrmaterialien. Der Lehrende, der fremde Materialien einsetzen will, muss nicht nur das Material zur Verfügung gestellt bekommen, sondern auch die Zusammenhänge und das übergreifende Konzept verstehen, das eben nicht auf den Folien steht. Um diese Art von Dokumentation zu erstellen, wurde beim Einsatz der Musoft-Folien in der Vorlesung Techniken des Softwareentwurfs I (TSE I) in Paderborn der Ton aufgezeichnet. Durch diese Aufzeichnungen wird es nicht nur Studierenden möglich, versäumte Vorlesungen nachzubereiten, sondern insbesondere wird anderen Lehrenden die Einarbeitung in die Materialien ermöglicht. Als Ergebnis liegen 5 90-minütige Vorlesungen im Format .mp3 (jeweils ca 25 MB) vor. Für Übungsaufgaben ist eine schriftliche Dokumentation in Form von Lösungshorizonten und Musterlösungen verfügbar. Damit können Lehrende die Aufgaben verstehen und Adaptionen gemäß ihren eigenen Anforderungen durchführen.

3.4.3 Rechtliches

Ein weiterer wichtiger Punkt, den es zu klären galt, war die rechtliche Absicherung des Medieneinsatzes. Das von uns verwendete Spiel Mississippi Queen ist eine Kreation des Spieleautors Werner Hodel und wurde produziert vom Goldschläger Verlag. Da es inzwischen nicht mehr produziert wird, sind die Urheberrechte wieder an Herrn Hodel zurückgefallen. Der Autor hat zugestimmt, dass Elemente des Spiels Verwendung in unseren Lehrveranstaltungen finden. Der Werbefilm über das Karussellager ist entstanden bei der Mannesmann Dematic, die mittlerweile als Siemens Dematic firmiert. Auch hier gelang es uns, die Einwilligung der Firma für den Einsatz des Werbevideos im Rahmen unserer Lehreinheiten zu erhalten. Die Firma arvato hat an der Erstellung der Videos zu diversen Lehreinheiten aktiv mirgewirkt und die Genehmigung erteilt, sowohl die neu erstellten als auch die bereits bei der Produktionsfirma vorliegenden Szenen zu verwenden. Zur Weitergabe der Materialien wurde im Rahmen des Gesamtprojekts die MuSofT-Lizenz erstellt, unter der alle unserer Lehrmaterialien stehen.

3.4.4 Technische Integration

Unter der Technischen Integration verstehen wir die Vorbereitung auf die tatsächliche technische Weitergabe der Daten an andere Lehrende. Hierfür ist das von KT4 entwickelte MuSofT-Portal gedacht. Wir haben dafür bei der Erstellung von Materialien auf die von KT4 verbreitete Liste mit Dateiformaten geachtet: Es werden Powerpoint-Folien eingesetzt, von denen alternativ eine pdf Version verfügbar ist. Beispielmaterialien (Templates, Beispielpflichtenhefte)

wurden parallel als Microsoft Word Dokument, Rich Text Format und als Ascii Text produziert. Die Animationen von Mississippi Queen sind im Macromedia Flash Format gespeichert und das Video des Karusselllagers ist als DivX codierter AVI verfügbar.

Ein wesentlicher Schritt der technischen Integration bestand in der Annotierung der Lehrmaterialien gemäß dem LOM-Standard. Durch diese Annotation wird es möglich, komplexe Materialien, die häufig aus einer großen Menge von einzelnen Dateien bestehen, so zu strukturieren, dass geeignete Programme (so genannte Learning Management Systeme) den inhaltlichen Zusammenhang dieser Dateien erkennen und diese entsprechend in ein elektronisch verwaltetes Kursangebot eingliedern können. Darüber hinaus enthalten die Metadaten elektronisch auswertbare Informationen über den Inhalt, Schwierigkeitsgrad, eventuell benötigtes Vorwissen und Querverbindungen zwischen den einzelnen Modulen. Diese Informationen können für weitergehende Funktionalitäten, wie etwa das automatische Anordnen von Kursen genutzt werden.

Die Aufteilung der Lerneinheit Videogestützte Anforderungsdefinition findet man in Tabelle 1. Für die Lehreinheit zielorientierte Anforderungsdefinition kann ein solche Aufteilung noch nicht vorgenommen werden, da die Übungsmaterialien noch erstellt werden.

4 Erfahrungen & Evaluierung

Die Basis für die Entwicklung der Lehreinheit „Videogestützte Anforderungsdefinition“ bildete die Beobachtung, dass die Studierenden nicht die notwendigen praktischen Kompetenzen im Umgang mit Anforderungen erworben hatten, die in Nachfolgeveranstaltungen benötigt wurden. Wir haben dieser Erkenntnis Rechnung getragen, indem wir durch den Einsatz multimedialer Elemente die Komplexität der von uns behandelten Beispiele gesteigert haben. Beim Einsatz dieser Materialien konnten wir feststellen, dass den Studierenden diese erhöhte Komplexität bewusst wird. Insbesondere waren die Studierenden besser in der Lage, in den Nachfolgeveranstaltungen die ihnen gestellten Entwicklungsaufgaben anzugehen.

Insgesamt wurde die Lehreinheit Videogestützte Anforderungsdefinition bereits mehrfach eingesetzt. Drei der Einsätze fanden im Rahmen der Vorlesung „Techniken des Softwareentwurfes I“(TSE I) an der Universität Paderborn in den Wintersemestern 2001, 2002 und 2003 statt. Weiterhin wurde die LE an der Universität Braunschweig und der Universität Kassel im Informatik-Studium eingesetzt. In der IT-Akademie der Firma Siemens, die in Kooperation mit der Universität Paderborn durchgeführt wird, werden ebenfalls Materialien aus unserer LE verwendet. Insgesamt sind so bereits ca. 1500 Studierende in Kontakt mit den Materialien unserer Projekts gekommen.

Verschiedene formale und informelle Evaluationsverfahren begleiteten diese Einsätze. In den folgenden Abschnitten sollen die Verfahren, ihre Ergebnisse und die Konsequenzen für die Lehreinheit dargestellt werden.

- **Studierendenbewertung** Bei den drei Einsätzen an der Universität Paderborn wurde die Vorlesung TSE I, die ungefähr zur Hälfte aus MuSofT-Materialien besteht, im Rahmen der studentischen Vorlesungskritik evaluiert. Die Ergebnisse dieser Evaluation finden sich in Tabelle 2. Dabei geht die verwendete Skala jeweils von 1(sehr gut) bis 7(sehr schlecht). Der positive Trend bei dieser Veranstaltung ist unverkennbar. Insbesondere

Lernmodul	*Gruppenobjekt*	*Medienobjekt*
Einführung		
	Vorlesung Einführung	
		Folien Einführung
		Audio-Annotation
	Präsenzübung Softwarequalitäten	
		Übungsblatt Softwarequalitäten
		Musterlösung Softwarequalitäten
Pflichtenheft		
	Vorlesung Aufbau des Pflichtenhefts	
		Folien MdP
		Video Dematic
		Audio Vorlesung
		Beispiel-Plichtenheft
	Heimübung Pflichtenheft	
		Übungsblatt Pflichtenheft
		Pflichtenheft Template
		Animationen Mississippi Queen
		Musterlösung Pflichtenheft
Modell des Problembereichs		
	Vorlesung MdP	
		Folien MdP
		Video Dematic
		Beispiel-Audio
		Beispiel Pflichtenheft
	Präsenzübung MdP/Use Cases	
		Übungsblatt MdP/Uses Cases
		Musterlösung
Geschäftsprozesse		
	Vorlesung Geschäftsprozesse	
		Folien Geschäftsprozesse
		Beispiel-Audio
	Heimübung Geschäftsprozesse	
		Übungsblatt Geschäftsprozesse
		Spielanleitung Mississippi Queen
		Animationen Mississippi Queen
		Musterlösung Übungsblatt

Tabelle 1: Aufteilung der Lerneinheit Videogestützte Anforderungsdefinition gemäß LOM

Frage	*2001*	*2002*	*2003*
Wie beurteilen Sie das Vorlesungsskript	4,4	4,0	2,5
Wie beurteilen Sie die Vorlesung gesamt	4,9	3,6	2,1

Tabelle 2: Die Vorlesung Techniken des Softwarentwurfs in der studentischen Kritik

die Verbesserung der Note für das Vorlesungsskript kann als Indiz dafür gewertet werden, dass unser beständiges Bemühen um eine Qualitätsverbesserung der Materialien Früchte trägt.

- **Gezielte Evaluation** Die studentische Vorlesungskritik gibt zwar einen allgemeinen Überblick über die Qualität der Vorlesung und in Freitextkommentaren können hier auch detaillierte Einzelmeinungen zum Ausdruck kommen, für eine gezielte Verbesserung der MuSofT-Materialien ist diese Evaluation jedoch zu allgemein. Wir haben daher gezielte Evaluationen bei den Studierenden durchgeführt, indem wir zeitnah zum Einsatz der MuSofT-Materialien einen Fragebogen verteilt haben. Eine detaillierte Auswertung würde den Rahmen hier sprengen, für das Jahr 2001 ist ein umfangreiches Dokument zu den Ergebnissen dieser Evaluation verfügbar [Hau02]. Wesentliche Ergebnisse der ersten Evaluation (also dem Einsatz in Paderborn 2001) waren, dass die Studierenden sich sehr unsicher sind, welchen Detailgrad die von ihnen zu erstellenden Lösungen haben sollten (vgl. Abb. 5). Als Konsequenz daraus haben wir die Dokumentenzentrierung als didaktisches Konzept in unserer Lehreinheit verankert, die auch von 80% der Studierenden im Semester 2002 als hilfreich eingestuft wurde. Andererseits zeigt die Evaluation des zweiten Einsatzes der (dann deutlich überarbeiteten) Materialien, dass die Unterschiede von den Studierenden nicht in dem Masse wahrgenommen werden, wie wir uns dies erhofft hatten. Die Befragung zum Einsatz in 2003 läuft zur Zeit noch, da der Einsatz der Materialien bis in den Dezember stattfand, eine vollständige Auswertung kann daher noch nicht erfolgen. Überblicksartig lässt sich jedoch bereits feststellen, dass auch in dieser gezielten Befragung sich das positive Bild der allgemeinen Evaluation widerspiegelt.

- **Beobachtung der Nachfolgeveranstaltung** Eine weitere wesentliche Grundlage für die (Weiter-) Entwicklung unserer Lehreinheit stellte der enge Kontakt zu den Veranstaltern der Nachfolgeveranstaltung dar. Im Studienplan für Informatik an der Universität Paderborn ist vorgesehen, dass im Anschluss an die Grundlagenveranstaltung Techniken des Softwareentwurfs ein Softwarepraktikum absolviert wird. In diesem müssen sich die Studierenden in kleinen Gruppen einer recht komplexen Softwareentwicklungsaufgabe stellen. Dabei sollen sie die zuvor theoretisch erworbenen Konzepte und Methoden praktisch umsetzen. Dies bietet uns die Gelegenheit, den realen Lernerfolg unserer Vorlesung zu beobachten. Aus diesen Beobachtungen entstand auch die Motivation für die grundlegende Konzeption der Lehreinheit. Während des Einsatzes der Musoft-Materielien haben wir nunmehr zweimal das Softwarepraktikum gerade in der Anfangs-, also der Anforderungsdefinitionsphase, beobachtet. Im Semester 2002 (aufbauend auf TSE im Wintersemester 2001/2002) haben wir eine Reihe von Gruppensit-

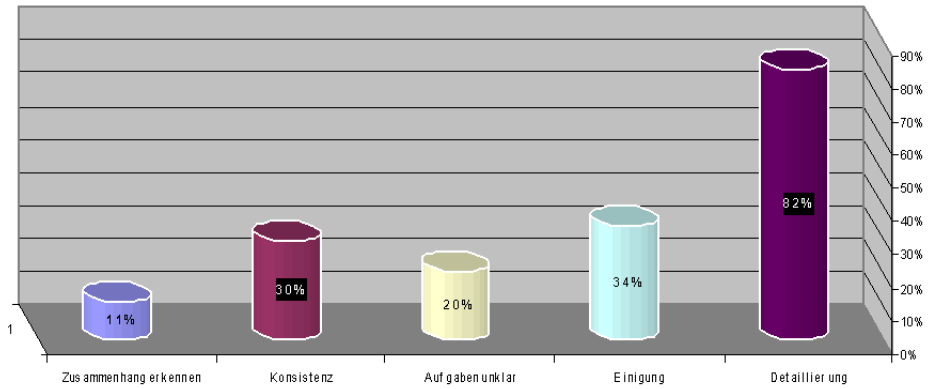


Abbildung 5: Antwortverteilung auf die Frage nach erkannten Problemen bei der Bearbeitung der Aufgaben, Mehrfachnennung war möglich (aus [Hau02])

zungen auf Video aufgezeichnet (siehe Abb.6) und versucht, durch Auswertung dieser Videos Hinweise auf Defizite im Vorwissen der Studierenden zu erhalten. Es stellte sich heraus, dass insbesondere die Dokumentenstruktur des im Softwarepraktikum geforderten Pflichtenheftes den Studierenden fremd war. Wir mussten jedoch auch bemerken, dass die Auswertung der Videoaufzeichnungen äußerst langwierig und aufwändig war und wenig konkrete Informationen erbrachte. Im Sommersemester 2003 konnten wir diese Informationsbeschaffung dadurch vereinfachen, dass einer der TSE-Tutoren auch eine Softwarepraktikumsgruppe leitete und daher gezieltes Feedback über festgestellte Defizite geben konnte. Wir konnten feststellen, dass die Studierenden den Umgang mit der UML als selbstverständliches Notationselement sicher erlernt hatten. Sie waren besser in der Lage, den Zusammenhang zwischen der komplexen Aufgabenstellung und den abstrakten Repräsentationen der Modelle zu verstehen. Wir können daher von einem nachweislichen Lehr- bzw. Lernerfolg unserer Veranstaltung sprechen. Da diese Aussage auf Beobachtungen einzelner Gruppen beruht, ist sie jedoch nicht allgemein quantifizierbar.

- **Peer-Review** Das Hauptziel des Rahmenprogramms NMB ist die Produktion weitergabefähiger Materialien. Um zu gewährleisten, dass die Materialien des Teilprojekts 1.1 dem Kriterium der Weitergabefähigkeit entsprechen, haben wir einen Peer-Review der Materialien durch 3 Lehrende durchgeführt. Dabei wurden die von uns erstellten Lehrmaterialien in Zusammenarbeit mit projekt-externen Lehrenden (Reiko Heckel von der Universität Paderborn und Albert Zündorf von der Uni Kassel) überarbeitet. Diese Kooperation ermöglichte es uns, Spezifika unsere Materialien aufzuspüren, die anderen Lehrenden nicht nachvollziehbar waren. Die Materialien wurden so angepasst, dass sie letztendlich von vier verschiedenen Lehrenden in mehreren verschiedenen Veranstaltungen eingesetzt werden konnten. Auch wenn diese Art der Evaluation kein direkt quantifizierbares Ergebnis erreicht und auch die resultierenden Veränderungen an den

Abbildung 6: Beobachtung einer Gruppe des Softwarepraktikums

Lehrmaterialien eher in Details sichtbar sind, so ist doch insbesondere dieses „Abschleifen“ von kleinen Ungenauigkeiten, missverständlich gebrauchten Begriffen etc., das die Einarbeitungszeit anderer Lehrender in diese Materialien signifikant verkürzt und Fehlerquellen ausschaltet.

5 Zusammenfassung

Im Rahmen des Teilprojekts 1.1 Videogestützte Anforderungsdefinition wurden Materialien geschaffen, die mit Hilfe multimedial aufbereiteter Beispiele die Phase der Anforderungsdefinition im Softwareentwicklungsprozess verdeutlichen. Durch verschiedene Evaluationsmethoden wurde dabei gleichsam den Interessen der Studierenden und der Lehrenden Rechnung getragen. So gelang es uns, eine Lehreinheit zu schaffen, die nachweislich in der Lage ist, Studierenden die Probleme und Techniken der Anforderungsdefinition klar zu machen und sie in die Lage zu versetzen, sich Aufgaben aus diesem Bereich zu stellen. Gleichzeitig haben wir durch eine inhaltliche, rechtliche und technische Aufbereitung im Hinblick auf die Weitergabe der Materialien dafür gesorgt, dass die von uns erreichten Projektergebnisse im Kontext des Gesamtprojekts MuSoft und darüber hinaus im Kontext des Rahmenprogrammes NMB weitergabefähig sind.

Literatur

[ADE03] ALFERT, KLAUS, ERNST-ERICH DOBERKAT und GREGOR ENGELS (Herausgeber): *Ergebnisbericht des Jahres 2002 des Projektes “MuSofT – Multimedia in*

der SoftwareTechnik", Band 133 der Reihe *Softwaretechnik-Memo*. Lehrstuhl für Software-Technologie, Fachbereich Informatik, Universität Dortmund, März 2003.

[DE02] DOBERKAT, ERNST-ERICH und GREGOR ENGELS (Herausgeber): *Ergebnisbericht des Jahres 2001 des Projektes "MuSofT – Multimedia in der Software-Technik"*, Band 121 der Reihe *Softwaretechnik-Memo*. Lehrstuhl für Software-Technologie, Fachbereich Informatik, Universität Dortmund, März 2002.

[FD96] FINKELSTEIN, A. und J. DOWELL: *A Comedy of Errors: the London Ambulance Service case study*. In: *8th International Workshop on Software Specification & Design IWSSD-8*, Seiten 2–4. IEEE CS Press, 1996.

[Hau02] HAUSMANN, JAN HENDRIK: *Ergebnisse der Evaluierung des Einsatzes von MuSofT-Materialien in der Vorlesung Techniken des Softwarentwurfs 1 im Wintersemester 2001/2002 an der Universität Paderborn*. Verfügbar über den MuSofT-Server, 2002.

Abschlussbericht der Lerneinheit 1.2 „Entwicklung von Informationssystemen“

Dirk Jesko

Die Lerneinheit 1.2 beschäftigte sich im Rahmen des Projekts „MuSofT – Multimedia in der Softwaretechnik“ mit der Erstellung von Materialien für Lehrveranstaltungen des Fachgebiets Datenbanken. Dieser Bericht fasst die Ergebnisse des Projektes zusammen. Zunächst wird auf einige didaktischer Aspekte, die Abgrenzung der betrachteten Inhalte und deren Strukturierung eingegangen. Daran schließen sich Aussagen zur technischen Realisierung an. Schließlich werden unsere während der Projektlaufzeit gesammelten Erfahrungen kurz diskutiert.

Inhaltsverzeichnis

1 Einleitung

Die Entwicklung von Datenbankanwendungen und Informationssystemen ist eine Aufgabe, die sowohl theoretische als auch praktische Kenntnisse erfordert. Die derzeitigen Lehrveranstaltungen vermitteln diese vorwiegend in Form von Vorlesungen mit Hilfe passiver Materialien. Die praktische Arbeit in Übungen, etwa die Modellierung von Anforderungen und die Arbeit mit Anfragesprachen wie SQL, erfolgt nur in unzureichendem Umfang. Daher wurden in der Lerneinheit 1.2 „Entwicklung von Informationssystemen“ des Projektes MuSofT insbesondere Materialien für Übungen erstellt. Die vorhandenen Vorlesungsmaterialien wurden weiterhin um multimediale Elemente erweitert, z.B. Videos, die den Datenbankeinsatz motivieren und die Anforderungserfassung am Beispiel verdeutlichen. Schließlich wurden die bestehenden und neuen Materialien so aufbereitet, dass sie, durch Bereitstellung im Internet, auch für ein ergänzendes Selbststudium verwendbar sind. Insbesondere wurde ein Ansatz für die Aufzeichnung und strukturierte Bereitstellung im Internet entwickelt.

Durch die multimediale Unterstützung von Vorlesungen werden Lehrinhalte orts- und zeitunabhängig verfügbar. Derzeit existieren mehrere Projekte, die sich damit beschäftigen, einen zentralen Zugang zu Vorlesungsmaterialien aus verschiedensten Fachbereichen zu schaffen. Beispiele hierfür sind die Open-Courseware-Initiative des MIT [Mas03], das Projekt 100-online der Universität Stuttgart [Stu03] oder die World Lecture Hall der Universität Texas [The03]. Die jeweiligen Portale bieten einen zentralen Zugriff auf die Materialien der Veranstaltungen. Diese reichen von Skripten (in unterschiedlichen Formaten), Foliensammlungen (die teilweise an die Präsentation im WWW angepasst wurden) bis hin zu Animationen und Werkzeugen (z.B. als Java-Applets) für Experimente.

Der vorliegende Bericht fasst die Ergebnisse des Teilprojekts 1.2 zusammen. In Abschnitt 2.1 wird zunächst ein Überblick des Datenbankentwurfs und der in der Lerneinheit betrachteten Themen gegeben. Daran schließen sich im Abschnitt 2.2 Aussagen zur Zielgruppe, den Lernzielen, etc. der Lerneinheit an. Um eine Wiederverwendung der Materialien zu gewährleisten, werden diese in Form kleiner Blöcke (Lernobjekte) bereitgestellt, die von übergreifenden Fallstudien begleitet werden. Abschnitt 2.3 gibt einen Überblick dieser Strukturierung und der Fallstudien. Abschnitt 3 beschreibt die technischen Details von zwei Ergebnissen des Projekts. Zum Einen einen Ansatz zur Bereitstellung von Videoaufnahmen von Vorlesungen, zum Anderen die Anpassung des SQuirreL SQL Client, einem Werkzeug für die Arbeit mit SQL. In Abschnitt 4 werden verschiedene Erfahrungen diskutiert, die wir durch das Projekt mit der Multimediaunterstützung von Lehrveranstaltungen gemacht haben. Schließlich gibt Abschnitt 5 eine Zusammenfassung der Ergebnisse und einen Ausblick möglicher weiterführender Arbeiten.

2 Vorstellung der Lerneinheit

2.1 Inhalte

Die Komplexität des Themenbereichs „Datenbanken und Informationssysteme“ verhindert eine umfassende Betrachtung aller Aspekte aus diesem Bereich im Rahmen des Projekts. Daher erfolgt eine Spezialisierung auf die grundlegenden Themen des Datenbankentwurfs. Thema-

tisch orientieren sich die erstellten Materialien an den am Institut angebotenen Grundlagenvorlesungen „Datenbanken I“ und „Datenmanagement“. Erstere betrachtet dabei mehr theoretische Aspekte, etwa die Grundlagen von Datenbankanfragesprachen in Form von Algebren und Kalkülen. Letztere ist vorwiegend auf das Grundstudium ausgerichtet und betrachtet verstärkt praktische Aspekte, etwa die Formulierung von Anfragen in SQL und die Anwendungsprogrammierung. Im folgenden geben wir einen kurzen inhaltlichen Überblick der Inhalte.

In der Vorlesung werden zunächst grundlegende Konzepte der Datenbankterminologie und -technik, der historischen Entwicklung von Datenbank-Management-Systemen (DBMS), Gründe für deren Einsatz, sowie Basis-Funktionen und Architekturen von DBMS dargestellt. Ziel ist die Motivation des Datenbankeinsatzes. Weiterhin werden Architekturen und Komponenten von Datenbanksystemen eingeführt. Die folgenden Themen bauen auf dieser Motivation auf und orientieren sich an den Phasen des Datenbankentwurfsprozesses.

In der Phase des konzeptionellen Entwurfs wird, basierend auf den während der Anforderungsanalyse erfassten Daten, eine erste formale, von der Implementierung unabhängige Beschreibung des Fachproblems erstellt. Dabei kommen verschiedene Modelle zum Einsatz, z.B. Entity-Relationship-Modelle [Che76] für die Datenmodellierung. Deren Notation und Semantik werden innerhalb der Lerneinheit eingeführt. Zusätzlich wird auf Erweiterungen und objektorientierte Methoden eingegangen.

Für die Implementierung von Datenbanksystemen (DBS) werden gewöhnlich andere, weniger abstrakte Modelle verwendet, z.B. das Relationenmodell ([Cod70]). Dieses hat als Grundlage relationaler Datenbanken die weiteste Verbreitung erfahren. Die Lerneinheit führt die grundlegenden Konzepte des Relationenmodells und deren Semantik formal und anhand von Beispielen ein. Ergänzend werden auch andere Modelle, z.B. das hierarchische Modell, kurz eingeführt. Ziel ist vorrangig die Vermittlung von Grundlagenwissen über das Relationenmodell, auf das verschiedene der folgenden Themen aufbauen, z.B. die Anfragesprachen.

Wie der Entwurf allgemeiner Softwaresysteme, erfordert auch der Entwurf von Datenbanken eine strukturierte Vorgehensweise. In der Lerneinheit wird ein Phasenmodell für den Datenbankentwurf eingeführt, das sich an denen der allgemeinen Softwareentwicklung orientiert, aber an die besonderen Anforderungen angepasst ist. Neben einer Übersicht der Phasen werden der konzeptionelle und der logische Entwurf genauer betrachtet. Diese Phasen erfordern insbesondere praktische Fähigkeiten, wie die Modellierung der Datenstrukturen mittels geeigneter Methoden, z.B. des Entity-Relationship-Modells (ERM) und deren Abbildung auf das Relationenmodell. Diese sollen durch die Bereitstellung von Übungsaufgaben und entsprechenden Werkzeugen vermittelt werden.

Ein zweiter, modellabhängiger Schritt des logischen Entwurfs befasst sich mit der Verfeinerung der erstellten Schemata. Im Rahmen der Lerneinheit wird speziell auf die Verfeinerung von Relationenschemata eingegangen. Ziel ist die Vermeidung von Redundanzen, ohne semantische Informationen zu verlieren. Zunächst werden wesentliche Begriffe wie *Funktionale Abhängigkeit*, *Schlüssel*, etc. eingeführt. Anschließend wird auf Normalformen und Verfahren zur Normalisierung und deren Eigenschaften eingegangen [EN00]. Diese Aspekte werden einerseits formal betrachtet. Andererseits erfolgt die Vermittlung praktischer Kenntnisse, etwa zur Analyse von funktionalen Abhängigkeiten und zur Normalisierung von Relationen, in Form von Übungsaufgaben.

Nachdem der logische Entwurf einer Datenbank abgeschlossen ist, kann das erstellte Schema auf einem konkreten DBS umgesetzt werden. Dies geschieht in der Phase der Datende-

finition mittels einer Datendefinitionssprache (DDL). Die Lerneinheit führt DDLs zunächst allgemein ein, wobei u.a. Anforderungen an relationale DDLs nach [Cod90] angegeben werden. Anschließend erfolgt eine umfassendere Darstellung der Konzepte der SQL-DDL sowie eine einführende Betrachtung anderer DDLs.

Nachdem das Datenbankschema erstellt und auf einem konkreten Datenbanksystem umgesetzt wurde, können Daten eingefügt und verändert werden. Weiterhin ist die Definition von *Anfragen* oder *Sichten* möglich, um basierend auf den in der Datenbank gespeicherten Relationen, neue zu definieren. Zunächst werden der Einsatz spezieller Anfragesprachen motiviert und deren Vor- und Nachteile gegenüber „konventionellen“ Programmiersprachen aufgezeigt. Weiterhin erfolgt die Einführung der Grundlagen von Anfrage- und Änderungsoperationen mittels algebraischer und kalkülartiger Notationen. Ziel des Lernmoduls ist die Vermittlung der notwendigen Kenntnisse, um informal beschriebene Anfragen oder Änderungen an eine Datenbank formal spezifizieren zu können.

In relationalen Datenbanken stellt die Structured Query Language (SQL) [DD99] die bedeutendste Anfragesprache dar. In der Lerneinheit werden die Konstrukte von SQL eingeführt. Dies umfasst den SQL-Kern (**`select`** ... **`from`** ... **`where`** ..., etc.) sowie erweiterte Sprachkonstrukte (Aggregatfunktionen, Gruppierung, Mengenoperationen, etc.). Weiterhin werden verschiedene SQL-Versionen vorgestellt und verglichen. Die praktische Arbeit mit SQL wird durch die Bereitstellung von Übungsaufgaben und Werkzeugen zum „Experimentieren“ mit der Sprache unterstützt. Ergänzend zu SQL wird in der Lerneinheit auch auf weitere Anfragesprachen, etwa *Query by Example* (QBE) und Erweiterungen relationaler Anfragesprachen eingegangen. Diese werden einführend vorgestellt und miteinander sowie mit der Relationenalgebra und den Kalkülen hinsichtlich ihrer Mächtigkeit verglichen.

Bei der Präsentation der Datenbestände ist es, z.B. aus Datenschutzgründen, nicht immer wünschenswert alle Daten auszugeben. Vielmehr werden, abhängig von der jeweiligen Anwendung, nur bestimmte Ausschnitte der Datenbestände verwendet. Die Architektur von Datenbanken trägt diesem Sachverhalt durch die logische Datenunabhängigkeit Rechnung. Die Lerneinheit führt das Konzept der *Sicht* als Mittel ein, diese zu erreichen. Anhand von SQL werden verschiedene Arten von Sichten eingeführt und es wird auf die Theorie änderbarer Sichten eingegangen. Weiterhin werden sich ergebende Probleme dargestellt.

Datenbankanfragesprachen wie SQL besitzen eine eingeschränkte algorithmische Mächtigkeit. Dies ist erforderlich, um bestimmte Eigenschaften wie Optimierbarkeit, Terminierung, etc. zu gewährleisten. Für konkrete Anwendungen ist diese Mächtigkeit aber oft nicht hinreichend. Daher wurden Ansätze entwickelt, um Anfragesprachen mit Programmiersprachen zu koppeln. Die Lerneinheit gibt einen allgemeinen Überblick derartiger Ansätze, u.a. PL/SQL sowie JDBC und SQLJ zur Kopplung von SQL und Java. Von zunehmender Bedeutung ist auch der Zugriff auf Datenbanken über das Internet. Daher werden entsprechende Techniken vorgestellt.

Eine weitere Anforderung, die ein DBMS erfüllen muss, ist die Sicherung der Daten und die Wahrung der Konsistenz. Dies ist insbesondere im Mehrbenutzerbetrieb von Bedeutung. Daher beschreibt die Lerneinheit die Definition von Integritätsbedingungen und Triggern, sowie die Vergabe von Rechten mittels SQL.

2.2 Didaktisches Konzept

2.2.1 Leitbild

Die in der Lerneinheit „Entwicklung von Informationssystemen“ erstellten Materialien basieren auf den am Institut angebotenen Vorlesungen „Datenbanken I“ und „Datenmanagement“ und sollen auch weiterhin in diesem Bereich eingesetzt werden. Die Vorlesungen richten sich an Studierende sehr unterschiedlicher Studiengänge (verschiedene Informatikstudiengänge aber auch Studiengänge anderer Fakultäten). Daher müssen die in der Lerneinheit erstellten Materialien entsprechend flexibel sein, d.h. sie müssen derart strukturiert werden, dass eine Zusammenstellung von Materialien für eine bestimmte Zielgruppe möglich ist. Zunächst soll das Umfeld in dem die Lerneinheit zum Einsatz kommen soll, beschrieben werden:

Studiengang: Die Lerneinheit soll vorrangig in Studiengängen der Fakultät für Informatik zum Einsatz kommen, d.h. sowohl in der Kern-Informatik, als auch in anwendungsorientierten Studiengängen wie Computer Visualistik, Ingenieurinformatik und Wirtschaftsinformatik. Weiterhin können Teile der Lerneinheit für Studiengänge mit Nebenfach Informatik, z.B. Maschinenbau, Mathematik, Wirtschaftsingenieurwesen Logistik etc. genutzt werden.

Studienfach: Die Lerneinheit gliedert sich in Studienfächer der angewandten und praktischen Informatik, speziell die Entwicklung von Datenbanken und Informationssystemen ein.

Berufliche Qualifikation/Einsatzfelder: Die erworbenen Kenntnisse können bei der Entwicklung und Anwendung von datenbankbasierten Softwaresystemen angewendet werden. Außerdem dienen sie als Grundlage für weitergehende Veranstaltungen, die sich z.B. mit den Interna von DBMS befassen. Diese wiederum sind für den späteren Einsatz im Bereich Datenbankadministration von Vorteil.

Vorkenntnisse: Die Lerneinheit befasst sich u.a. mit der Formalisierung von Anfragesprachen und Modellierungskonzepten des ER-Modells. Für diese Themen sind gewisse mathematische Vorkenntnisse im Bereich Algebra, Mengenlehre und Logik erforderlich. Für die Anwendungsentwicklung sind Vorkenntnisse zu den verwendeten Programmiersprachen wünschenswert.

Ausbildungssituation: Die Lerneinheit ist vorrangig für die Ausbildung an einer Universität oder Fachhochschule vorgesehen. Je nach Studiengang und geplanter Spezialisierung ist sie im Grund- oder Hauptstudium angesiedelt. In Magdeburg werden die zugrunde liegenden Vorlesungen als Pflicht- bzw. Wahlpflichtfach im 3. oder 4. Semester des Grundstudium oder im 5. Semester des Hauptstudiums der Informatikstudiengänge angeboten.

2.2.2 Lernziele

Die Lerneinheit vermittelt grundlegende Kenntnisse für den Entwurf und die Implementierung von Datenbanken und Datenbankanwendungen. Insbesondere sollen die Formalisierung

von Anforderungen mittels entsprechender Entwurfsmodelle und deren Abbildung auf Modelle konkreter Datenbanksysteme gelehrt werden. Ein weiterer wichtiger Aspekt, der vermittelt werden soll, ist die Formalisierung von Anfragen, die meist in Form natürlicher Sprache gegeben sind, mittels spezieller Datenbankanfragesprachen (insbesondere SQL). Der Studierende soll dabei u.a. erlernen, welche Anfragen sich überhaupt mit einer derartigen Sprache ausdrücken lassen.

2.2.3 Lernszenario

Die entwickelten Materialien sollen vorrangig in der Präsenzlehre (Vorlesungen, Übungen, etc.) eingesetzt werden. Aber auch die Unterstützung des ergänzenden Selbststudiums ist geplant. Im wesentlichen ergeben sich zwei Lernszenarios:

Präsenzlehre (Vorlesung/Übung): In Vorlesungen werden weiterhin vorrangig herkömmliche Präsentationstechniken wie Folien eingesetzt. Diese werden an geeigneter Stelle durch Animationen und Videos erweitert, um beispielsweise Verfahren und Algorithmen zu verdeutlichen. Für Übungen werden vermehrt interaktive Materialien eingesetzt, um die praktische Arbeit mit den in der Lerneinheit vorgestellten Verfahren, Sprachen etc. zu vertiefen.

Selbststudium/Weiterbildung: Für dieses Szenario werden zum einen die Materialien aus den Vorlesungen in strukturierter Form über das Internet bereitgestellt. Ergänzend werden beispielsweise Videoaufnahmen der Vorlesung angeboten, um so dem Studierenden die Möglichkeit zu geben, die entsprechenden Erläuterungen zu wiederholen. Weiterhin erfolgt die Bereitstellung zusätzlicher Materialien und Hinweise, die einer Vertiefung des Stoffes dienen und Hinweise geben sollen, in welcher Richtung diese erfolgen können. Die Grundlage der Präsentation und Strukturierung der Materialien bilden in diesem Fall HTML-Seiten. Die für die Erstellung von Visualisierungen erstellten Werkzeuge werden ebenfalls für Experimente zur Verfügung gestellt. Damit soll es dem Studierenden ermöglicht werden, die vermittelten Konzepte, etwa Verfahren des logischen Datenbankentwurfs, selbst zu vertiefen.

2.3 Strukturkonzept

Die Inhalte der Lerneinheit wurde bereits im Abschnitt 2.1 vorgestellt, wobei bereits eine gewisse Gliederung in Themenkomplexe erfolgte. Diese sind aber für die Definition von Abhängigkeiten zwischen einzelnen Lernobjekten noch zu grob ist. In diesem Abschnitt sollen daher zunächst die einzelnen konkret geplanten Lernmodule aufgelistet und in gewissem Umfang weiter in Gruppenobjekte untergliedert werden.

In Abbildung 1 ist die Struktur der Lernmodule der Lerneinheit dargestellt. Weiterhin wurden diese in gewissem Umfang weiter in Gruppenobjekte untergliedert. Auf eine vollständige Auflistung der Gruppenobjekte sowie verschiedener Versionen von Lernmodulen und Gruppenobjekten (z.B. angepasst an bestimmte Zielgruppen) wurde aus Platzgründen verzichtet. Nicht alle in der Abbildung angegebenen Themen werden im gleichen Umfang behandelt. Die Kernthemen, für die die überwiegende Zahl der Materialien, Werkzeuge und Übungsaufgaben erstellt wird, sind entsprechend gekennzeichnet. Auf die Festlegung von Beziehungen

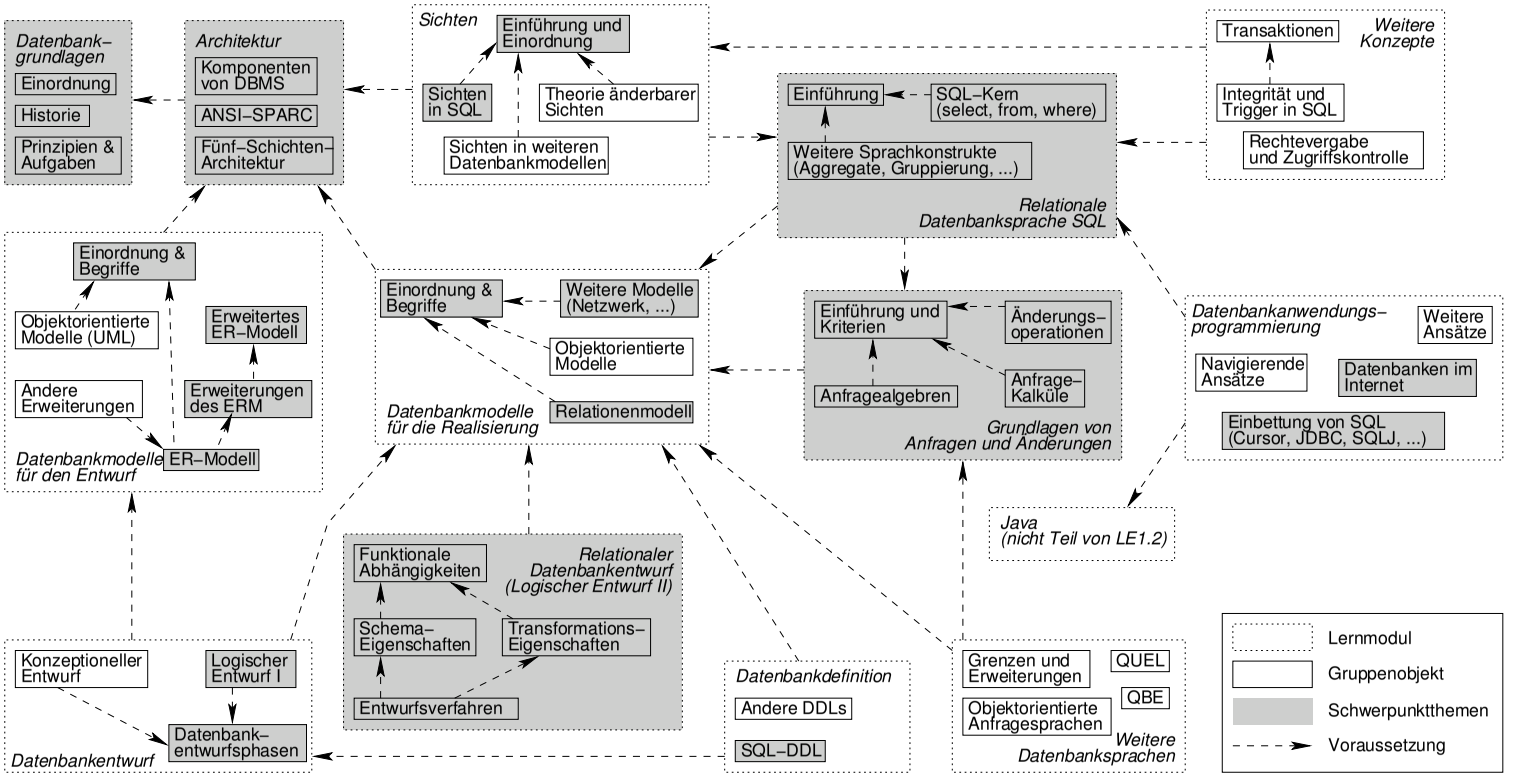


Abbildung 1: Struktur und Abhängigkeiten der Lernmodule und Gruppenobjekte

der Gruppenobjekte untereinander wurde ebenfalls verzichtet, da dies erst sinnvoll möglich ist, wenn deren Inhalte weitestgehend feststehen.

3 Technische Realisierung

Im Rahmen des Teilprojektes 1.2 wurden im wesentlichen drei Aufgaben bearbeitet:

- Anpassung und Erweiterung des SQuirreL SQL Client für Arbeit mit SQL und die Visualisierung der Tabellen und deren Beziehungen in Datenbanken
- Aufzeichnung von Vorlesungen und Bereitstellung im Internet.
- Konzipierung von Drehbüchern für die Erstellung von Videos realer Vorgänge aus dem Bereich Logistik und der damit verbundenen Datenverarbeitung. Die Videos wurden dann bei der arvato AG (`http://www.arvato.de/`) erstellt.

Die folgenden beiden Abschnitte gehen auf einige Aspekte der Realisierung der ersten beiden Punkte ein.

3.1 SQL-Tool

Um Datenbankanwendungen entwickeln zu können, ist insbesondere die Kenntnis von Anfragesprachen von Bedeutung. Diese lassen sich mit passiven Materialien oder Übungen auf dem Papier jedoch nur bedingt vermitteln. Daher bestand ein Ziel in der Erstellung oder Anpassung eines Programms, das die direkte Arbeit mit Datenbanken und insbesondere der Ausführung von Anfragen gestattet. Für die Auswahl waren verschiedene Aspekte von Bedeutung, u.a. Plattform- und Datenbankunabhängigkeit, Erweiterbarkeit und Laden und Speichern von Skripten. Weiterhin muss die Oberfläche an unterschiedliche Einsatzszenarien anpassbar sein, z.B. Präsentation und Übungen am Rechner. Schließlich soll das Werkzeug Visualisierungen ermöglichen. Insbesondere sollten Tabellen dargestellt werden, sowie deren durch Schlüssel und Fremdschlüssel definierten Beziehungen. Auch eine Hervorhebung von Attributen mit bestimmten Eigenschaften, z.B. Schlüssel, sollte möglich sein. Insbesondere bei der Formulierung von Anfragen mit Joins wird die Bestimmung der Attribute für die Verbundbedingung dadurch erleichtert.

Die Anforderung der Plattform- und Datenbankunabhängigkeit lässt sich durch den Einsatz eines auf Java und JDBC basierenden Tools realisieren. Derzeit existiert verschiedene derartiger Werkzeuge, z.B. DBVisualizer [Min03], SQuirreL SQL Client [Squ03] und Independent SQL Tool [isq03]. Unter diesen ist, nach unserer Kenntnis, DBVisualizer das einzige, welches eine graphische Darstellung bietet. Andererseits ist der Quellcode nicht frei verfügbar und es gibt keine Möglichkeit für Erweiterungen. Zwischenzeitlich ist das Tool auch nicht mehr frei verfügbar. Die besten Voraussetzungen bot der SQuirreL SQL Client. Eine graphische Darstellung ermöglichte das Tool zunächst nicht, jedoch besteht über eine Plugin-Architektur die Möglichkeit für Erweiterungen.

Eine Reihe nützlicher Funktionalitäten stand bereits in Form von Plugins zur Verfügung, weitere kamen während der Projektlaufzeit von anderer Stelle hinzu. Folgende Funktionalitäten waren für das Teilprojekt insbesondere von Bedeutung:

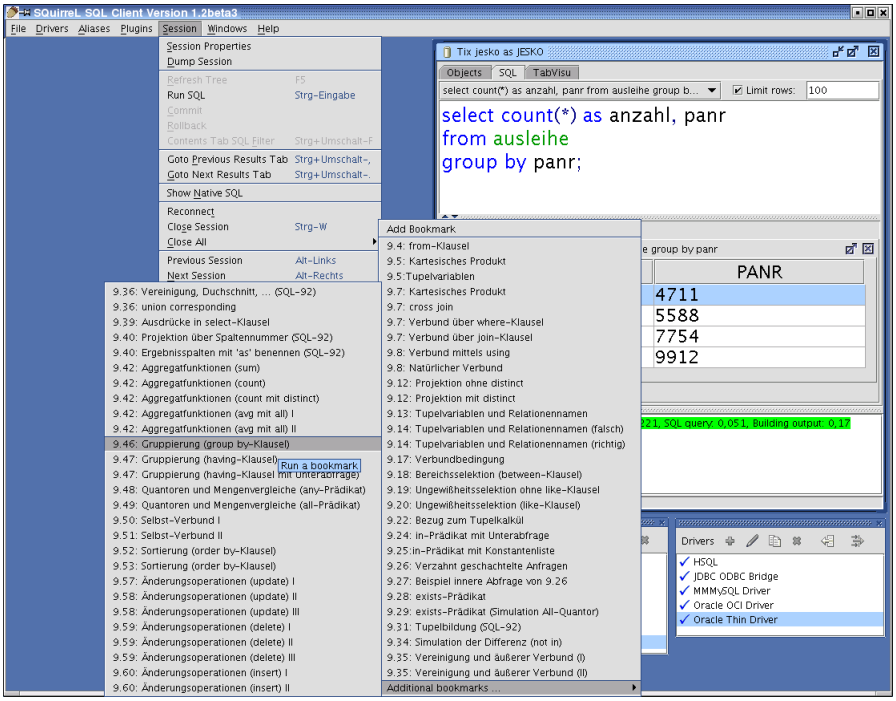


Abbildung 2: Anfragen aus der Vorlesung als Bookmarks im SQuirreL SQL Client

SQLScript Plugin: Es gestattet das Laden und Speichern von einzelnen Anfragen und kompletten Skripten. Diese können dann beispielsweise in der Vorlesung oder in Übungen geladen, ausgeführt, verändert etc. werden. Diese bildet die Grundlage für die Bereitstellung von Aufgabenstellungen und die Übermittlung von Lösungen zu Übungsaufgaben.

SQLBookmark Plugin: Es bietet ebenfalls die Möglichkeit, Anfragen vorzudefinieren und im Menü bereitzustellen. Wir haben derzeit alle Anfragen, die auf den Folien der Vorlesung verwendet werden, in Form von Bookmarks bereitgestellt, so dass diese in der Vorlesung leicht zugänglich sind (vgl. Abbildung 2).

Look and Feel Plugin: Es ermöglicht die Anpassung der Oberfläche (Fonts, Farben, etc.) und somit den Einsatz in unterschiedlichen Umgebungen, z.B. zur Präsentation in Vorlesungen (erfordern größere Fonts) oder für Übungen am Rechner.

SQL Validator Plugin: Es ermöglicht die syntaktische Validierung einer SQL-Anfrage, wobei ein Web Service genutzt wird [mim03]. Dies ist insbesondere für Übungen und das Selbststudium von Interesse.

JEdit Plugin: Es erlaubt das Syntax-Highlighting von SQL-Anfragen. So werden beispielsweise Schlüsselworte, Tabellen- und Attributnamen hervorgehoben, was hilfreich bei der Erstellung von Anfragen ist und auch die Präsentation in Vorlesungen unterstützt.

Für die Visualisierung war eine Bibliothek für die Darstellung von Graphen erforderlich. Wir haben daher verschiedene derartige Frameworks für Java analysiert, u.a. GenGEd [Bar00] und DiaGen [MK99], die auch Funktionen für die Erstellung von Editoren bereitstellen. GenGEd basiert auf Graph Transformation und bietet eine GUI, um Symbole, Transformationsregeln, Editor etc. zu spezifizieren. DiaGen basiert auf Hypergraphen und Hypergraph-Grammatiken, die in Textform definiert werden. Abschließend wird daraus automatisch Java-Code generiert, der manuell erweitert werden kann. Die Definition der Regeln erwies sich als schwierig, insbesondere im Falle von DiaGen (rein textuelle Definition). Weitere Probleme waren die Plattformabhängigkeit (GenGED) und Fehler beim automatischen Layout (DiaGen). Weiterhin wurden verschiedene Frameworks untersucht, die ausschließlich für die Visualisierung von graph-basierten Strukturen verwendet werden. Diese werden jedoch meist kommerziell vertrieben und kamen daher nicht zum Einsatz.

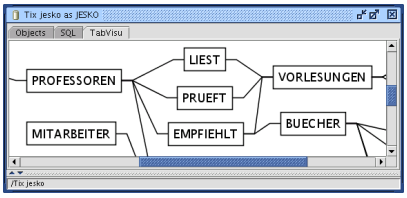
Schließlich wurden Prototypen des Plugins basierend auf Jazz und dessen Nachfolger Piccolo [BMG00] erstellt (siehe Abbildung 3). Diese Frameworks dienen insbesondere für die Erstellung von „Zoomable Userinterfaces“ (ZUI). Piccolo ist in Java implementiert und steht im Sourcecode zur Verfügung. Es bietet Funktionalitäten für Visualisierungen im 2D-Bereich. Funktionen für das Editieren (Markieren, Verschieben, Kopieren, Löschen etc.) stehen ebenfalls zur Verfügung. Schließlich sind die notwendigen graphischen Konstrukte für die Visualisierung der Datenstrukturen bereits enthalten.

Die realisierte Visualisierungskomponente gestattet u.a. auch ein gewisses semantisches Zooming, d.h. je nach Zoomfaktor werden mehr oder weniger Informationen dargestellt. Insbesondere bei Präsentationen ist es damit möglich, nicht erforderliche Informationen auszublenden. In Abbildung 3 ist dies beispielhaft dargestellt. Ein Nachteil von Piccolo ist das Fehlen von Algorithmen für ein automatisches Layout. Derartige Funktionalitäten bot jedoch keines der untersuchten Graphikpakete in einer verwendbaren Form an, sodass darauf zunächst verzichtet werden musste.

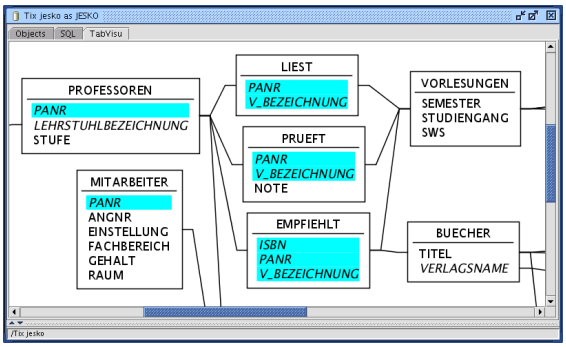
Eine weitere Funktion, die bei der Visualisierung geplant war, ist die Beschränkung auf Tabellen, die in einer Anfrage verwendet werden. Damit soll die schrittweise Erstellung von Anfragen und das Verständnis von Verbundanfragen unterstützt werden. Theoretisch werden über das JDBC-Interface alle dafür notwendigen Informationen bereitgestellt, d.h. Namen von Tabellen und Attributen, die im Ergebnis der Anfrage erscheinen. Daraus ließe sich ermitteln, welche Tabellen an der Anfrage beteiligt sind und welche Beziehungen bestehen. In der Praxis hat sich allerdings gezeigt, dass viele Treiber die notwendigen Methoden nicht implementieren. Der Treiber für Oracle lieferte beispielsweise keinen Tabellennamen. Daher konnte diese Funktion ebenfalls nicht realisiert werden.

3.2 Vorlesungsvideos

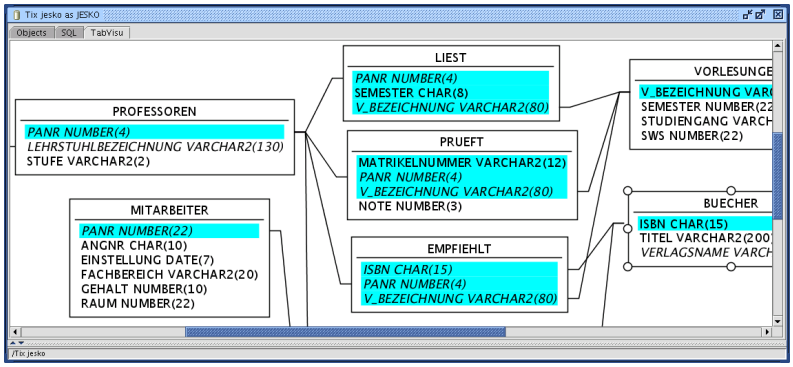
Durch die Bereitstellung der Vorlesungsvideos sollen Studierende bei der Wiederholung und Vertiefung von Lehrinhalten unabhängig von Zeit und Ort unterstützt werden. Dazu wurde eine Webseite entwickelt, so dass jeder zu deren Nutzung nur einen gängigen PC, einen Internetanschluss und einen geeigneten Video- und Audiodekoder benötigt. Eine Einschränkung auf ein bestimmtes Betriebssystem (Linux, Windows, ...) sollte vermieden werden. Dies wurde durch den konsequenten Einsatz offener, allgemein akzeptierter und unterstützter Technologie wie XML, XSL, HTML, DivX, MP3 und JavaScript erreicht. Eine weitere Forderung war die



(a) Tabellennamen



(b) Tabellennamen und alle Attribute



(c) Alle Informationen

Abbildung 3: Grafische Darstellung von Tabellen und Beziehungen aus einer Datenbank im SQuirreL SQL Client mit verschiedenen Detaillierungsstufen

Unabhängigkeit von den in der Vorlesung verwendeten Medien. Es soll keine Rolle spielen,

ob der Dozent Folien oder eine Powerpoint-Präsentation verwendet. Die verwendete Technik sollte sich auf ein schnurloses Mikrofon sowie einen handelsüblichen, digitalen Camcorder mit FireWire-Anschluss beschränken. An spezieller Software werden ein Webserver (Apache), Software zur Verarbeitung der XML-Dateien, Schnittsoftware sowie geeignete Codecs benötigt. Als Hardware für den Webserver haben wir einen durchschittlichen PC unter Linux eingesetzt.

Die Webschnittstelle zeigt als kleinste Struktureinheit Videoclips, die jeweils zu genau einem Folienbild korrespondieren. Damit können Studierende jeweils spezielle Themen auswählen und brauchen nur die für sie interessanten Clips zu laden. Die Zerlegung eines Videos in einzelne Clips ermöglicht somit ein genaues Navigieren und spart unnötigen Übertragungsaufwand. Zur Navigation unterstützen wir zwei Strukturierungsvarianten mit folgenden Strukturebenen:

nach Terminen: Der Zugriff erfolgt an Hand der Vorlesungen und Termine, d.h. zunächst wird die *Vorlesung* ausgewählt, dann der *Vorlesungstermin* und ein *Thema* (umfasst durchschnittlich ca. 6 Folien). Schließlich kann eine einzelne Folie mit den zugehörigen Aufzeichnungen gewählt werden. In Abbildung 4 ist diese Strukturierung dargestellt.

nach Themen: Der Zugriff erfolgt an Hand von Themen mit zunehmender Spezialisierung. In diesem Fall werden zunächst ein *Thema* und ein *Unterthema*. Dann kann wieder eine bestimmte Folie zu diesem Thema mit den zugehörigen Medien ausgewählt werden.

Das Video zeigt den Dozenten oder die Erstellung eines Tafelbildes. Die wichtigste Information ist allerdings das Gesprochene, also das Audio-Signal. Der Nutzer kann weiterhin auswählen, ob nur die Audio-Spur oder auch die Video-Spur präsentiert werden soll. Dadurch lässt sich das Übertragungsvolumen der verfügbaren Bandbreite entsprechend anpassen.

Die Beschreibung des Erstellungsprozesses würde den Rahmen dieses Berichtes sprengen. Eine umfassende Beschreibung geben wir in [SJS03, Jes03]. Es sollen im folgenden nur die wesentlichen Arbeitsschritte kurz erwähnt werden:

Aufbereitung der Folien: Zunächst wurden die Folien, die in unserem Fall als PostScript-Dateien vorlagen, so aufgeteilt, dass für jede Seite eine Datei vorlag. Weiterhin wurden die einzelnen Dateien in die verschiedenen notwendigen Formate unterteilt. Diese Aufgabe übernahm ein Shell-Script.

Erstellung der Medien: Das aufgezeichnete Video wurde mittels einer Schnittsoftware in die gewünschten Segmente aufgeteilt in das DivX- und MP3-Format kodiert. Diese Formate erwiesen sich als besonders geeignet, da sie bei den erforderlichen, geringen Datenraten noch eine ausreichende Qualität liefern. Für diese beiden Arbeitsschritte erwiesen sich VirtualDub [Vir03] unter Windows und transcode [Tra03] unter Linux als die geeignetste Lösung. Insbesondere, da diese eine Batchbearbeitung oder den Einsatz in Shell-Skripten ermöglichen. Damit ließ sich der Anteil manueller Arbeitsschritte reduzieren.

Erfassung der Metadaten: Neben den Folien und dem Video werden weitere auf der Web-Seite weitere Informationen dargestellt, etwa Verweise auf Bücher, Schlagworte etc.

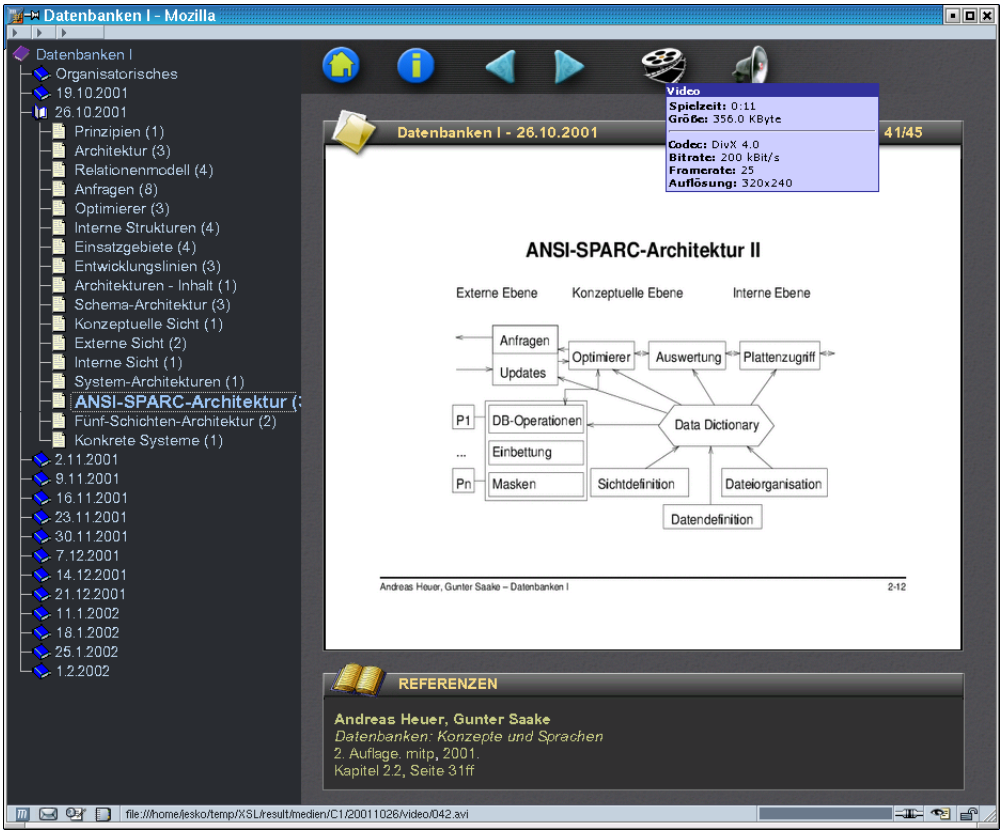


Abbildung 4: Gesamtansicht der Webseite (Menü nach Terminen strukturiert)

sowie technische Daten der Medien (Dateiformate, Größe, Spieldauer, etc.). Schließlich sind noch Informationen für die Generierung der HTML-Seiten erforderlich, z.B. Dateiname der Folien und Themen zur Gruppierung. Die Verwaltung dieser Metadaten erfolgt in einer XML-Datei. Vor der Erstellung der HTML-Dateien müssen daher alle notwendigen Informationen erfasst werden. Dies erfolgt vorwiegend in Handarbeit. Lediglich die Eingabe einiger technischer Daten wurde durch Skripte automatisiert.

Erstellung der WWW-Seite: Mittels XSL Transformations (XSLT) werden aus den, in XML beschriebenen, Metadaten automatisch die HTML-Seiten erzeugt und strukturiert. Dabei kommt Xalan [Xal03] als XSLT-Prozessor zum Einsatz. Alle erstellten Dateien müssen abschließend ggf. noch an eine für den Web-Server erreichbare Stelle kopiert werden.

Die für diesen Ansatz verwendeten Metadaten sind teilweise so allgemein gehalten, dass sie auch in anderem Zusammenhang verwendet werden können. Ein Teil der Metadaten nimmt lediglich eine Einordnung jeder einzelnen Folie in eine vierstufige Hierarchie vor und beinhaltet

z.B. Titel der einzelnen Blöcke. Weiterhin werden jeder Folie, unabhängig von den Videoaufnahmen, u.a. Referenzen und Schlagworte zugeordnet. Im Folgenden soll daher noch kurz auf die Verwaltung der Metadaten eingegangen werden.

Um einen Überblick zu bekommen, zeigt Abbildung 5 die Metadaten in Form eines UML-Klassendiagramms. Die eigentliche Verwaltung erfolgt, wie bereits erwähnt, in einer XML-Datei. Erfasst werden Informationen zu den Folien und deren inhaltlicher Strukturierung, sowie zu den eigentlichen Veranstaltungen und den dabei erstellten Medien (Video- und Audiodateien). Diese Aufteilung in drei Gruppen ist durch die Packages in Abbildung 5 angedeutet, die folgende Elemente zusammenfassen:

`General`: beinhaltet allgemeine Elemente, z.B. `Book`, `VFormat` und `AFormat`, für die Spezifikation der Informationen zu Büchern und den verwendeten Codierungen.

`Content`: beinhaltet Elemente zur Beschreibungen einzelner Folien und deren Einordnung. Die verwendete vierstufige Hierarchie orientiert sich am LOM-Standard [IEE02]. Im Abschnitt des Koordinationsteams 2 wird noch genauer auf diesen Standard eingegangen. Die Struktur spiegelt sich in den Elementen `LearningUnit` (Materialien zu einem Themengebiet, z.B. „Anfragesprachen“), `LearningModule` (Materialien zu einem Unterthema, z.B. „SQL“), `GroupObject` (Sammlung mehrerer zusammenhängender Folien, z.B. „select-Klausel I“ bis „select-Klausel III“) und `MediaObject` (einzelne Folie und zugehörige Informationen) wieder.

`Courses`: beinhaltet schließlich die Informationen zu konkreten Kursen. Dies umfasst Referenzen zu den verwendeten Folien (`MediaObject`s), Informationen zu den Medien (`Video`, `Audio`) und die Strukturierung in Veranstaltungen (`Lecture`) und Themen `Topic`. Letztere beinhalten lediglich einen Titel für eine Reihe von `MediaObject`-Elemente. Dieser könnte auch aus dem zugehörigen `GroupObject` ermittelt werden. Wir verwenden jedoch aus zwei Gründen eine Kopie des Titels. Einerseits vereinfacht sich das XSLT-Skript zur Erstellung der HTML-Dateien. Andererseits besteht die Möglichkeit, bei Bedarf einen kürzeren Titel für die WWW-Seite anzugeben. Aus dem Diagramm ist ersichtlich, dass die Segmente und somit die `MediaObject`s auf zwei Arten einem `Course` zugeordnet werden können. Zum einen über `Lecture` und `Topic` und zum anderen über `Info`. Letztere Zuordnung wurde speziell für Segmente mit allgemeinen Inhalten eingeführt, z.B. solchen mit organisatorischen Informationen. Derartige `MediaObject`s werden auch nicht in die Hierarchie eingeordnet. Während die Angaben in der zweiten Gruppe (`Content`) wiederverwendbar sind, ist dies bei den Informationen zu den Kursen in der Regel nicht der Fall, da sich die technischen Daten der Medien (z.B. Länge und Größe der Clips) mit jeder Veranstaltung ändern, auch wenn identische Folien verwendet werden.

Für die Beschreibung der Abbildung auf die DTD und die ausführliche Erläuterung der einzelnen Elemente sei abschließend auf [SJS03, Jes03] verwiesen. Im Rahmen des Projektes wurden diese Daten für alle Folien der Vorlesung erfasst.

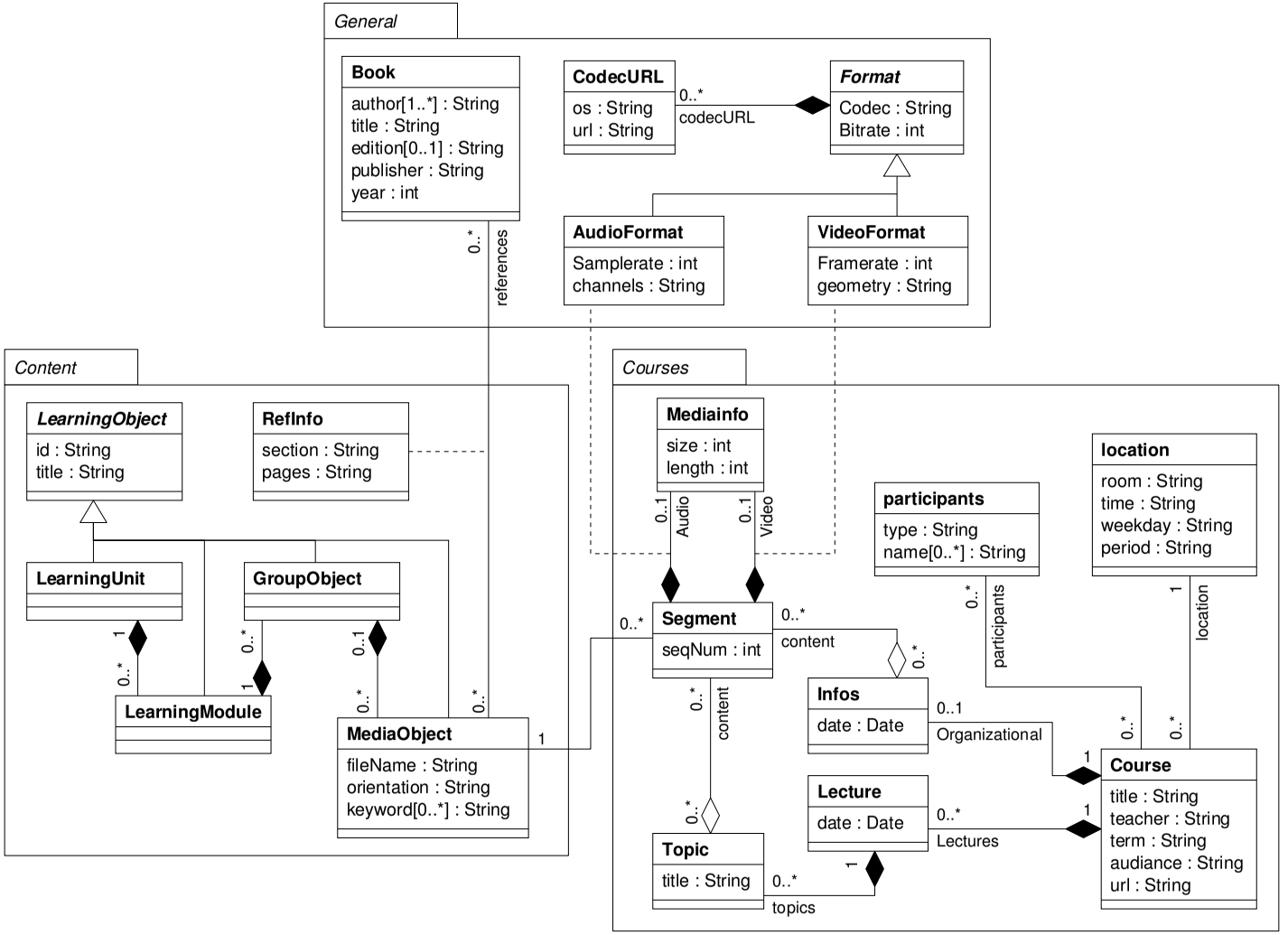


Abbildung 5: Daten zur Vorlesung als UML-Klassendiagramm

4 Erfahrungen

Ziel des Projektes war die Unterstützung der Lehre im Bereich Datenbanken durch „neue“ Medien zur Kommunikation (Internet, ...) und Präsentation (Multimedia, ...). Befragungen bei den Studierenden haben ergeben, dass derartige Angebote vorrangig positiv aufgenommen wurden. Im Laufe der Bearbeitung hat sich eine Reihe von Ansatzpunkten für eine Unterstützung durch Medien und Werkzeuge ergeben. Jedoch hat sich auch gezeigt, dass die Erstellung, insbesondere von multi- und hypermedialen Materialien sehr aufwendig ist. Die Planung und Erstellung von Animationen und Videos hat sich beispielsweise als schwieriger und aufwendiger erwiesen, als anfangs erwartet. Die Erstellung der Drehbücher, die Aufnahmen und der Schnitt ist nur mit professioneller Hilfe möglich, wenn Filme einer „brauchbaren“ Qualität entstehen sollen. Dies gilt insbesondere, wenn Szenen durch Schauspieler nachgestellt werden. Die daraus resultierenden Produktionskosten erlauben den Einsatz derartiger Materialien daher nur in sehr beschränktem Umfang.

Weiterhin ist der Einsatz auf Grund des hohen Erstellungsaufwandes nur für Themengebiete sinnvoll, die sich inhaltlich nicht oder nur wenig ändern, z.B. Grundlagenvorlesungen. Für Lehrveranstaltungen, bei denen die Inhalte schnell „veraltern“, ist der Einsatz vieler Materialien, auf Grund des hohen Aufwands für die Erstellung und Pflege, nur bedingt möglich. Der Einsatz von realen Videos als Grundlage von Übungsaufgaben erscheint uns ebenfalls als kaum realisierbar, da sich diese relativ häufig ändern, was wiederum einen nicht vertretbaren Aufwand bei der Anpassung der Medien verursacht. Als Beispiel sollen abschließend die in Zusammenarbeit mit mediaproject erstellten Videoaufnahmen erwähnt werden. Obwohl die beiden für das Teilprojekt interessanten Filme nur wenige Minuten lang sind und lediglich eine kurze Motivation und einen Teil eines Beispiels zur Modellierung bieten, hat die Konzipierung, Erstellung und Überarbeitung der (rudimentären) Drehbücher mehrere Tage in Anspruch genommen.

Im Rahmen des Projektes wurde durch den Ansatz für die Videoaufzeichnungen versucht einen Kompromiss zu finden, indem die Studierenden bei der Wiederholung des Stoffes unterstützt werden. Durch verschiedene Verfahrensweisen und Skripte konnte hierbei der Aufwand in vertretbaren Grenzen gehalten werden. Abschließend sollen die damit gemachten Erfahrungen kurz erwähnt werden, wobei vorrangig auf technische Aspekte eingegangen wird. Wir haben im Anschluss an die Vorlesung auch eine Befragung der Studierenden durchgeführt. Die Ergebnisse sind im Abschnitt 6 detaillierter beschrieben.

Vom technischen Standpunkt hat sich gezeigt, dass für die Aufnahmen ein normaler digitaler Camcorder hinreichend ist. Eine professionelle Ausrüstung ist nicht erforderlich, da die Qualität durch die spätere Kodierung ohnehin stark reduziert wird. Die Aufbereitung der Videos und insbesondere der Metadaten erwies sich trotz der Automatisierung durch die verschiedenen Skripte als sehr arbeitsintensiv. So wurden für die Aufzeichnung und Aufbereitung einer 90-minütigen Vorlesung etwa 8 bis 10 Stunden benötigt. Bei einer Wiederverwendung ist allerdings mit einem geringeren Aufwand zu rechnen, da insbesondere die aufwendige Eingabe diverser Metadaten nicht erneut erfolgen muss. Weiterhin stellte sich die Frage, welche Technik für die Bereitstellung erforderlich ist. Es hat sich gezeigt, dass für die ca. 200 Studierenden, die an der Vorlesung teilnahmen, kein spezieller Server notwendig ist. Die Ablage der etwa 2,5 GB und der monatliche Transfer von etwa 30GB konnte durch einen „normalen“ PC bewältigt werden. Bei einer größeren Anzahl von Studierenden dürfte dieser allerdings bald

Tabelle 1: Technische Daten

Merkmal	Quantität
DV-Rohdaten je Vorlesungstermin (90 min)	20 GB
Gesamtes Material für WWW-Seite je Vorlesungstermin	160 MB
davon Videodaten	150 MB
Anzahl Folien je Vorlesungstermin (min/max/avg)	17/46/30
Größe der Videoclips (min/max/avg)	0,3/36,5/5 MB
Dauer eines Videoclips in Minuten (min/max/avg)	0:08/20:13/3:00
Umfang der WWW-Seite von zwei Veranstaltungen	4,5 GB
Zeit für Aufnahme und Bearbeitung eines Vorlesungstermins	6 bis 10 Stunden

an Grenzen stoßen. Tabelle 1 fasst einige der wesentlichen Ergebnisse der Evaluierung der technischen Aspekte zusammen.

5 Zusammenfassung

Ziel des Teilprojektes 1.2 war die Verbesserung der Lehre im Bereich der Datenbanken und Informationssysteme durch den Einsatz „neuer“ Medien zur Kommunikation und Präsentation. Die Grundlage für die Entwicklungen bildeten die bereits existierenden Vorlesungen „Datenbanken I“ und „Datenmanagement“. Diese werden bereits seit mehreren Jahren regelmäßig für Studierende verschiedener Fachrichtungen angeboten. Eine vollständige Unterstützung der gesamten Vorlesung mit multimedialen Materialien, z.B. videogestützte Beispiele, Animationen, Werkzeuge für praktische Übungen, war auf Grund des großen Stoffumfangs während der Projektlaufzeit nicht realisierbar. Daher wurden im wesentlichen drei Teilaufgaben bearbeitet:

1. Videoaufnahmen realer Vorgänge zur Motivation und für ein Beispiel
2. Anpassung eines Werkzeugs für Übungen mit SQL und Bereitstellung der Anfragen aus der Vorlesung
3. Erarbeitung eines Ansatzes zur effizienten Aufzeichnung und Bereitstellung von Videoaufnahmen der Vorlesung.

Die Erstellung der Videos beschränkte sich auf die Konzipierung der Drehbücher. Die eigentlichen Dreharbeiten wurden von einem externen Unternehmen übernommen. Nur so konnte unserer Ansicht nach die geforderte Qualität erreicht werden.

Für das Gebiet der Anfragesprachen bot sich die Erweiterung eines bestehenden Werkzeugs (SQuirreL SQL Client) an, da dieses bereits gute Vorausetzungen bot und nur geringfügige Anpassungen notwendig waren. Eine umfangreiche Erweiterung war die Erstellung eines Plugins für die Visualisierung der Tabellen in einer Datenbank und deren Beziehungen. Erfahrungsgemäß fördert eine derartige Darstellung das Verständnis bei der Definition von Anfragen in SQL. Durch die Möglichkeit, vorbereitete Anfragen zu laden und zu editieren bietet sich dieses Werkzeug auch für den Einsatz in Vorlesungen und Seminaren an.

Schließlich wurde ein Ansatz zur Bereitstellung von Vorlesungsvideos entwickelt und beim Einsatz im Wintersemester 2001/02 („Datenbanken I“) und im Wintersemester 2002/03 („Multimediadatenbanken“) evaluiert. Eine Befragung der Studierenden hat dabei gezeigt, dass die Videos vorwiegend positiv bewertet wurden. Es hat sich anfangs gezeigt, dass die Bereitstellung in kurzen Sequenzen mit einem erheblichen Arbeitsaufwand verbunden ist. Durch den Einsatz diverser Skripte und die Wiederverwendung der Metadaten konnten zwischenzeitlich jedoch viele Arbeitsschritte automatisiert werden. Durch weitere Verbesserungen der Abläufe und Werkzeuge sollte sich der manuelle Aufwand auf etwa die doppelte Vorlesungszeit reduzieren lassen.

6 Evaluierung

Im Wintersemester 2001/02 wurde erstmals eine Vorlesung („Datenbanken I“) in der zuvor beschriebenen Art aufgezeichnet und bereitgestellt. Die Vorlesung wurde von etwa 200 Studierenden besucht. Im praktischen Einsatz wurde untersucht, wie aufwendig die Erstellung ist und welche technische Voraussetzungen gegeben sein müssen. Die Ergebnisse dieser technischen Evaluierung wurden bereits im Abschnitt 4 erläutert. Weiterhin sollte untersucht werden, ob die Videos von den Studierenden angenommen werden. Dies erfolgte zum Ende des Semesters mittels eines Fragebogens, sowie durch Gespräche mit einigen Studierenden. Zunächst war dabei von Interesse, ob seitens der Studierenden die technischen Möglichkeiten für den Zugriff auf die Materialien zur Verfügung stehen. Daher wurden zunächst gefragt, welche Art der Netzwerkverbindung bestand. Hier hat die Auswertung ergeben, dass etwa 90% der befragten Studierenden entweder das Uni-Netzwerk nutzten oder eine DSL-Verbindung besaßen, so dass auch die Übertragung größerer Datenmengen kein Problem darstellte. Hinzu kam, dass die Studierenden selbst CDs anfertigten und verbreiteten.

Wünschenswert wäre eine Untersuchung hinsichtlich der Videonutzung auf die Häufigkeit der Vorlesungsbesuche gewesen. Diese lässt sich allerdings nur bedingt durchführen, da keine Anwesenheitslisten geführt werden und auch die Nutzung des Videos nicht eindeutig bestimmt werden kann. Letzteres insbesondere, da die Aufnahmen auch auf CD bereitgestellt oder von den Studierenden zunächst vollständig heruntergeladen und abschließend offline betrachtet wurden. Daher beschränkte sich der Fragebogen auf die Aussagen der Studierenden. Für die Nutzung der Videos und die Besuche der Vorlesung wurden vier Möglichkeiten von „nie“ bis „häufig“ angeboten. Weiterhin bestand die Möglichkeit die persönliche Meinung zu äußern. Einige Ergebnisse sind in Tabelle 2 zusammengefasst. Von den Befragten gaben jeweils etwa 50% an, dass sie die Vorlesung häufig, bzw. selten besuchten. Aufgrund der zuvor erwähnten Probleme bei der objektiven Bewertung sind diese Zahlen allerdings nur bedingt aussagekräftig. Weiterhin wurde gefragt, welche Art der Bereitstellung (kleine Szenen oder ganze Vorlesung) bevorzugt und ob das Angebot als hilfreich angesehen wurde. Hier zeigte sich, dass etwa 60% der Befragten die thematische Gliederung bevorzugten. Die überwiegende Zahl der Befragten (etwa 70%) sah das Angebot als hilfreich an.

Zusammenfassend lässt sich aus der Befragung ableiten, dass die Ergänzung und Bereitstellung der Folien mit Video- und Audiomitschnitten in der beschriebenen Form von den Studierenden überwiegend positiv aufgenommen wurde. Nur wenige gaben an, dass es für sie keinen Vorteil hatte. Erwartungsgemäß wurde das Video häufiger genutzt, wenn die Vorlesung

Tabelle 2: Nutzung des Videos

Kategorie	Vorl.-besucher	Vorl.-abwesende
nie	15%	0%
gelegentlich	61%	47%
häufig	15%	46%
keine Angabe	9%	7%

selten besucht wurde. Inwieweit das Videoangebot und der Vorlesungstermin diese Ergebnisse beeinflusst lässt sich allerdings nicht feststellen. Um brauchbare Aussagen zu erhalten müsste die Vorlesung mehrfach mit dem Video angeboten werden. Auch eine objektive Bewertung einer Verbesserung der Lernergebnisse, z.B. an Hand von Prüfungsergebnissen, ist aus den genannten Gründen nicht möglich.

Im Wintersemester 2002/03 wurde, unabhängig vom MuSofT Projekt, eine weitere Vorlesung („Multimediadatenbanken“) aufgezeichnet und bereitgestellt. Die Ergebnisse waren ähnlich.

Literatur

[Bar00] BARDOHL, ROSWITHA: *GenGEd: Visual Definition of Visual Languages based on Algebraic Graph Transformation*. Verlag Dr. Kovač, 2000. (Dissertation, Technische Universität Berlin, FB Informatik).

[BMG00] BEDERSON, BENJAMIN B., JON MEYER, and LANCE GOOD: *Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java.* In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Toolkit Support for UI, pages 171–180, 2000.

[Che76] CHEN, P. P.: *The Entity-Relationship Model – Towards a Unified View of Data*. ACM Transactions on Database Systems, 1(1):9–36, 1976.

[Cod70] CODD, E.: *A Relational Model for Large Shared Data Banks*. Communications of the ACM, 13(6):377–387, June 1970.

[Cod90] CODD, E.: *The Relational Model for Database Management*, volume Version 2. Addison-Wesley, Reading, MA, 1990.

[DD99] DATE, C. J. and HUGH DARWEN: *A Guide to the SQL Standard: A User's Guide to the Standard Database Language SQL*, volume 4. Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.

[EN00] ELMASRI, R. and S. NAVATHE: *Fundamentals of Database Systems*. Benjamin Cummings, Redwood City, CA, 3 edition, 2000.

[IEE02] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERING, INC., LEARNING TECHNOLOGY STANDARDIZATION COMMITTEE (LTSC): *Draft Standard for Learning Object Metadata (LOM)*, July 2002. http://ltsc.ieee.org/.

[isq03] *WWW-Seite zum iSQL-Viewer*. online, 2003. http://isql.sourceforge.net/.

[Jes03] JESKO, DIRK: *WWW-Seite zur Aufnahme von Vorlesungen*. online, 2003. http://wwwiti.cs.uni-magdeburg.de/iti_db/forschung/musoft/video/.

[Mas03] MASSACHUSETTS INSTITUTE OF TECHNOLOGY: *MIT OpenCourseWare*. online, 2003. http://ocw.mit.edu/.

[mim03] *WWW-Seite zum Mimer SQL Validator Web Service*. online, 2003. http://sqlvalidator.mimer.com/index.html.

[Min03] MINQ SOFTWARE: *WWW-Seite zum DBVisualizer*. online, 2003. http://www.minq.se/products/dbvis/index.html.

[MK99] MINAS, MARK and OLIVER KÖTH: *Generating diagram editors with DiaGen*. In NAGL, MANFRED, ANDY SCHÜRR, and MANFRED MÜNCH (editors): *Proceedings: Applications of Graph Transformations With Industrial Relevance: International Workshop and Symposium AGTIVE'99, Kerkrade, The Netherlands, September 1-3*, volume 1779 of *Lecture notes in computer science*, pages 433–440, Berlin, Heidelberg, 1999. Springer-Verlag Inc.

[SJS03] SCHMITT, INGO, DIRK JESKO und GUNTER SAAKE: *Multimediaunterstützung von Vorlesungen – Ein Erfahrungsbericht*. Preprint 14, Otto-von-Guericke-Universität Mageburg, November 2003.

[Squ03] *WWW-Seite zum SQuirreL SQL Client*. online, 2003. http://squirrel-sql.sourceforge.net/.

[Stu03] STUTTGART, UNIVERSITÄT: *100-online*. online, 2003. http://www.uni-stuttgart.de/100-online/.

[The03] THE UNIVERSITY OF TEXAS AT AUSTIN: *World Lecture Hall*. online, 2003. http://www.utexas.edu/world/lecture/.

[Tra03] *WWW-Seite zum Linux Video Stream Processing Tool*, 2003. http://www.theorie.physik.uni-goettingen.de/~ostreich/transcode/.

[Vir03] *WWW-Seite zur VirtualDub*, 2003. http://www.virtualdub.org/.

[Xal03] *WWW-Seite zum XSLT-Prozessor Xalan*. online, 2003. http://xml.apache.org/xalan-j/index.html.

Der MuSofT-Abschlussbericht

Olaf Scheel, Johannes Magenheim

Abschlussbericht für die Lerneinheit 1.3 *Softwareengineering in der Informatiklehrerausbildung* und die Lerneinheit 2.4. *Dekonstruktion von Softwaresystemen*.

Inhaltsverzeichnis

1 Einleitung

Dies ist der Abschlussbericht für die Lerneinheit 1.3: *Softwareengineering in der Informatiklehrerausbildung* und die Lerneinheit 2.4: *Dekonstruktion von Softwaresystemen*. Beide Lerneinheiten wurden für die Informatiklehrer-Ausbildung an der Universität Paderborn gestaltet, große Teile sind bereits eingesetzt und evaluiert worden.

Die Materialien zur Lerneinheit *Softwareengineering in der Informatiklehrerausbildung* und der damit zusammenhängenden Fallstudie *Hochregallager* kamen in Teilen während der Veranstaltungen *Didaktik der Informatik* und *Grundlagen der Informatik für Lehramtsstudierende* im Jahr 2002 zum Einsatz. Die gesamte Fallstudie wurde dann im Seminar *Informatische Lernwerkstatt für die Sekundarstufe II* im Sommersemester 2003 an der Universität Paderborn eingesetzt und dort auch evaluiert.

Die Materialien zur Lerneinheit *Dekonstruktion von Softwaresystemen* und die zugehörige Fallstudie *Schulkiosk/Warenwirtschaftssystem* wurden im Wintersemester 2003/2004 im Rahmen des Seminars *Lernwerkstatt Informatik* eingesetzt.

Im Verlauf des Jahres 2003 wurde vor allem die Arbeit an den beiden Fallstudien verstärkt. Sie wurden mit weiteren multimedialen Materialien angereichert, um den Studierenden jeweils eine variantenreiche Explorationsumgebung für die Erkundung des zugrunde liegenden Informatiksystems zu bieten. Es hatte sich bei den ersten Einsätzen der Materialien gezeigt, dass die Wiederverwendbarkeit in verschiedenen Veranstaltungen und Veranstaltungsformen eher gegeben ist, wenn die Module nicht in sich abgeschlossen, sondern offen - im Extremfall sogar fragmentarisch - sind. Dies geschieht in Übereinstimmung mit dem ursprünglichen Grundgedanken des MuSofT-Projektes, primär Präsenzveranstaltungen multimedial zu unterstützen und keine eigenständigen, geschlossenen Kurse für das Online-Lernen bereitzustellen.

Insgesamt fokussiert der hier vorliegende Abschlussbericht in seiner Ergebnisdarstellung auf die erzeugten multimedialen Produkte (content), die didaktisch-methodischen Rahmenbedingungen ihres Einsatzes (context) sowie auf die aus der Evaluation gewonnenen Erkenntnisse über den praktischen Einsatz der Materialien und weniger auf den Prozess ihrer Herstellung während der Laufzeit des Projekts. Dieser kann detaillierter den vorhergehenden Jahresberichten entnommen werden.

2 Vorstellung der Lerneinheiten

2.1 Was soll gelehrt werden

2.1.1 Lerneinheit 1.3

In der Lerneinheit *Softwareengineering in der Informatiklehrerausbildung* soll den Studierenden anhand der Lerneinheit übergreifenden Fallstudie *Hochregallager* (siehe auch Teilprojekt Engels *Video-gestützte Anforderungsdefinition* und Saake *Entwicklung von Informationssystemen*) das Konzept des *soziotechnischen Informatiksystems* und damit zusammenhängend der *Systemorientierten Didaktik* [Mag01] vermittelt werden.

Die einzelnen Lernmodule der Lerneinheit beschäftigen sich darüber hinaus mit Methoden der Software-Technik und untersuchen sie im Hinblick auf ihre Verwendbarkeit in (schul-)unterrichtlichen Zusammenhängen. Lernmaterialien sind zum einen Texte zu den oben ge-

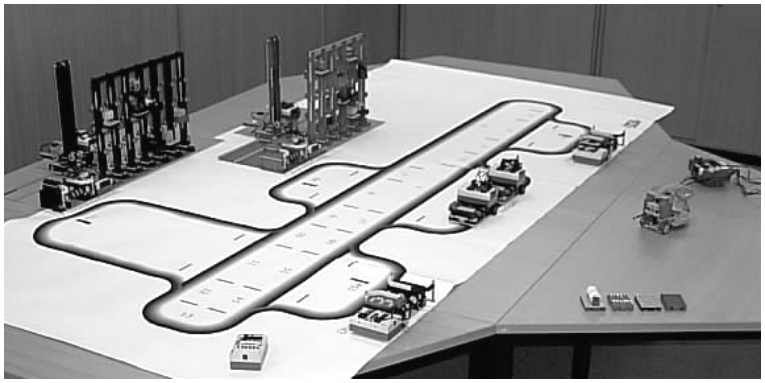


Abbildung 1: Das LEGO Hochregallager

nannten Konzepten, aber vor allem multimediale Materialien zur Fallstudie Hochregallager (in Abbildung 1 ist das LEGO-Modell eines solchen Lagers zu sehen). Inhalte können mit Hilfe der Methode der Dekonstruktion aus dem bereitgestellten Materialien erarbeitet und zu einem individuellen Portfolio zusammengestellt werden. Darüber hinaus sind die Materialien zur Fallstudie als Beispiele für theoretische Konzepte der Informatik verwendbar. Zum Materialpool gehören ferner Erkundungs- und Entwicklungsaufgaben rund um das Szenario einer automatischen Kommissionierstation, in dem die Studierenden ihr explorativ am Hochregallager erworbenes Wissen im Sinne des blended learning konstruktiv einsetzen können. [Mag03c]

Konkret beschäftigt sich das Modul 1 *Soziotechnisches Informatiksystem* (StIs) mit dem vorgenannten Konzept - ausgehend von Informatiksystemen und ihrer Bedeutung für die Fachwissenschaft und -didaktik. Der Begriff des StIs wird eingeführt, das man verkürzt als Informatiksystem mit seinem technischen und sozialen Umfeld charakterisieren kann. Die Fallstudie Hochregallager dient dabei als Beispiel für ein solches System. Das nachfolgende Modul 2 *Systemorientierte Didaktik* beschäftigt sich mit Unterrichtsinhalten und Methoden einer Fachdidaktik, die sich an dem Konzept des StIs orientiert. Beispiele für diese beiden Teilaspekte werden wieder aus der Fallstudie der Lerneinheit gewählt. Die beiden anschließenden Module beschäftigen sich nun näher mit diesen Methoden. In Modul 3 aus einer fachwissenschaftlichen Perspektive[1], in Modul 4 aus einer fachdidaktischen. Innerhalb dieses Moduls wird ein mögliches Vorgehensmodell für den Informatikunterricht entwickelt, das durch Beispiele und Aufgaben mit der Fallstudie verknüpft wird. Das letzte Modul der Lerneinheit beschäftigt sich dann mit den verschiedenen Rollen der an Entwicklungsprozessen beteiligten Personen.

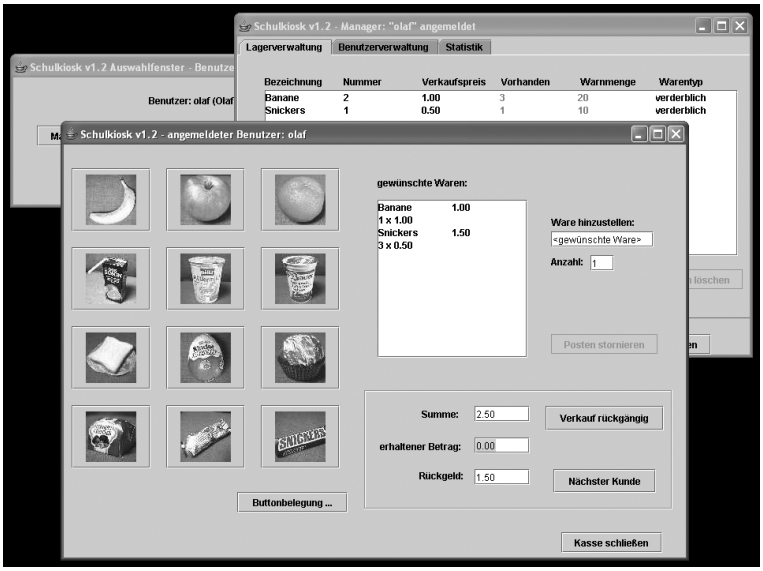


Abbildung 2: Oberfläche des Schulkiosk v1.2

2.1.2 Lerneinheit 2.4

Die Lerneinheit *Dekonstruktion von Softwaresystemen* setzt sich mit einzelnen Teilaspekten des didaktischen Ansatzes der Dekonstruktion von Softwaresystemen und von soziotechnischen Informatiksystemen auseinander. Als Fallstudie wird hier ein *Schulkiosk/Warenwirtschaftsystem* (Abbildung 2) verwendet. Die einzelnen Module enthalten Lernobjekte zu den damit verbundenen (Analyse-)Methoden, Sichtweisen auf Informatiksysteme, aber auch zum Gebrauch von Werkzeugen.

Während in der Lerneinheit 1.3 parallel zur Fallstudie Arbeitsaufträge zu einem neuen Szenario von den Studierenden bearbeitet werden können, soll in den praktischen Phasen dieser Lerneinheit durch die Studierenden eine Erweiterung des bestehenden Systems zu einem Online-Shop geleistet werden. Somit steht hier weniger die Konstruktion eines neuen Systems im Vordergrund, wie in Lerneinheit 1.3, sondern der Gedanke des Reengineerings. Da die von uns implementierte Schulkiosk-Software in Java nach dem MVC-Muster entworfen wurde, ist dies von den Studierenden nach einer Erkundungsphase mit didaktischen Fenstern (Näheres dazu im Kapitel *Aufbau*) in das Informatiksystem zu leisten. Dabei kommen verschiedene Ausbaustufen der Schulkiosk-Software zum Einsatz. (Siehe dazu das Kapitel *Aufbau*.)

[1] Dies schließt die Beschäftigung mit Konzepten der Softwaretechnik wie Entwurfsmuster und Vorgehensmodelle ein.

2.2 Didaktisches Konzept

Das didaktische Konzept für die Lerneinheiten 1.3 und 2.4 ist geprägt durch den Begriff des *Informatik Lernlabors*. Hierunter soll verstanden werden, dass die innerhalb des MuSofT-Projektes für die Präsenzlehre entwickelten Materialien nicht nur im Präsenzteil (im engeren Sinne) der Veranstaltungen genutzt werden sollen, sondern auch bei der Nachbereitung durch die Studierenden und für Übungen. Die Materialien sollen dabei für einen nach konstruktivistischen Ideen ausgerichteten Lernprozess geeignet sein und nicht ausschließlich für dozentenzentrierte Phasen der Präsenzveranstaltung. Schlüsselbegriffe sind hier: blended learning, Erkundung von didaktische Szenarien, kooperatives Lernen in Lerngruppen und didaktische Fenster. Die für die Lerneinheiten erstellten Materialien werden auf den am Institut vorhandenen Steam-Server[2] als Lernplattform[3] eingestellt und somit für die Studierenden nutzbar.

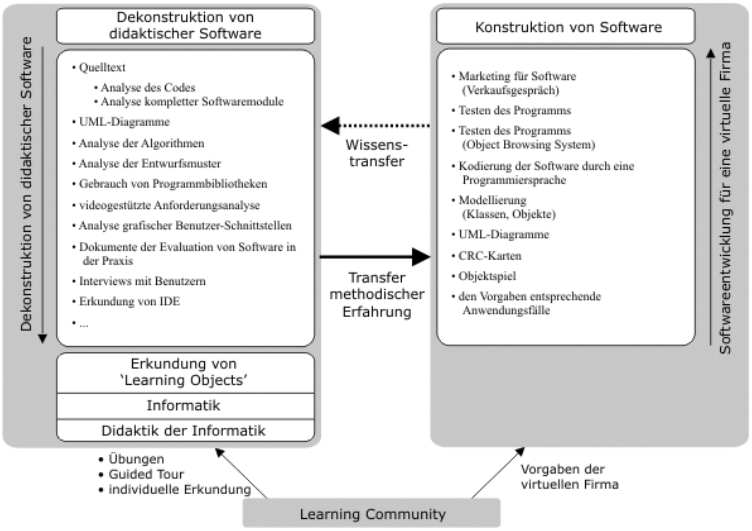


Abbildung 3: Lernprozesse im Informatik Lernlabor

Das MuSofT-Projekt soll Präsenzlehre durch multimediale Lernmodule unterstützen und aufwerten. Aus diesem Grund wurden innerhalb des Teilprojektes keine Materialien für reine Online-Kurse ohne Präsenzphasen entwickelt. Stattdessen wurden Lernobjekte erstellt, die für den gezielten punktuellen Einsatz in Präsenzveranstaltungen wie Vorlesungen oder auch Seminaren geeignet sind, aber auch für die Nacharbeitung der Studierenden, für Selbstlernphasen, Übungen und durch Groupware unterstützte Präsenzseminare, in denen netzgestützt kooperativ gearbeitet wird. Unterstützt werden sollen also unterschiedliche Phasen des blended learning mit unterschiedlich starker Aktivität der Studierenden und des Dozenten. - All dies zusammengefasst unter dem Schlagwort des *Lernlabors*, für die zum jetzigen Zeitpunkt

[2] http://steam.upb.de/ bzw. http://gauge.upb.de/

[3] Das innerhalb des Projektes entwickelte Portal dient nur als Distributionsplattform für Dozenten, ist aber nicht als Lernplattform für Studierende gedacht.

bereits ein weiteres Modul *Computerspiel* im Rahmen eines anderen Projektes entstanden ist. Ein viertes Modul *Online Shop* befindet sich in Vorbereitung.

Das didaktische Konzept des Lernlabors beruht dabei aus Sicht der Studierenden auf vier Säulen:

- Unterweisung durch den Dozenten unter exemplarischer Benutzung der für die Fallstudie erstellten Materialien
- Selbstständige Erkundung der Materialien mit Hilfe einer Lernplattform
- Praktische Anwendung des Gelernten durch Konstruktion eines neuen oder Erweiterung eines vorhandenen Systems
- Kooperatives Lernen und Arbeiten in einer Learning Community

Das Wechselspiel zwischen erkundenden, dekonstruktiven Phasen und entwickelnden, konstruktiven Phasen ist in Abbildung 3 genauer dargestellt. (Siehe dazu auch [Mag03c].)

2.3 Aufbau

Die Struktur der Lerneinheiten wurde bereits im Kapitel *Was soll gelehrt werden* beschrieben. Die Strukturierung der Materialien selbst richtet sich bei der Arbeit an den Fallstudien nun nach den jeweiligen Erkundungs- und Entwicklungsaufträgen sowie nach den jeweiligen didaktischen Fenstern. Grundsätzlich ist es aber so, dass sämtliche Materialien, die zum Seminar gehören, den Studierenden in Form einer Materialsammlung über die Lernplattform (momentan *Steam*) angeboten werden. Dies schließt auch Materialien (beispielsweise zusätzliche Videos oder Animationen) ein, die nicht für die Bearbeitung der jeweiligen Aufgabe notwendig sind, sondern nur unterstützend wirken, bzw. unterschiedliche Lernertypen berücksichtigen. Die Studierenden können sich dann eine individuelle Kollektion angereichert mit eigenen Materialien als Grundlage ihrer weiteren Arbeit erstellen. Wie wirksam die einzelnen Materialien sind und auf welchen Ebenen der Darstellung am wirksamsten der Transfer von Wissen und Methodik bzw. bestimmter Sachverhalte stattfindet, müsste aber Gegenstand einer weitergehenden Untersuchung sein.

Im Folgenden werden nun einzelne Materialien zu Erkundungsaufträgen und didaktischen Fenstern beschrieben.

2.3.1 Lerneinheit 1.3

Die multimedialen Bausteine bzw. Lernobjekte der Lerneinheit werden im Wesentlichen durch die Fallstudie des Hochregallagers bestimmt. Diese wurden so aufbereitet, dass sie das Wechselspiel von Dekonstruktion und Konstruktion widerspiegeln, wie in Abbildung 3 beschrieben. Der Teilaspekt Dekonstruktion wird durch das in der Arbeitsgruppe implementierte LEGO-Hochregallager (Abbildung 1) realisiert, der Aspekt der Konstruktion durch eine durch RCX[4]-Bausteine gesteuerte automatische Kommissionierstation. Für diesen Teil des Seminars werden von uns aber nur die grundlegende LEGO-Hardware und Videos des Vorbildes bereitgestellt. Alles andere soll von den Studierenden selbst implementiert werden.

[4] Robotic Command Explorer – Der programmierbare *Legostein*

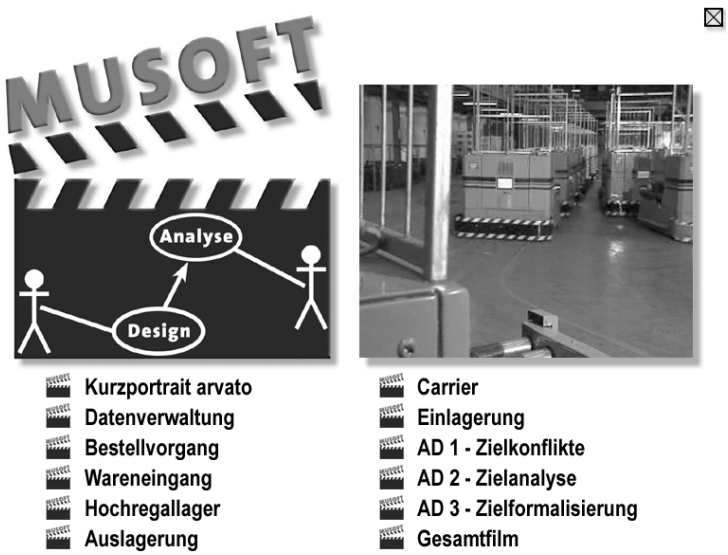


Abbildung 4: Videos des arvato Hochregallagers

Ausgangspunkt für die Fallstudie ist das Hochregallager von Bertelsmann arvato in Gütersloh. Aus diesem Lager stehen Videos bereit, die zum einen Anwendungsfälle im realen Lager zeigen und helfen Modellbildungsaspekte zu thematisieren, zum anderen lassen sich daraus Aufgaben generieren - sowohl für Erkundungsaufträge mit kleinen Reengineeringaufgaben als auch für die Entwicklung der Kommissionierstation.

Das Modell des Hochregallagers in LEGO Mindstorms besteht aus zehn RCX-Bausteinen: zwei für die Regalbediengeräte, zwei für die beiden autonomen Transportfahrzeuge, vier für die Übergabestationen, einer für die Token-Vergabe der Infrarot-Kommunikation und ein weiterer für den Gabelstapler und die Auftragseingabe. Das Modell wird den Studierenden komplett mit Steuerungssoftware in Java präsentiert. Durch verschiedene Erkundungsaufträge mit kleinen Reengineeringaufgaben sollen die Seminar-Teilnehmer die Software und ihre Entwurfsprinzipien kennen lernen und dann in einem weiteren Schritt das Gelernte auf einen neuen Gegenstand, nämlich die automatische Kommissionierstation, übertragen. Bei der eigenen Entwicklung kommt dann auch methodisches Wissen über Softwareentwicklungsprozesse zum Tragen.

Für die ersten Erkundungsaufträge steht den Studierenden die Steuerungssoftware noch nicht zur Verfügung. Nur Videos des arvato-Lagers, das LEGO-Lager auf phänomenologischer Ebene und verschiedene mehr oder weniger abstrakte Flash-Animationen zu Regalbediengerät und autonomen Transportfahrzeug werden den Studierenden angeboten. Für den Fall, dass den Studierenden in zukünftigen Seminaren das LEGO-Lager nicht physisch zur Verfügung steht, sind von allen Abläufen Fotos und Videos (Abbildung 6) im Material-Portfolio vorhanden.

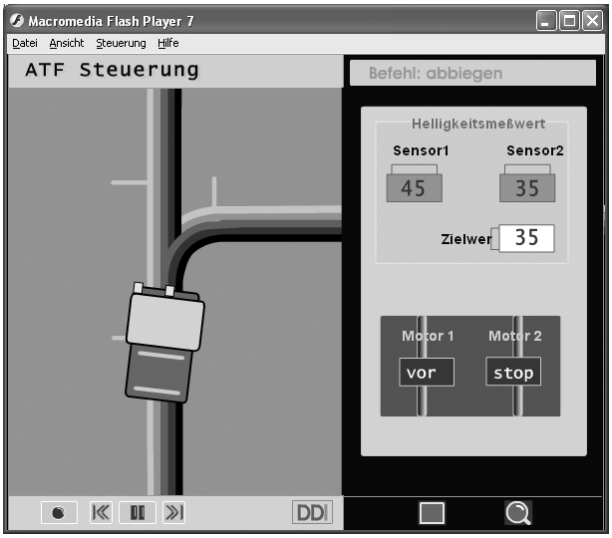


Abbildung 5: Flash Animation des autonomen Transportfahrzeuges

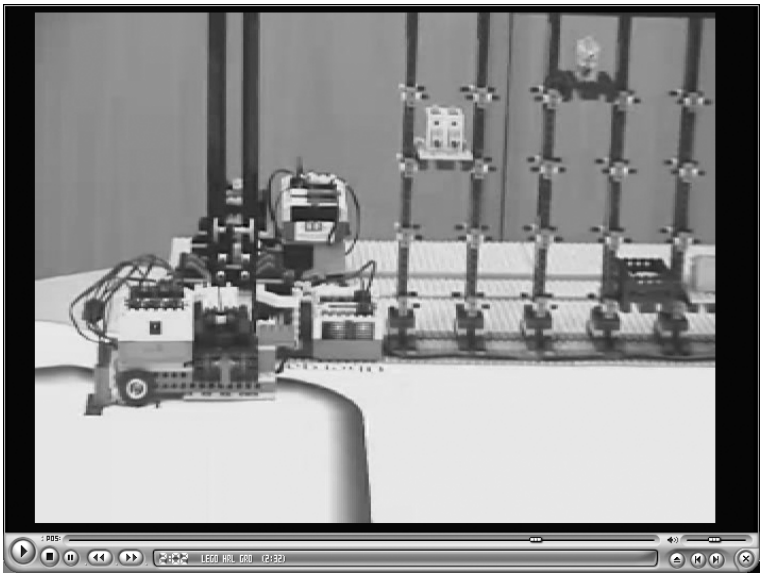


Abbildung 6: Videos des LEGO Hochregallagers

Die Seminar-Teilnehmer sollen sich nun zunächst auf Entwurfsebene Gedanken über eine mögliche Steuerung des LEGO-Lagers Gedanken machen. Im weiteren wird dies bis hin zu eigenen Klassendiagrammen weitergeführt. Diese werden dann mit der existierenden Hochregallagersteuerung in Java, veranschaulicht durch Klassen- und Sequenzdiagramme, verglichen. Dabei können verschiedene Designentscheidungen und Qualitätskriterien diskutiert werden.

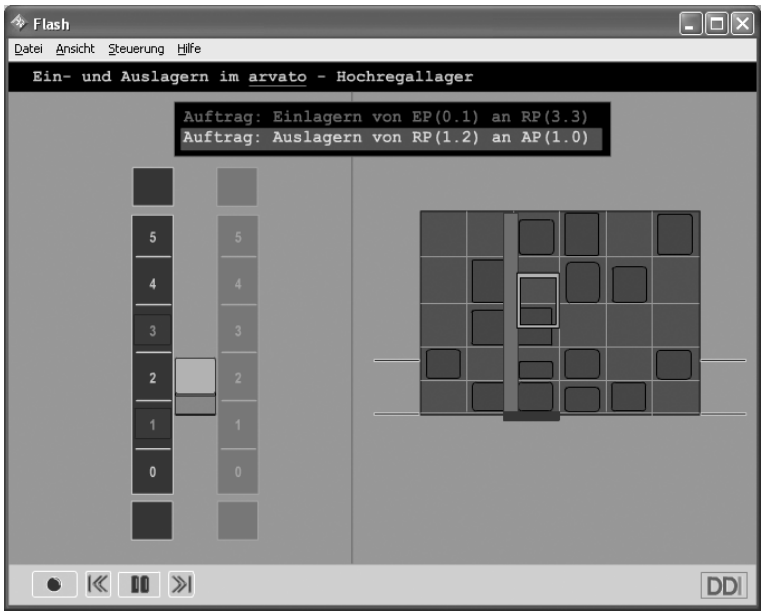


Abbildung 7: Flash Animation eines Auslagerungsvorganges bei arvato

Die anschließenden Erkundungsaufträge widmen sich der Exploration des Java-Quelltextes für die einzelnen RCX-Einheiten. Durch kleine Reengineering-Aufträge erhalten die Studierenden einen situierten Anlass, sich mit dem Quelltext näher zu beschäftigen. Die Aufträge werden dabei mit Hilfe der Videos des arvato-Lagers generiert, unterstützt von Flash-Animationen über einzelne Abläufe in diesem Lager (Abbildung 7). Hierbei ist zunächst nur der Quelltext einzelner RCX-Einheiten betroffen, nicht aber die Kommunikation zwischen den Einheiten. Aufträge können hier die Veränderung des Regalbediengerätes oder des autonomen Transportfahrzeuges sein. Dies schließt kleine Änderungen der LEGO-Hardware ein, zum Beispiel der Sensorenarten. Da die Studierenden auch zu hause die Möglichkeit zur Lösung von Programmieraufgaben haben sollen, wird ihnen hierfür eine von uns entwickelte Simulationsumgebung (Abbildung 8) angeboten, auf der exakt der gleiche Java-Code lauffähig ist, wie auf den (realen) RCX-Einheiten.

Die weiteren Arbeitsaufträge beschäftigen sich dann mit der Kommunikation. Die Kommunikation über die Infrarot-Schnittstelle der RCX-Bausteine soll erkundet und verändert werden. Hierfür stehen Animationen zum Datenaustausch im LEGO-Hochregallager (Abbildung

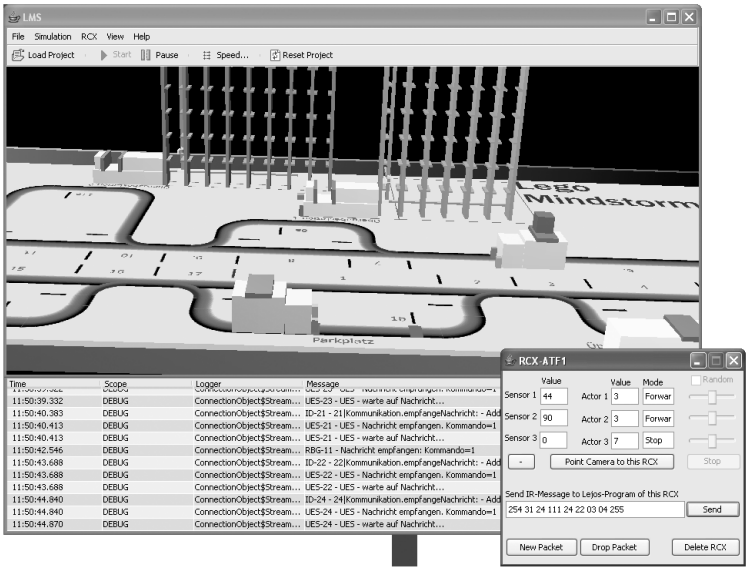


Abbildung 8: LEGO Mindstorms Simulator

9) zur Verfügung, aber auch Hintergrundinformationen zu den Kommunikationsklassen der leJOS[5]-Bibliothek (mit der die Java-Programmierung der RCX-Bausteine erst möglich wird) und Animationen zu verschiedenen allgemeinen Kommunikationsmodellen (die nur zum Teil auch mit RCX-Bausteinen verwirklicht werden können).

Damit ist die Exploration des vorgefertigten Informatiksystems beendet. Die Studierenden sollten dabei die Fähigkeiten erworben haben, die zum Bau eines eigenen Systems aus RCX-Bausteinen notwendig sind. Diese Entwicklung soll in Form einer virtuellen Firma geschehen, die den Auftrag zur Erstellung einer automatischen Kommissionierstation erhält. Eine solche Kommissionierstation (allerdings von Menschen bedient) kommt auch in den Videos des arvato-Lagers vor.

Eine mit MLCad[6] nachgebildete Version des physischen Teils der studentischen Lösung aus dem Sommersemester 2003 ist in Abbildung 10 zu sehen.

2.3.2 Lerneinheit 2.4

Das didaktische Konzept und die Materialien zur Lerneinheit 2.4 sind äquivalent zu denen in der Lerneinheit 1.3. Fallstudie ist hier eine Schulkiosk-Software, die in der Entwicklungsumgebung Fujaba[7] realisiert wurde. Die Wahl fiel hier im Gegensatz zur Lerneinheit 1.3 auf diese Entwicklungsumgebung, da hier die Programmentwicklung mit Hilfe von grafischen Notatio-

[5] http://lejos.sourceforge.net/
[6] http://www.lm-software.com/mlcad/
[7] http://www.fujaba.de/

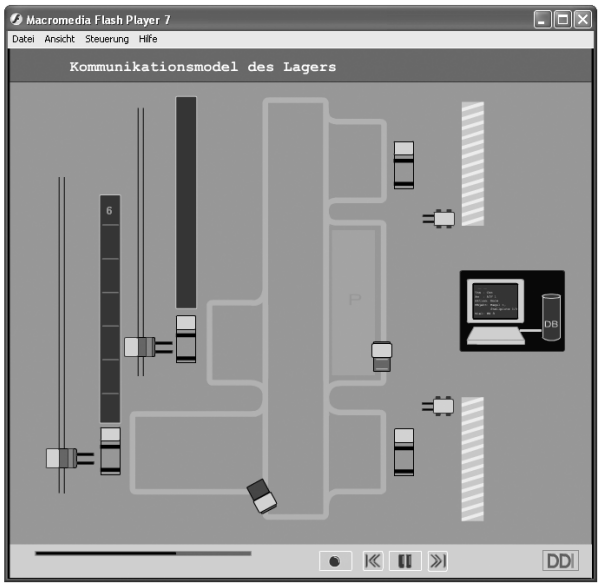


Abbildung 9: Flash Animation der Kommunikation im LEGO Hochregallager

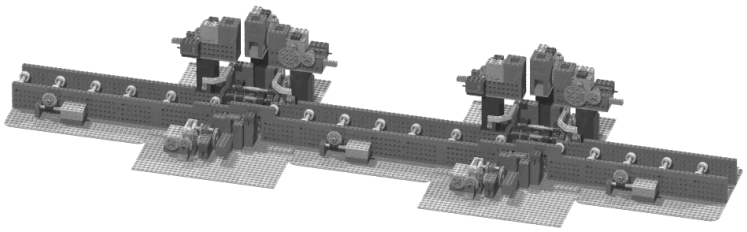


Abbildung 10: Die automatische LEGO Kommissionierstation

nen möglich ist und somit der oft als schwierig bezeichnete Bruch zwischen den Notationen in der Entwurfs- und der Implementierungsphase von Programmen vermindert wird.[8]

Abbildung 11: Schulkiosk-Videos

Auch zu dieser Fallstudie stehen Videos mit Anwendungsfällen (Abbildung 11) eines realen Vorbildes zur Verfügung. Auch hier können diese Videos für die Generierung von Aufgaben verwendet werden. - Aber auch um die verschiedenen Ausbaustufen, in denen die Software existiert, zu begründen.

Die Software existiert in drei Ausbaustufen (1.0, 1.1, 1.2), an denen verschiedene Inhalte thematisiert werden können. Dies geschieht in so genannten *didaktischen Fenstern*. An der Schulkiosk-Software lassen sich grundlegende Konzepte der objektorientierten Programmierung (z.B. Assoziation, Aggregation) aber auch Entwurfsmuster (z.B. MVC, Observer-Pattern, Iterator) erörtern. Aus diesem Grund stehen für alle Ausbaustufen Klassendiagramme zur Verfügung (Abbildung 13).

Version 1.0 (Abbildung 12) ist der Ausgangspunkt für eigene Erweiterungen der Studierenden. Sie besteht nur aus einem rudimentären Kassenmodul und einer einfachen Lagerverwaltung. Das didaktische Fenster hätte hier im Sinne der Dekonstruktion die Aufgabe von der Funktionalität und der Oberfläche der didaktischen Software auf das dahinter liegende Modell der (Fach-)Klassen zu schließen. Die unpraktische Lösung für das Kassenmodul eröffnet ein didaktisches Fenster zum Thema *Softwareergonomie* und ISO-Richtlinien.

[8] Der Einsatz dieses Tools für die Programmierung von LEGO-RCX-Bausteinen war im Rahmen des MuSofT-Projektes nicht möglich, wird aber momentan von der Arbeitsgruppe erprobt.

Abbildung 12: Oberfläche des Schulkiosk v1.0

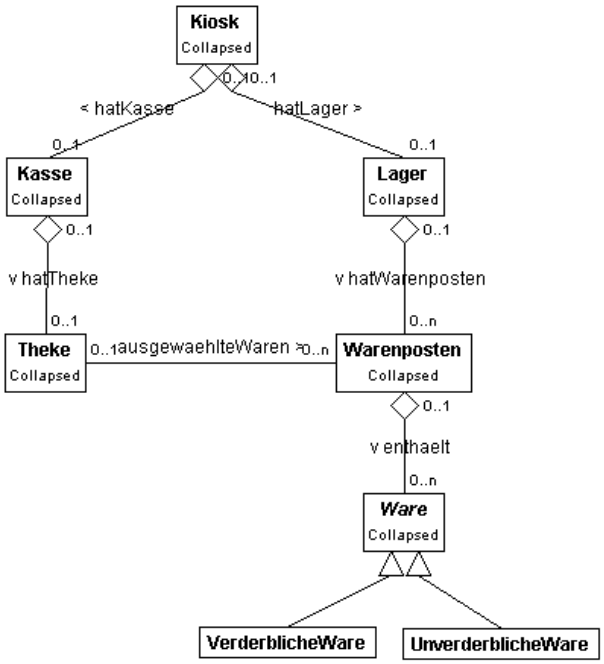


Abbildung 13: Zentrale Klassen des Schulkiosk v1.0

Version 1.1 verfügt über eine erweiterte Kasse, die auch das Stornieren und rückgängig Machen ganzer Verkaufsvorgänge gestattet. Die Oberfläche besitzt nun Schnelltasten. Auch Wechselgeld kann berechnet werden. Zusätzlich hat diese Version eine Benutzerverwaltung, und Verkaufsvorgänge werden nun aufgezeichnet. Thematisieren lässt sich hier die Softwareentwicklung als kooperativer Kommunikationsprozess. Verschiedene Interessen in der Entwurfsphase können zu unterschiedlichen Entscheidungen führen. Die Studierende nehmen verschiedene Rollen im Entwicklungsprozess ein. So müssen zum Beispiel Interessen der Mitarbeiter des Kiosk bzgl. Daten- und Persönlichkeitsschutzes mit Interessen der Kiosk-Besitzer bzgl. Abrechnung, Überwachung und Leistungserhebung abgewogen werden. Hier spielen soziale und rechtliche Fragen eine Rolle, die mit entsprechenden Hintergrundmaterialien (z.B. Datenschutzgesetz) thematisiert werden.

In Version 1.2 (Abbildung 2) wird dann zusätzlich ein Statistikmodul eingeführt. Hier werden Verkaufsvorgänge nach Benutzern getrennt detailliert aufgezeichnet. Dies gestattet auch die Abrechnung der Tagesumsätze des Kiosk.

Darüber hinaus bietet die Schulkiosk-Software auch die Möglichkeit, Konzepte wie Ereignis- und Ausnahmebehandlung und (Sortier-)Algorithmen im Zusammenhang mit Schulunterricht zu behandeln.

2.4 Anforderungen an die Lehr/Lernumgebung und technische Realisierung

Die Materialien zu den Lerneinheiten 1.3 und 2.4 werden von der AG Didaktik der Informatik an der Universität Paderborn momentan in Seminaren auf einem Steam-Server angeboten. Diese Art der Web gestützten Präsentation erfüllt schon die meisten Anforderungen, die unsere Arbeitsgruppe an die Lehr/Lernumgebung stellt: Benutzerverwaltung, Zugriffsrechte, Möglichkeit für die Studierenden eigene Ergebnisse zu präsentieren, synchrone und asynchrone Kooperation über Email und Chat, Unterstützung vielfältiger Datenformate. Allerdings können keine Workflows definiert werden, es fehlen teilweise Möglichkeiten, Materialien geeignet zu strukturieren und die Möglichkeit für die Studierenden, individuelle Sichtweisen auf die Materialien zu definieren. Da die von uns durchgeführten Seminare relativ klein sind, ist eine weitere Unterstützung von Community-Features durch die Plattform nicht notwendig. Aufgrund der Präsentation über WWW bzw. Webbrowser sind wir auf für das WWW-geeignete Datenformate angewiesen, was aber nicht zu besonderen Einschränkungen geführt hat. Die von uns verwendeten Datenformate sind alle mit Hilfe von kostenlosen Programmen nutzbar: Java, HTML, PDF, Flash und MPEG. Die Entwicklungsumgebung Fujaba, die in der Lerneinheit 2.4 genutzt wird, steht ebenfalls kostenfrei zur Verfügung. In Einheit 1.3 wird das Together Control Center[9] genutzt, das zumindest an Universitäten meist kostenlos zur Verfügung steht. Für diese Umgebung wurde von uns mit dem *Together MindstormsTool* eine frei verfügbare Erweiterung entwickelt, die die Programmierung von LEGO-RCX-Bausteinen ohne zusätzliche Tools ermöglicht. Das MindstormsTool selbst wurde in Java entwickelt und als Modul in das Together Control Center integriert. Das Tool verfügt über eine Schnittstelle, die auch eine

[9] http://www.togethersoft.de/ - Die Entwicklungsumgebung stellt UML-Notationen zur Verfügung und sollte nach Projekt interner Absprache hauptsächlich genutzt werden.

Integration in andere Entwicklungsumgebungen ermöglicht. Es ist unter der MuSofT-Lizenz (siehe den Bericht des Koordinationsteams *Nachhaltigkeit* veröffentlicht worden. [10]

Als weiteres wichtiges Werkzeug ist im Zusammenhang mit dem MuSofT-Projekt die oben schon angesprochene Simulationsumgebung für LEGO-Mindstorms entstanden. (Abbildung 8) Die Umgebung ermöglicht, dass Studierende auch in der Nacharbeitung von Seminaren oder anderen Veranstaltungen Programme für den RCX-Baustein erproben können. Die Umgebung wurde in Java realisiert und ist sowohl unter Windows- wie unter Unix-Systemen lauffähig. Sie besteht aus zwei Komponenten:

1. *Simulationsmaschine*: Die Simulationsmaschine bietet eine virtuelle dreidimensionale Welt an, in der sich LEGO-Roboter bewegen und interagieren können. Sie wurde mit Hilfe von Java 3D[11] realisiert und gestattet das Laden verschiedener Szenarien, Ansichten aus verschiedenen Kameraperspektiven und die Detailbetrachtung durch Zoom.

2. *Lejos-Emulation*: Die Lejos-Emulation gestattet die Emulation von Java/leJOS-Programmen, die für den realen RCX-Baustein geschrieben wurden, auf einem virtuellen Baustein. Die Programme müssen dafür nicht verändert werden und können über die Simulationsumgebung auf einzelne im Netz verteilte Controller geladen werden. Die Kommunikation zwischen Controllern und Simulationsmaschine läuft über RMI[12].

Die Konfiguration der Simulation bzw. von verschiedenen Szenarien, wie zum Beispiel dem LEGO-Hochrregallager ist über XML-Dateien möglich. Bisher wurden verschiedene Szenarien realisiert, die im engen Zusammenhang mit der Fallstudie Hochregallager stehen. Die Software ist ebenfalls unter die MuSofT-Lizenz gestellt worden und somit frei verfügbar[13].

3 Erfahrungen

Eine zentrale Erfahrung aus dem MuSofT-Projekt ist, dass die Entwicklung von durchgängigen Fallstudien mit multimedialen Elementen sehr zeitaufwändig ist. Insbesondere seien hier die Entwicklung von Animationen in Flash (vor allem die interaktiven) und der Dreh von Videos unter Einbeziehung externer Partner genannt. Kritisch muss man sogar feststellen, dass bei den Videos, so schön sie auch geworden sind, die Kosten vermutlich in keinem adäquaten Verhältnis zum Nutzen stehen. (Obwohl der Nutzen auf Seiten der Studierenden sicher schwer messbar ist.)

Auch die Zusammenarbeit unter den verschiedenen Arbeitsgruppen hätte intensiver sein können, war aber wegen der sehr unterschiedlichen Schwerpunkte in der jeweiligen Gruppe mit unterschiedlichen Studenten und unterschiedlichen Themen nicht besser zu erwarten. Weitere Lerneinheit übergreifende Fallstudien hätten hier eventuell einen positiven Effekt gebracht. (Wie das Beispiel Hochregallager zeigt.)

Da durch die Materialien hauptsächlich Präsenzlehre unterstützt werden soll, sind diese in Bezug auf das Gesamtangebot fragmentarisch und nur im jeweiligen Kontext gültig. Daraus

[10] http://ddi.uni-paderborn.de/tmt
[11] http://java.sun.com/
[12] Remote Methode Invocation
[13] http://www.lms-project.org/

ergibt sich, dass deren Präsentation über das MuSofT-Portal insbesondere für die Nutzung durch Studierende nur eingeschränkt sinnvoll erscheint. Am ehesten ist nach unserer Meinung die Wiederverwendbarkeit der multimedialen Materialien zu den Fallstudien möglich.

4 Evaluierung

Im Sommersemester 2003 wurden die Materialien zur Lerneinheit 1.3 *Softwareengineering in der Informatiklehrerausbildung* und zur Fallstudie *Hochregallager* im Seminar *Informatische Lernwerkstatt für die Sekundarstufe II* evaluiert. Die Materialien zur Lerneinheit 2.4 *Dekonstruktion von Softwaresystemen* werden momentan (im Wintersemester 2003/2004) innerhalb des Seminars *Lernwerkstatt Informatik* verwendet. Die Evaluation folgt zum Semesterende.

Die Evaluation der Lerneinheit 1.3 hatte folgende Bestandteile:

- Eingangsfragebogen
- ergänzende Gruppendiskussion
- Prozessbeobachtung
- Abschlussfragebogen
- Produktbewertung

Die Durchführung und die Ergebnisse der Teilbereiche werden im folgenden kurz zusammengefasst.

4.1 Wie wurde die LE evaluiert?

4.1.1 Eingangsfragebogen

Der Eingangsfragebogen beschäftigte sich primär mit Fragen zur Person, den Vorerfahrungen bzgl. Konzepten der Softwareentwicklung und Informatik Didaktik sowie mit multimedialen Lernmaterialien.

Acht Studierende des Informatik Lehramtes Sek. II haben am Seminar teilgenommen. Die Fachsemesterzahl lag im Durchschnitt bei 6-7. Es handelte sich um zwei Studentinnen und sechs Studenten. Ungefähr die Hälfte der Studierenden waren bereits mit multimedial aufbereiteten Lernmaterialien an der Universität in Berührung gekommen (ein Student in der Vorlesung *Techniken des Softwareentwurfs* auch schon mit Materialien des MuSofT-Teilprojektes Engels). Die anderen in der Regel mit Audioannotation an Vorlesungsfolien. Das didaktische Konzept der *Dekonstruktion von Informatiksystemen* war allen aus der Theorie bekannt, außer zwei Studierenden hatten sie aber keine praktischen Erfahrungen mit Lernmaterialien, die nach diesem Konzept entworfen wurden. Bis auf einen Studenten kannte niemand das Konzept des *Blended Learning*, das im Seminar eingesetzt und reflektiert werden sollte. Drei Studierende hatten aus anderen Seminaren schon praktische Erfahrung mit LEGO Mindstorms, die anderen Studierenden hatten nur in Kindertagen LEGO-Modelle gebaut.

Abbildung 14: Seminarteilnehmer beim Bauen der Modelle

Obwohl es sich um Studierende des Hauptstudiums handelte, wussten drei Personen in der Eingangsbefragung nichts mit den Begriffen *Vorgehensmodell* und *Entwurfsmuster* anzufangen, da es im Seminar ja darum gehen sollte, ein solches Vorgehensmodell für den Informatikunterricht vorzustellen und weiterzuentwickeln. Die übrigen kannten aber verschiedene Vorgehensmodelle und konnten Beispiele für einfache Entwurfsmuster nennen.

Nach den Vorerfahrungen wurden die Erwartungen an das Seminar und Einschätzungen abgefragt. Fünf Studierende erwarteten vor allem, mit LEGO Mindstorms einen spielerischen und motivierenden Zugang zum Informatikunterricht der Sekundarstufe I kennen zu lernen. Allerdings wurde auch hier schon die Befürchtung geäußert, dass der Zugang eher für die Sekundarstufe II geeignet wäre. (Tatsächlich waren die im Seminar eingesetzten Materialien auch für den mittelbaren Einsatz in dieser Stufe gedacht.) In der Sekundarstufe II sahen die Studierenden die Möglichkeit, durch LEGO Mindstorms auf praxisnahe Weise komplexe(re) Problemstellungen einzuführen. Aber auch dort wurde von zwei Studierenden die mögliche höhere Motivation von Schülern hervorgehoben.

Die Einstellungen der Studierenden bzgl. multimedialer Studienmaterialien waren vor dem Seminar ambivalent. Chancen, wie die zum fächerübergreifenden Arbeiten und zur Veranschaulichung theoretischer Konzepte, standen in den Bemerkungen Forderungen nach genauer Abstimmung der Materialien und genauen Lernzielen gegenüber. Die Eignung des Beispiels Hochregallager sahen die Studenten für die Sekundarstufe II und die Informatiklehrerausbildung, nicht aber für die Sekundarstufe I.

4.1.2 Prozessbeobachtung

Unter dem Begriff *Prozessbeobachtung* werden hier sowohl die unsystematischen Beobachtungen während des Seminars als auch die (vereinzelte) Aufzeichnung von Bildschirmvideos während Rechnerarbeitsphasen im Seminar zusammengefasst. Die Beobachtungen während des Seminars zeigten eine ungewöhnlich motivierte Seminargruppe. Selten haben wir bisher mit Gruppen zu tun gehabt, die so viel Zeit außerhalb des Seminars in die Vorbereitungen investiert haben. Die Beobachtungen zeigten aber auch, dass die Teilnehmer noch relativ wenig praktische Erfahrungen mit objektorientierter Modellierung gemacht hatten. So neigten sie bei der Entwicklung von Klassendiagrammen zu Superklassen mit maximaler Funktionalität.

Die Auswertung der Bildschirmvideos ergab, dass die Studierenden sehrwohl bei konkreten Fragestellungen während des Entwurfs oder der Implementierung die angebotenen Flash-Animationen nutzten. Dies betraf auch diejenigen (z.B. Regalbediengerät), die im Abschlussfragebogen als (wenn das reale Modell zur Verfügung steht) überflüssig bezeichnet wurden.

4.1.3 Abschlussfragebogen

Der Abschlussfragebogen hatte nicht primär zum Ziel, die Eignung einzelner (multimedialer) Materialien zu evaluieren, sondern diente dem Vergleich von Erwartungen und Erfahrungen der Studierenden und der Evaluation des verwendeten didaktischen Ansatzes der Dekonstruktion sowie der LEGO Mindstorms Roboter. Die Eignung der Materialien wurde nur im Zusammenhang mit diesem Ansatz überprüft. An der Abschlussbefragung haben ebenfalls acht Studierende teilgenommen.

Die ersten Fragen des Abschlussbogens widmeten sich der Fragestellung, ob die Studierenden LEGO Mindstorms auch weiterhin für den Informatik-Unterricht an Schulen und die Informatiklehrerausbildung für geeignet halten. Für die Sekundarstufe I hielten alle Teilnehmer Mindstorms als Zugang zur Informatik für geeignet, setzten aber eine deutlich verminderte Komplexität der Beispiele voraus. Auch für die Sekundarstufe II sahen noch zwei Studierende die Gefahr der zu hohen Komplexität, betonten aber den motivierenden und fachlich geeigneten Zugang zu Themen der Informatik, besonders der Modellierung und Programmierung. Für die Informatiklehrerausbildung sahen sie die uneingeschränkte Eignung und betonten die höhere Motivation und die Abwechslung zu üblichen Seminarthemen.

Die im Seminar eingesetzte Fallstudie *Hochregallager* (zusammen mit dem Entwicklungsauftrag *Kommissionierstation*) schätzten alle Studierenden für die Sekundarstufe I als zu komplex ein. Nur zwei Teilnehmer wiederholten dieses Urteil für die Sekundarstufe II, die übrigen hielten die Fallstudie für einsetzbar, wenn die Rahmenbedingungen stimmen. Alle Teilnehmer meinten aber, dass das Beispiel für die Informatiklehrerausbildung an der Universität gut geeignet sei.

Der nächste Fragenkomplex widmete sich den Lernschwierigkeiten der Studierenden. Hier wurde von den Teilnehmern angemerkt, dass die Übertragung von Konzepten der Hochregallagersteuerung auf die Kommissionierstation gut möglich war, wenn der Sachverhalt verstanden war. Als Schwierigkeit wurde aber das Bauen des LEGO-Modells angemerkt (drei Teilnehmer), die Umsetzung der Kommunikation (zwei Teilnehmer) und das Debugging der Java-Programme auf dem RCX.

Als Verbesserungsvorschläge nannten die Studierenden im Anschluss vor allem die bes-

Abbildung 15: Seminarteilnehmer beim Programmieren der Roboter

sere (online) Dokumentation von Ergebnissen und ein kleinschrittigeres Vorgehen mit mehr Reflektionsphasen.

Die nächsten Fragen handelten von einzelnen bereitgestellten Materialien bzw. Materialarten. Die Methode der Erkundungsaufträge zum Hochregallagersystem bewerteten alle Studierenden als gut. Der Einsatz von CRC-Karten und Objektspiel wurde aber von einer Person bemängelt, da das Konzept unklar geblieben wäre. Die eingesetzten Animationen *autonomes Transportfahrzeug, Regalbediengerät und Kommunikationsmodelle* wurden von mehr als der Hälfte als hilfreich bezeichnet, zwei Teilnehmer bemerkten, dass sie mit Ausnahme der Kommunikation keine über das reale Modell hinausgehenden Erkenntnisse erbracht hätten. Den bereitgestellten Quellcode des Hochregallagers empfanden alle Teilnehmer als hilfreiche Vorlage für den eigenen Code. Alle Teilnehmer empfanden den Umfang informatischer Problemstellungen im Seminar als angemessen, lediglich zwei Personen merkten die zu lange Bauphase der LEGO-Modelle negativ an. Den Entwicklungsauftrag *Kommissionierstation* bewerteten alle Teilnehmer als angemessene Aufforderung zum Transfer von methodischem und deklarativem Wissen.

Die letzte Frage des Abschlussfragebogen befasste sich mit dem methodischen Ablauf des Seminars. Die Struktur der Veranstaltung, die das Konzept der Dekonstruktion von Informatiksystemen nicht nur inhaltlich auf einer Metaebene behandelte, sondern dieses auch selbst einsetzte, wurde von den Studierenden durchweg positiv gewertet. Bemängelt wurde der zu große Zeitraum, der für den Bau der LEGO-Modelle notwendig war. (Aber von den Veranstaltern im voraus billigend in Kauf genommen wurde.)

4.1.4 Produktbewertung

Wie zuvor schon angesprochen, wurde in der Startphase des Seminars anhand der ersten Klassendiagramme deutlich, dass die Studierenden über wenig praktische Erfahrungen im Entwurf objektorientierter Programme verfügten. Sie waren aber trotzdem in der Lage, alle gestellten Aufgaben zur Zufriedenheit zu lösen. Bei den anschließenden Reflektionsphasen wurden dann Bewertungskriterien für die Produkte deutlich gemacht, so dass die damit in eigener Erfahrung situiert erworbenen Kenntnisse zu einer Verbesserung der Kenntnisse geführt haben sollten. (Dies müsste aber näher untersucht werden.) Einzig der abschließende Entwicklungsauftrag wurde nicht vollständig beendet, da es Probleme mit der Kommunikation zwischen den RCX-Einheiten gab. Der Grund dafür ist aber nur zum Teil auf die Programme der Studenten zurückzuführen.

4.1.5 Anmerkung zur Evaluation

Abschließend sollte zur Evaluation der Materialien gesagt werden, dass die Anwendung der verschiedenen Untersuchungsinstrumente noch relativ unsystematisch geschehen ist. Insofern kann die Evaluation nur als Vorstudie für eine umfassendere systematische Untersuchung gelten. Trotzdem erlaubt sie uns bis zu einem gewissen Grad, die Eignung der eingesetzten Materialien zu beurteilen. Allerdings ist (eigentlich) die Übertragung auf andere Lerngruppen unzulässig, vor allem auch wegen der kleinen Teilnehmerzahl.

4.2 Welche Ergebnisse wurden damit nachgewiesen?

Zusammenfassend kann das Konzept der *Dekonstruktion von Informatiksystemen* unter Anwendung von Erkundungsaufträgen als geeigneter Zugang zu informatischen Fragestellungen bezeichnet werden. Ob dieser Zugang besser ist als andere, müsste aber durch eine weitergehende Studie untersucht werden. Die Materialien zur Fallstudie Hochregallager sind hinreichend, müssen aber im Detail noch verbessert werden. Vor allem fehlt noch eine geeignete Plattform zur Präsentation der multimedialen Materialien und Portfolios innerhalb des Seminars. Die Bauergebnisse der aktuellen Gruppe können für zukünftige Seminare genutzt werden, so dass der Kritikpunkt der ausgedehnten Bauphase beseitigt wäre. Die Erkundungsaufträge können aufgrund der Hinweise so weit verbessert werden, dass sie die jeweils am besten geeigneten Arbeitsmaterialien besonders hervorheben. Darüber hinaus sollte das Material-Portfolio zumindest noch um Verweise auf Grundlagen der Softwareentwicklung ergänzt werden, um die eventuell vorhandenen Schwächen auf diesem Gebiet aufzufangen.

Literatur

[ADE03a] ALFERT, KLAUS, ERNST-ERICH DOBERKAT und GREGOR ENGELS (Herausgeber): *Ergebnisbericht des Jahres 2002 des Projektes "MuSofT – Multimedia in der SoftwareTechnik"*, Band 133 der Reihe *Softwaretechnik-Memo*. Lehrstuhl für Software-Technologie, Fachbereich Informatik, Universität Dortmund, März 2003.

[ADE+03b] ALFERT, KLAUS, ERNST-ERICH DOBERKAT, GREGOR ENGELS, MARC LOHMANN, JOHANNES MAGENHEIM und ANDY SCHÜRR: *MuSofT – Multimedia in der SoftwareTechnik*. In: SIEDERSLEBEN, JOHANNES und DEBORA WEBER-WULFF (Herausgeber): *SEUH 8 Software Engineerung im Unterricht der Hochschulen Berlin 2003*, Seiten 70–80, Heidelberg, Februar 2003. dPunkt-Verlag.

[DE02] DOBERKAT, ERNST-ERICH und GREGOR ENGELS (Herausgeber): *Ergebnisbericht des Jahres 2001 des Projektes "MuSofT – Multimedia in der SoftwareTechnik"*, Band 121 der Reihe *Softwaretechnik-Memo*. Lehrstuhl für Software-Technologie, Fachbereich Informatik, Universität Dortmund, März 2002.

[JM03] JOHANNES MAGENHEIM, SIEGRID SCHUBERT: *Blended Learning im Informatikstudium*. In: *Informatik 2003, Innovative Informatikanwendungen, Bd. 2, Proceedings der 33. Jahrestagung der GI*, Seiten 73–79, Frankfurt a. M., 2003.

[Mag01] MAGENHEIM, JOHANNES: *Informatiksystem und Dekonstruktion als didaktische Kategorien - Theoretische Aspekte und unterrichtspraktische Implikationen einer systemorientierten Didaktik der Informatik*. In: *informatica didactica, Zeitschrift für fachdidaktische Grundlagen der Informatik*, Nummer 3 in *2001*, 2001.

[Mag03a] MAGENHEIM, JOHANNES: *Demands on Digital Media in an Informatics Learning Lab - Medial Aspects of an interactive Learning Environment for Software*

Engineering. In: *Proceedings of the The 7th World Multi-Conference on SYSTEMICS, CYBERNETICS AND INFORMATICS SCI 2003*, Orlando, 2003.

[Mag03b] MAGENHEIM, JOHANNES: *ILL: The Informatics Learning Laboratory - Connecting Learning Communities with Communities of Practice in a web based Learning Laboratory for Informatics*. In: *Proccedings of ED-MEDIA 2003, World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Honolulu, 2003.

[Mag03c] MAGENHEIM, JOHANNES: *Informatik Lernlabor – Systemorientierte Didaktik in der Praxis*. In: HUBWIESER, PETER (Herausgeber): *Informatische Fachkonzepte im Unterricht, Proceedings der infos2003, 10.GI-Fachtagung Informatik und Schule*, Seiten 13–31, Garching, 2003.

[Mag03d] MAGENHEIM, JOHANNES: *Social, affective and normative aspects of learning in ICT-enriched Teaching Environments*. In: *Proceedings of the TC3 IFIP-workshop 'ICT and the teacher of the future' Melbourne 2003*, Melbourne Australien, Januar 2003.

[Mag03e] MAGENHEIM, JOHANNES: *Wissensmanagement, Dekonstruktion und Learning Communities in der Softwaretechnik - Didaktische Konzepte im BMBF-Projekt MuSofT*. In: RINN, U. und J. WEDEKIND (Herausgeber): *Didaktik der neuen Medien*, Seiten 255–269, Münster, 2003. Waxmann.

Teilprojekt 2.1 – Software-Architektur

Jörg Pleumann

Teilprojekt 2.1 hat sich mit dem Thema „Software-Architektur“ beschäftigt. In seinem Rahmen sind zwei graphische Modellierungswerkzeuge für die strukturellen und dynamischen Aspekte von Softwaresysstemen entwickelt worden. Als Notation kommt jeweils eine Teilmenge der UML 2.0 zum Einsatz. Die Werkzeuge zeichnen sich durch einen an den Bedürfnissen der Lehre orientierten Funktionsumfang aus. Dazu zählt unter anderem die Möglichkeit, ein modelliertes System zu simulieren. Diese Simulation erlaubt Einblicke in das Verhalten des konkreten modellierten Systems und trägt gleichzeitig zu einem vertieften Verständnis der Laufzeitsemantik der Modellierungssprache bei. Durch eine multimediale Untermalung der Simulation wird eine Brücke zwischen abstrakten Modellen und realen Problemen geschlagen und zudem die Motivation der Studierenden zur Beschäftigung mit dem Formalismus erhöht. Kern beider Anwendungen ist ein Framework zur Entwicklung lehretauglicher Modellierungswerkzeuge, das ebenfalls innerhalb des Teilprojektes entstanden ist.

Inhaltsverzeichnis

1 Einleitung

Dieser Bericht beschäftigt sich mit dem MuSofT-Teilprojekt 2.1, das sich inhaltlich dem Thema „Software-Architektur“ gewidmet hat. Er umfaßt im wesentlichen das abschließende, dritte Projektjahr. Überlegungen und Ergebnisse der beiden ersten Jahre werden nur dort aufgegriffen, wo es für das Verständnis nötig ist. Für Details hierzu sei auf die früheren Ergebnisberichte [DE01, ADE02] verwiesen.

Primäres Ziel des Teilprojektes war die Entwicklung eines syntaxgesteuerten, graphischen Editors, der die Vermittlung von Grundlagen- und Erfahrungswissen zu Software-Architekturen durch praktische Arbeit unterstützt. Der Begriff einer Software-Architektur ist innerhalb der bestehenden Literatur nicht völlig eindeutig definiert. Es besteht jedoch in Standardwerken wie [GS94, BMR$^+$96] weitgehend Konsens darüber, daß die Architektur eines Softwaresystems wesentliche strukturelle und dynamische Eigenschaften des Systems in einer frühen Phase von dessen Entwicklung festlegt. Das Abstraktionsniveau einer Architektur liegt im allgemeinen relativ hoch. Man redet hier eher über die Verschaltung einzelner Komponenten zu einem System als über die konkreten Klassen, mit deren Hilfe diese Komponenten (bei einer objektorientierten Lösung) realisiert werden. Dieser Sichtweise, die auch dem entspricht, was in Dortmunder Softwaretechnik-Vorlesungen gelehrt wird, haben wir uns innerhalb des Teilprojektes angeschlossen.

Eine zentrale Entscheidung innerhalb des Teilprojektes war die Wahl der graphischen Notation, die das Werkzeug unterstützen sollte. Um die eingangs geschilderte Sichtweise von Architekturen zu reflektieren, wurde eine Notation benötigt, die strukturelle und dynamische Aspekte eines Softwaresystems angemessen berücksichtigt. Gleichzeitig sollte die Notation eine so präsize Laufzeitsemantik besitzen, daß sie eine Simulation von modellierten Systemen im Sinne eines schrittweisen „Durchspielens“ oder Testens erlaubt. Ziel dieser Simulation war, den aktuellen Systemzustand und den Kommunikationsfluß zwischen Komponenten beobachtbar zu machen und so zu einem vertieften Verständnis des Systemverhaltens wie auch der allgemeinen Semantik der Beschreibungssprache beizutragen.

Die Wahl fiel bereits in einer frühen Phase des Projektes auf eine Notation, die auf Real-Time Object-Oriented Modeling (ROOM) [SGW94] bzw. einer Variante der Unified Modeling Language (UML) [BRJ99] für Echtzeitsysteme [SR98, Lyo98] basiert. Wie sich bereits im ersten Projektjahr ankündigte, sollte diese Notation Bestandteil des UML-Standards werden, was mit der inzwischen verabschiedeten Version 2.0 der UML auch der Fall ist.

Bei der Planung und Realisierung der Lerneinheit wurde sehr bewußt Wert darauf gelegt, kein Werkzeug zu schaffen, das in der Leistung oder im Funktionsumfang mit professionellen Modellierungswerkzeugen konkurriert. Der Fokus lag vielmehr auf einem Werkzeug, das in vielerlei Hinsicht *leichtgewichtig* sein sollte. Dies umfaßt insbesondere eine Beschränkung des Funktionsumfangs auf das, was für die Lehre benötigt wird. Mit dieser Beschränkung einher geht eine Reduktion der Komplexität in der Benutzerschnittstelle, die für eine geringere Einarbeitungszeit der Studierenden sorgt und damit einen effizienteren Einsatz als Hilfsmittel in Übungen erlaubt. Ein Nebeneffekt des reduzierten Funktionsumfangs ist ein geringerer Ressourcenbedarf der gesamten Anwendung, der dafür sorgt, daß sie auch auf technisch einfacher ausgestatteten Rechnern zügig arbeitet.

Der weitere Bericht ist wie folgt gegliedert: Abschnitt 2 stellt das Werkzeug DAVE vor, das zunächst nur als Prototyp geplant war, aber aus einer Reihe von Gründen in den Rang eines eigenständigen Werkzeugs erhoben wurde und damit ein zusätzliches Ergebnis des Teilprojektes darstellt. In Abschnitt 3 folgt dann mit dem Werkzeug SAM eine Vorstellung der zentralen Ergebnisse des Teilprojektes. Die Reihenfolge ergibt sich aus der Tatsache, daß SAM inhaltlich wie technisch eine Obermenge von DAVE darstellt. Abschnitt 4 beleuchtet die technische Realisierung beider Anwendungen. Abschnitt 5 beschreibt ausführlich einen evaluierten Einsatz, der im Sommersemester 2003 durchgeführt wurde. Der abschließende Abschnitt 6 gibt eine kurze Zusammenfassung des Berichts.

2 Vorstellung der Lerneinheit DAVE

Das Werkzeug DAVE war anfangs nur als kleinerer Prototyp des Architektur-Werkzeugs SAM geplant. Um inhaltlich und technisch eine abgeschlossene Einheit zu bilden, die frühzeitig sinnvoll eingesetzt und evaluiert werden kann, wurde DAVE auf die dynamischen Anteile der zuvor beschriebenen Notation beschränkt, also auf Zustandsdiagramme [Har87, HN96]. Daraus leitet sich auch der Name DAVE (Dortmunder Automatenvisualisierer und -editor) ab[1].

Zustandsdiagramme werden in der UML nicht nur zur Beschreibung dynamischer Anteile von Architekturen verwendet. Sie können auch das diskrete Verhalten der Instanzen einer Klasse oder die erlaubten Aufrufreihenfolgen der Methoden einer Schnittstelle spezifizieren. Im Bereich eingebetteter Systeme, wo Aktualisierungen oder Korrekturen der Software nach dem Ausliefern des Systems nur schwer möglich (weil sich die Software in einem nur lesbaren Speicher befindet) oder sehr teuer sind (weil damit eine Rückrufaktionen verbunden ist), dienen Zustandsdiagramme zur detaillierten Spezifikation des Systemverhaltens. Ihre formale Semantik erlaubt die Analyse des Systems mittels Simulationen oder Model Checking. Durch

[1] Die namentliche Verwandtschaft mit David Harel, dem Erfinder der „klassischen“ Zustandsdiagramme, ist dabei durchaus beabsichtigt.

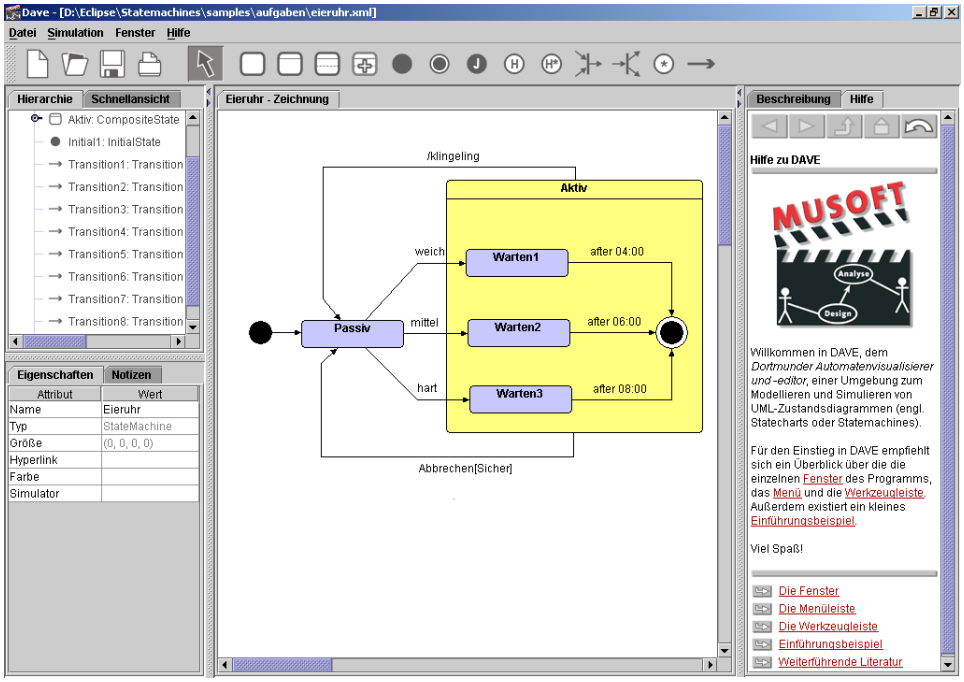


Abbildung 1: Bearbeiten eines Zustandsdiagramms mit DAVE

ihre vielfältigen Einsatzmöglichkeiten sind Zustandsdiagramme Teil praktisch aller Vorlesungen, die sich mit Softwaretechnik im allgemeinen oder mit UML im besonderen beschäftigen.

Aufgrund der großen Bedeutung von Zustandsdiagrammen und da DAVE bereits in der Frühphase seiner Entstehung eine gewisse Eigendynamik entwickelt hatte, haben wir uns entschlossen, das Werkzeug als eigenständiges Produkt zusätzlich zu dem eigentlich geplanten Werkzeug SAM zu pflegen und weiterzuentwickeln. Die Evaluation des ersten Einsatzes von DAVE (siehe Abschnitt 5) hat diese Entscheidung bestätigt. Auch aus softwaretechnischer Sicht macht sie Sinn: Da SAM technisch und inhaltlich eine Obermenge von DAVE darstellt, wird die Komplexität und damit der Wartungsaufwand auf zwei überschaubare Produkte verteilt.

2.1 Modellieren von Zustandsdiagrammen

Die primäre Funktionalität von DAVE ist das Modellieren von UML-Zustandsdiagrammen. Die erstellten Diagramme können gedruckt, gespeichert und zu einem späteren Zeitpunkt wieder geladen werden. Abbildung 1 zeigt ein Bildschirmfoto von DAVE. In der Mitte des Fensters befindet sich der Arbeitsbereich mit dem erstellten Modell – in diesem Fall ein einfaches Beispielmodell einer elektronischen Eieruhr, das im Werkzeug auch als Einführungsbeispiel enthalten ist. Am linken Rand befinden sich verschiedene Elemente zur Navigation innerhalb

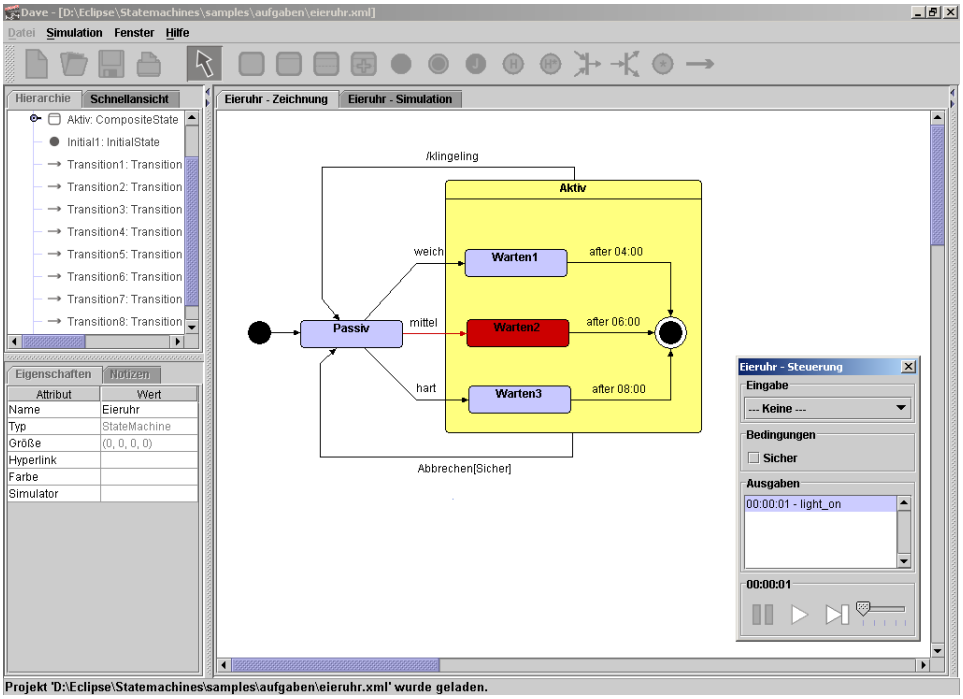


Abbildung 2: Simulieren eines Zustandsdiagramms mit DAVE

des Modells und zum Bearbeiten der Eigenschaften einzelner Elemente. Am rechten Rand befindet sich ein Hypertextbereich, in welchem die Online-Hilfe des Programms, Annotationen zu Modellen oder sonstiger Lehrstoff angezeigt werden.

2.2 Simulieren von Zustandsdiagrammen

Um eine Brücke zwischen Syntax und Semantik von Zustandsdiagrammen zu schlagen und den Studierenden einen besseren Eindruck in deren Laufzeitverhalten zu ermöglichen, unterstützt DAVE die Simulation des erstellten Modells. Basis für diese Simulation ist die in der UML-Spezifikation [Obj03] enthaltene Semantikbeschreibung für Zustandsdiagramme. Zur Kontrolle der Simulation erscheint ein zweites, kleineres Fenster, dessen unterer Teil der Steuerung einer Stereoanlage ähnelt (siehe Abbildung 2). Die einzelnen Schalter dienen zum Pausieren der Simulation, zum erneuten Starten oder zum Ausführen eines einzelnen Schrittes. Zudem kann die Geschwindigkeit der Simulation festgelegt werden. Für jeden Schritt der Simulation kann im oberen Teil des Fensters ein Ereignis ausgewählt werden, das vom Zustandsdiagramm verarbeitet werden soll. Die beiden Abschnitte in der Mitte des Fensters dienen zur Kontrolle der Überwachungsbedingungen und zur Anzeige von Ausgabeereignissen, die während der Simulation erzeugt werden. Aktive Zustände und schaltende Transitionen werden

während der Simulation im Zustandsdiagramm farblich hervorgehoben. Das Diagramm selbst wird während der Simulation in einen nur lesbaren Modus versetzt, damit keine Inkonsistenzen entstehen können.

2.3 Multimediale Untermalung der Simulation

Zum Überwinden der Kluft zwischen abstraktem Formalismus und realen Anwendungen – und damit als Beitrag zur Motivation der Studierenden – bietet DAVE auch die Möglichkeit, die Simulation eines Zustandsdiagramms multimedial zu untermalen. Wird diese Funktion genutzt, dann erscheint zur Steuerung der Simulation nicht das zuvor erwähnte Fenster, sondern eines, das eine graphische Darstellung eines realen Gerätes enthält. Den Studierenden wird der Eindruck vermittelt, daß das von ihnen erstellte Zustandsdiagramm die (eingebettete) Steuerungssoftware für ein vorhandenes Stück Hardware realisiert. Entsprechend besteht ihre Aufgabe darin, das Verhalten des Gerätes auf der Basis natürlichsprachlicher Anforderungen möglichst gut zu modellieren.

Das im Arbeitsbereich erstellte Modell erhält seine Eingabeereignisse und die aktuellen Werte der Überwachungsbedingungen von diesem Gerät. Ebenso werden Ausgabeereignisse an das Gerät geleitet und wirken sich auf dessen Zustand aus. Zustandsänderungen des Gerätes werden durch Änderungen der graphischen Darstellung wiedergegeben. Gelangt das Gerät in einen fehlerhaften Zustand, dann kann dies ebenfalls entsprechend dargestellt werden, und die Simulation endet. Damit das Zusammenspiel zwischen Modell und Gerät überhaupt möglich ist, müssen Ereignisse und Überwachungsbedingungen zuvor mit den Studierenden vereinbart werden. Dies kann zum Beispiel im Rahmen eines Aufgabentextes geschehen, wie ihn Abbildung 2.3 zeigt. Eine solche Aufgabenstellung stellt die Studierenden nicht vor größere Probleme, da sie der üblichen Vorgehensweise des Programierens „gegen“ eine feste Schnittstelle entspricht.

DAVE enthält zur Zeit zwei multimediale Visualisierungen solcher Geräte:

- Eine Waschmaschine, bei der Eigenschaften wie Motorgeschwindigkeit sowie Zu- und Ablauf von Wasser kontrolliert werden können. Das Zustandsdiagramm der Studierenden soll einen kompletten Waschvorgang realisieren. Die Visualisierung gibt Hinweise auf die Korrektheit der Lösung: Zu Beginn der Simulation wird schmutzige Wäsche in die Maschine gelegt. Am Ende eines korrekten Waschvorgangs sollte diese Wäsche sauber sein. Fatale Fehler im Zustandsdiagramm enden zum Beispiel in einer Überschwemmung. Abbildung 4 zeigt einige Bilder der Waschmaschine zu verschiedenen Zeitpunkten der Simulation.

- Eine Kaffeemaschine, bei der das Zustandsdiagramm die Heizvorrichtung kontrolliert und korrekt auf die Einstellungen des Benutzers reagieren soll. Zudem müssen bestimmte „Sicherheitsbestimmungen“ wie eine automatische Abschaltung nach einer vorgegebenen Zeit berücksichtigt werden. Auch hier gibt die Visualisierung Hinweise auf die Korrektheit der Lösung: Erscheint Kaffee in der Kanne und schaltet sich die Maschine nach der geforderten Zeit automatisch ab, dann ist die Lösung gutartig – geht die Maschine in Flammen auf, dann ist die Lösung verbesserungsbedürftig.

Aufgabe:

Die für ingenieurmäßige Spitzenleistungen bekannte Firma LS10 Haushaltsgeräte bringt in Kürze ihre neue Waschmaschine auf den Markt. Die Hardware des Gerätes funktioniert bereits und ist in der Lage, über eine Reihe von Ein-/Ausgabesignalen mit ihrer Umwelt zu kommunizieren. Ihre Aufgabe besteht darin, eine passende Rechnersteuerung mit Hilfe eines Zustandsdiagramms zu entwerfen.

Das gewünschte Verhalten der Maschine sieht wie folgt aus: Nach dem Einschalten beginnt die Maschine den Hauptwaschgang, der insgesamt 30 Minuten dauert. Zu Beginn des Hauptwaschgangs wird für zwei Minuten die Wasserzufuhr eingeschaltet, während des Hauptwaschgangs läuft der Motor mit geringer Geschwindigkeit. Auf den Hauptwaschgang folgt das Schleudern, das insgesamt 10 Minuten dauert, von denen die letzten 2 Minuten zum Abpumpen des Wassers genutzt werden. Am Ende des gesamten Programms wird ein akustischer Alarm ausgelöst. Die Maschine kann nur gestartet werden, wenn die Luke geschlossen ist, und während eines laufenden Programms wird die Luke verriegelt, so daß sie nicht geöffnet werden kann. Aus Sicherheitsgründen darf nicht geschleudert werden, wenn die Maschine überladen ist. In diesem Fall wird bereits in den letzten zwei Minuten des Hauptwaschgangs abgepumpt. Die Maschine kann jederzeit während des Programms durch erneutes Drücken des Netzschalters abgeschaltet werden. Wird sie anschließend wieder eingeschaltet, nimmt sie das Programm an der alten Position auf. Die Maschine wird im geschlossenen Zustand geliefert.

Die Hardware der Maschine liefert ihnen potentiell folgende Eingabesignale:

- `DOOR_CLOSED` – Tür wurde geschlossen
- `DOOR_OPENED` – Tür wurde geöffnet
- `POWER` – Netzschalter wurde betätigt

Sie können die Hardware mit folgenden Ausgabesignalen kontrollieren:

- `MOTOR_OFF` – Motor aus
- `MOTOR_SLOW` – Motor auf niedrige Umdrehungszahl
- `MOTOR_FAST` – Motor aus hohe Umdrehungszahl
- `WATER_OFF` – Wasserzufuhr aus
- `WATER_ON` – Wasserzufuhr an
- `PUMP_OFF` – Pumpe aus
- `PUMP_ON` – Pumpe an
- `DOOR_LOCK` – Tür verriegeln
- `DOOR_UNLOCK` – Tür freigeben
- `BEEP` – Akustisches Signal auslösen

Der interne Gewichtssensor setzt die Überwachungsbedingung `LOAD_HEAVY` auf wahr, wenn die Maschine überladen ist.

Abbildung 3: Beispielhafter Aufgabentext für DAVE

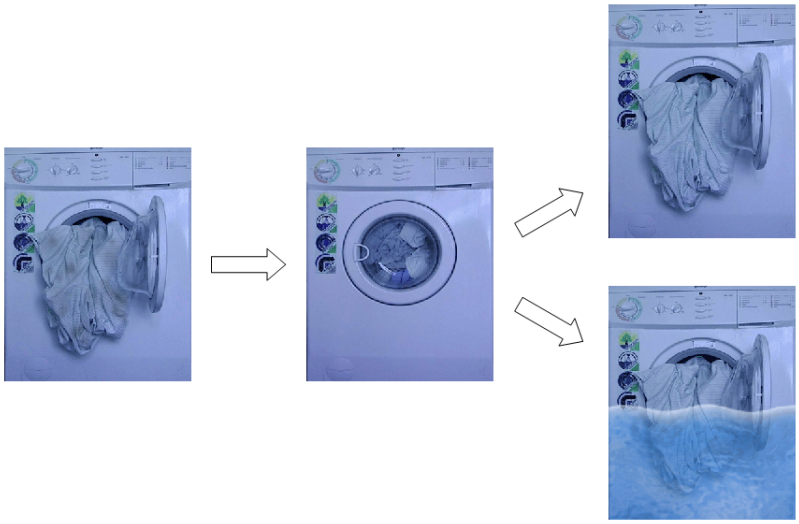


Abbildung 4: Untermalung der Simulation durch eine Waschmaschine

Beide Beispiele wurden mit relativ geringem Aufwand digital fotografiert und anschließend entsprechend nachbearbeitet bzw. in einzelne „Kacheln“ unterteilt, die je nach aktuellem Zustand kombiniert werden können. Die Umsetzung innerhalb der Simulationssteuerung ist mit einem gewissen, von der Komplexität des gewünschten Verhaltens abhängigen Aufwand verbunden. Da jedoch auf der Basis jeder Visualisierung mehr als eine Übungsaufgabe gestellt werden kann, zahlt sich dieser Aufwand schnell aus. Neue Visualisierungen können dem Werkzeug über eine definierte Schnittstelle hinzugefügt werden.

3 Vorstellung der Lerneinheit SAM

Während DAVE eher ein Prototyp ist, der sich im positiven Sinne verselbständigt hat, stellt das Werkzeug zur Softwarearchitektur-Modellierung (SAM) das primäre Ergebnis des Teilprojektes dar. SAM baut technisch wie inhaltlich auf DAVE auf und erweitert diesen um Modellierungsmöglichkeiten für die strukturellen Anteile eines Softwaresystems. Die vom Werkzeug unterstützte Notation unterstützt die wesentlichen Kontrukte, die in der UML zur Beschreibung von Architekturen vorgesehen sind:

- Die Konzepte *Kapsel*[2], *Port*, *Protokoll* und *Konnektor* definieren strukturelle Komponenten und deren potentielle Kommunikationsbeziehungen. Über Aggregation ergeben sich aus primitiven Kapseln komplexere Kapseln und komplette Systeme.

[2]Wird in der Spezifikation eigentlich als *Part* bezeichnet, aber aufgrund der leichten Verwechselbarkeit mit *Port* bevorzugen wir den aus der älteren Literatur stammenden Begriff *Kapsel*.

- Das reaktive Verhalten jeder Kapsel – und damit auch des gesamten Systems – wird über Zustandsdiagramme beschrieben. Die Ereignisse, die das Zustandsdiagramm einer Kapsel als Ausgabe produziert, können über Konnektoren an andere Kapseln weitergegeben und dort konsumiert werden.

3.1 Modellieren von Architekturen

Aus der Notation ergeben sich zusätzlich zu den aus DAVE bekannten Zustandsdiagrammen zwei weitere Diagrammtypen, die vom SAM unterstützt werden:

- Eine spezielle Art von Klassendiagramm dient zur Spezifikation der einzelnen Kapsel- und Protokollklassen. Jeder Kapsel werden in diesem Diagramm die Ports hinzugefügt, über die sie mit der Außenwelt kommunizieren kann. Jeder Port „spricht“ ein bestimmtes Protokoll und nimmt dabei eine bestimmte Rolle ein (etwa entsprechend Client und Server). Für jedes Protokoll werden die ein- und ausgehenden Signale definiert. Zudem wird hier festgelegt, welche Kapsel das Gesamtsystem (analog zur einer Java-Klasse, deren `main`-Methode ausgeführt wird) repräsentiert.

- Die Struktur jeder Kapsel wird über ein Diagramm spezifiziert, das sich am ehesten mit einem Objektdiagramm vergleichen läßt. Hier werden andere Kapseln instantiiert und über Konnektoren untereinander sowie mit den Ports der sie umgebenden Kapsel verbunden. Damit zwei Ports miteinander verbunden werden können, müssen sie das gleiche Protokoll verwenden, aber entgegengesetzte Rollen einnehmen. Eine Ausnahme bildet ein spezieller Port innerhalb jeder Kapsel, der sogenannte Endport. Dieser repräsentiert das Zustandsdiagramm der Kapsel selbst, also quasi deren Verhalten. Er kann mit allen anderen Ports unabhängig von Protokoll und Rolle verbunden werden.

Abbildung 5 zeigt das Klassendiagramm eines Compilers. Man erkennt im Diagramm die einzelnen Grundkomponenten des Compilers – lexikalische, syntaktische und semantische Analyse sowie die Code-Generierung – und die Protokolle, die deren Zusammenspiel reglementieren. Verschiedene Varianten von Compiler-Architekturen liegen dem Werkzeug als Fallbeispiel bei.

3.2 Simulieren von Architekturen

Aus den gleichen Beweggründen, die schon bei DAVE angeführt wurden, unterstützt auch SAM die Simulation des modellierten Systems: Den Studierenden soll über das Verständnis der reinen Syntax der Architekturbeschreibung hinaus ein Einblick in deren Laufzeitsemantik ermöglicht werden. Tatsächlich gilt dieses Argument sogar für eine Architekturbeschreibung in weitaus stärkerem Maße, denn das Gesamtverhalten einer aus mehreren, nebenläufig agierenden Kapseln bestehenden Architektur erschließt sich ungleich schwerer als das eines einzelnen Zustandsdiagramms. Bei SAM spielt die Simulation also eine noch wichtigere Rolle als bei DAVE. Gleichzeitig ist sie technisch schwieriger umzusetzen, da von jeder Kapsel potentiell beliebig viele Instanzen existieren können und das Zusammenspiel aller Kapseln und der daraus resultierende Gesamtzustand des System korrekt sein müssen.

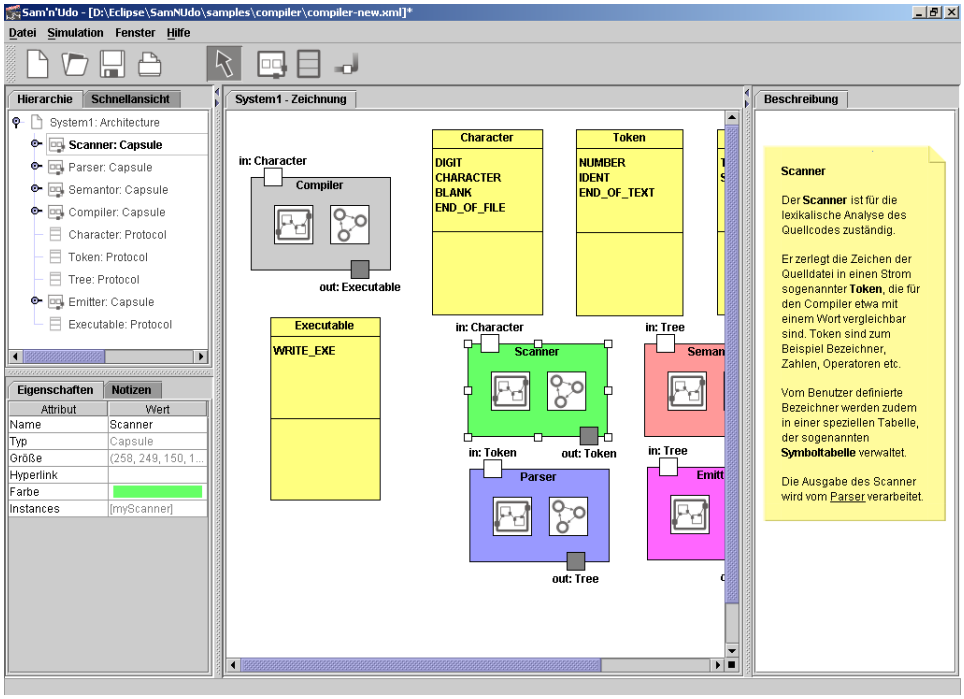


Abbildung 5: Bearbeiten einer Architektur in SAM

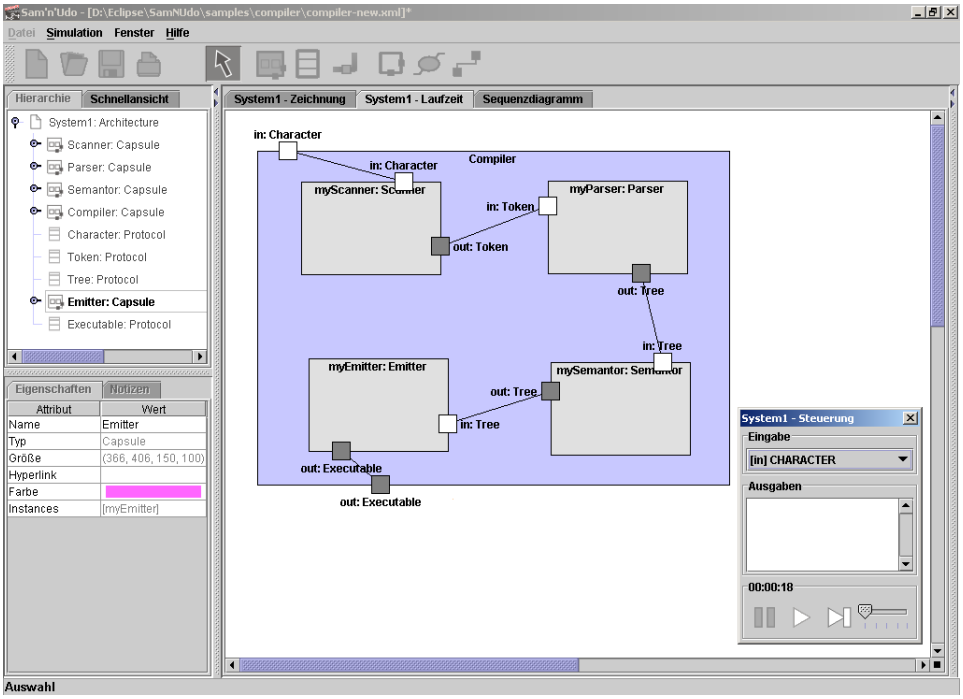


Abbildung 6: Simulation einer Architektur in SAM

Basis für die Architektursimulation ist wieder die Semantik von Zustandsdiagrammen. Jede Instanz einer Kapsel besitzt eine eigene Instanz des Zustandsdiagramms der zugehörigen Kapsel.[3] Jede Kapsel verwaltet zudem eine Warteschlange von Ereignissen, die vom Zustandsdiagramm schrittweise konsumiert und verarbeitet werden. Aus den Konnektoren zwischen einzelnen Kapseln ergibt sich ein Fluß von Ereignissen innerhalb des Systems.

Beim Start der Simulation erscheint das von DAVE bekannte Steuerungsfenster (siehe Abbildung 6 – hier erkennt man auch deutlich, daß es sich bei dem Compiler um eine „Pipes and Filters“-Architektur handelt). Mit ihm können Eingabeereignisse für die nach außen geführten Ports des Systems produziert werden. Gelangen im Verlauf der Simulation Ausgabeereignisse an diese Ports, dann werden sie angezeigt. Ebenfalls beim Start der Simulation erscheint ein neues Diagramm, das eine Instanzsicht des Systems bietet. In diesem Diagramm sind die Kapseln der obersten Ebene des Systems zu sehen. Ein Doppelklick auf eine der Kapseln zeigt deren strukturelle Innereien oder das Zustandsdiagramm inklusive der aktuellen Konfiguration an. Um den Ereignisfluß im Gesamtsystem leichter nachvollziehen zu können, wird die Kommunikation zwischen Kapseln während der gesamten Simulation in einem Sequenzdia-

[3] Korrekt betrachtet teilen sich alle Instanzen das gleiche Zustandsdiagramm, aber jede verwaltet ihre eigene Zustandskonfiguration.

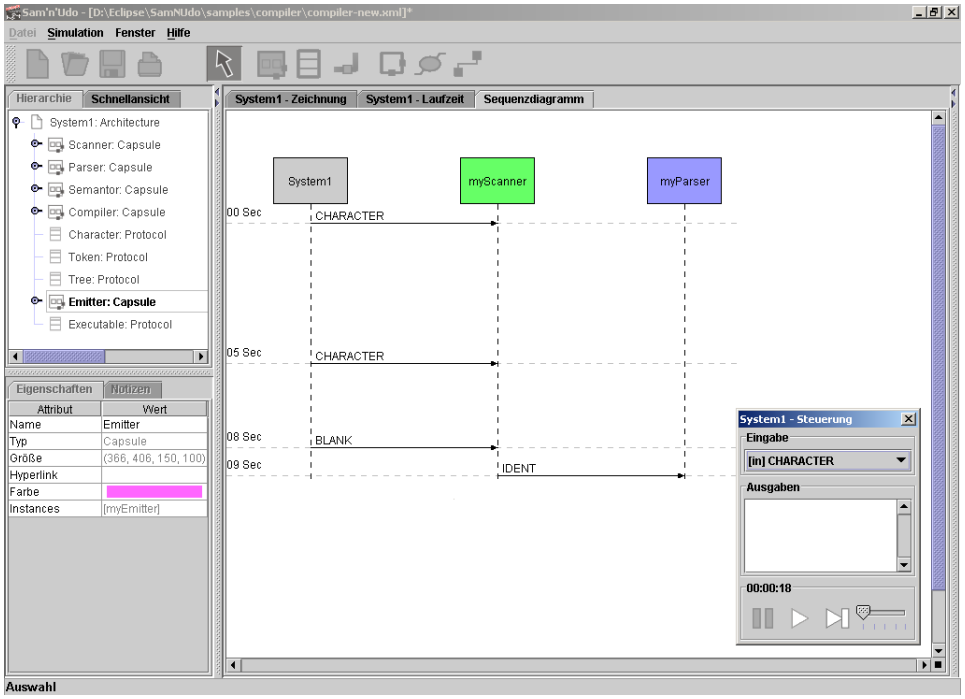


Abbildung 7: Sequenzdiagramms einer Simulation in SAM

gramm aufgezeichnet (siehe Abbildung 7). Die Menge der Kapseln, die an diesem Diagramm beteiligt sind, kann frei gewählt werden.

3.3 Multimediale Untermalung der Simulation

Es bietet sich offensichtlich an, die neuartige Idee der multimedialen Untermalung einer Simulation, die in DAVE sehr erfolgreich umgesetzt wurde, auch in SAM zu nutzen. Die beiden Werkzeuge sind sowohl technisch als auch inhaltlich eng verwandt. Sie haben zudem beide mit dem grundsätzlichen Problem zu kämpfen, die Studierenden zum Einsatz einer formalen graphischen Notation zu motivieren und ihnen anhand möglichst realistischer Beispiele den Nutzen dieser Notation vor Augen zu führen.

Die Idee zur multimedialen Untermalung kam jedoch erst während der Entwicklung von DAVE auf. Mithin gehört sie nicht zum ursprünglichen Anforderungskatalog von SAM; dort war nur eine Animation der Simulation innerhalb des Diagramms geplant. Da die personellen Kapazitäten des Teilprojektes bis zum Ende des Jahres 2003 mit der Fertigstellung von SAM und der Durchführung und Auswertung der Evaluation ausgelastet waren, war eine Umsetzung der Untermalung zeitlich nicht mehr möglich. Sie wird vermutlich erst während der Verlängerung von MuSofT im Frühjahr 2004 geschehen. Zu den geplanten Beispielen zählen

etwa eine Menge von Geldautomaten, die zusammen mit dem Zentralrechner einer Bank ein verteiltes System bilden, oder eine Kreuzung, die mit mehreren gleichartigen Ampeln bestückt wird.

4 Technische Realisierung

Wie sich bereits in einer frühen Phase des Projektes herausstellte, deckten sich die technischen Anforderungen von DAVE und SAM an einigen Stellen mit denen von Teilprojekt 3.3, das sich mit dem Unified Process beschäftigt. Dort sollten unter anderem zwei Werkzeuge entstehen, mit denen auf der Basis einer vorgegebenen Notation – der des Unified Process – Prozeßmodelle visualisiert und bearbeitet werden können.

Eine Implementierung mehrerer solcher Modellierungswerkzeuge von Grund hätte ein ebenso häufiges (Neu-) Erfinden des Rades bedeutet. Dies schien weder aus softwaretechnischer Sicht noch angesichts der zur Verfügung stehenden personellen Kapazitäten sinnvoll. Da es zudem als sinnvoll erachtet wurde, möglichst viele im Lehrbetrieb eingesetzte Applikationen mit einem gemeinsamen Look and Feel auszustatten, wurde zunächst ein allgemeines, auf Java/Swing basierendes Framework realisiert, das die Entwicklung individueller lehretauglicher Modellierungswerkzeuge erleichtert.

Um der Zielsetzung Rechnung zu tragen, daß mit dem Framework keine vollwertigen professionellen Werkzeuge, sondern eher leichtgewichtige Werkzeuge für die Lehre erstellt werden sollen, wurde es Lightweight Modeling Framework (LIMO) getauft. Das Framework setzt sich aus drei Hauptkomponenten zusammen: Einer mehr oder minder festen Benutzerschnittstelle inklusive eines lauffähigen Applikationsrahmens, einem darunterliegenden variablen Metamodell und einem Hypertextsystem, das mit Lehrstoff gefüllt werden kann. Alle Komponenten sowie ihr Zusammenspiel werden in den folgenden Abschnitten beschrieben.

4.1 Benutzerschnittstelle

Mit Hinblick auf einen Einsatz im Rahmen der Lehre wurde Wert darauf gelegt, die Benutzerschnittstelle einfach, intuitiv und möglichst wenig ablenkend zu gestalten. Der Löwenanteil des Bildschirms wird von der graphischen Darstellung des Modells in Anspruch genommen, da dies der eigentliche Fokus der Applikation ist. Sämtliche Aspekte, die unmittelbar die Arbeit am Modell betreffen – etwa das Verschieben, Vergrößern, Verkleinern oder Verbinden von Elementen – wird in diesem Bereich gehandhabt. Es ist in keinem Fall notwendig, verschachtelte Menüstrukturen zu bemühen, um häufig auftretende Funktionen wie Laden, Speichern oder Drucken aufzurufen oder dem Modell neue Elemente hinzuzufügen. All diese ständig benötigten Funktionen sind in einer Werkzeugleiste (siehe 8) am oberen Rand des Fensters verfügbar, wobei Sorge dafür getragen wurde, jede Funktion mit einem großen und möglichst aussagekräftigen Symbol auszustatten.

Außer dem eigentlichen Modellierungsbereich und der Werkzeugleiste stellt die Benutzerschnittstelle der Applikation noch einige Komponenten bereit, die aus anderen Anwendungen bekannt sein dürften und deshalb nur kurz angerissen werden:

- Der *Navigator* (siehe 9) erleichtert dem Benutzer das Zurechtfinden in – speziell komplexeren – Modellen. Dazu zeigt er zum einen eine baumartige Hierarchie des Modells

Abbildung 8: Typische Werkzeugleiste

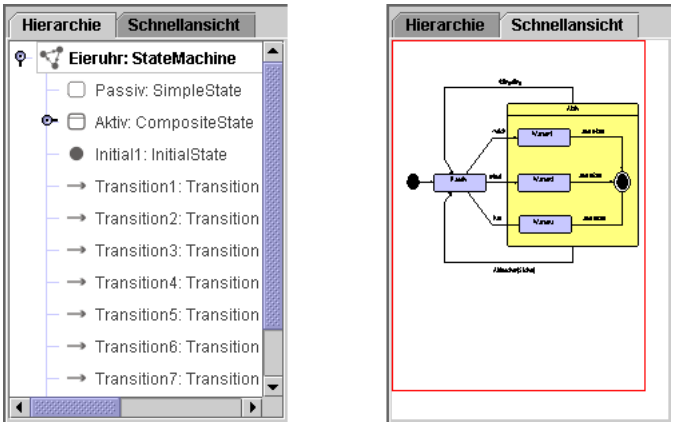


Abbildung 9: Navigator

(auf der Basis der Schachtelung von Elementen). Diese Ansicht ermöglicht eine schnelle Auswahl einzelner Elemente mit dem Effekt, dass diese Elemente auch im Arbeitsbereich zentriert angezeigt und ausgewählt werden. Neben der Hierarchie enthält der Navigator noch eine Übersicht des Modells in der Art einer Landkarte. Diese zeigt das Modell aus der Vogelperspektive und gibt einen Überblick über die Positionierung aller Modellelemente. Sie ist immer dann hilfreich, wenn der Umfang eines Modells einen einzelnen Bildschirm sprengt.

- Der *Inspektor* (siehe 10) stellt die Eigenschaften des aktuell ausgewählten Modellelementes dar und erlaubt ebenso deren Bearbeitung. Er kommt immer dann ins Spiel, wenn Eigenschaften von Elementen keine unmittelbare graphische Repräsentation besitzen oder aus anderen Gründen nicht in der Zeichnung bearbeitet werden können. Neben diesen Eigenschaften, die in einer Tabelle dargestellt werden, bietet der Inspektor noch die Möglichkeit, zu jedem Modellelement eine längere Freitextnotiz einzugeben.

Dem üblichen Prinzip der Positionierung von Navigationselementen folgend, sind Inspektor und Navigator am linken Bildschirmrand angeordnet. Alle vier können problemlos ausgeblendet werden, wenn sie nicht benötigt sind – etwa wenn innerhalb einer Vorlesung nur ein Modell präsentiert werden soll.

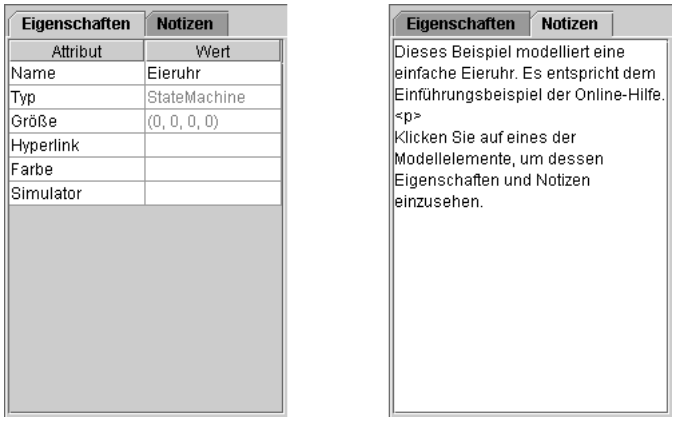


Abbildung 10: Inspektor

4.2 Metamodell

Die im vorangehenden Abschnitt vorgestellte Benutzerschnittstelle bietet verschiedene Sichten auf ein Modell, dessen Syntax und Semantik von einem applikationsspezifischen Metamodell bestimmt wird. Außer den Figuren, die zur graphischen Repräsentation von Elementen benötigt werden, ist es genau dieses Metamodell, das den eigentlichen Unterschied zwischen den verschiedenen existierenden und möglichen Modellierungswerkzeugen ausmacht. Es ist auch gleichzeitig der Teil, der mit den größten Implementierungsaufwand nach sich zieht, so dass es Sinn machte, an dieser Stelle nach einer möglicht allgemeinen Lösung zu suchen.

Die Lösung, die innerhalb des LIMO-Frameworks Anwendung findet, basiert auf dem Gedanken eines generischen Metamodells oder – anschaulicher ausgedrückt – auf einem Baukasten von Elementen, die sich leicht zu applikationsspezifischen Metamodellen kombinieren lassen. Das generische Metamodell verwendet Ideen, die auch in der Meta Object Facility (MOF) [Obj02] oder anderen Ansätzen zur Metamodellierung zu finden sind. Es ist jedoch weder eine Implementierung des MOF noch handelt es sich dabei um ein Meta-Metamodell im strikten Sinne. Dies liegt daran, dass die Elemente der applikationsspezifischen Metamodelle aus pragmatischen Gründen von denen des generischen Metamodells erben (statt diese zu instantiieren).

Im Kern besteht das generische Metamodell aus einer Graphstruktur, deren (schachtelbare) Knoten spezielle Konzepte oder Entitäten des Applikations-Metamodells repräsentieren und deren Kanten Beziehungen zwischen diesen Konzepten oder Entitäten herstellen. Jedes Element kann mit einer beliebigen Anzahl von eindeutig benannten Attributen oder Assoziationsenden dekoriert sein.

- Attribute dienen zur Speicherung primitiver Werte. Die derzeit unterstützten Datentypen sind *Boolean*, *Integer*, *String*, *Color* und *Date*. Ein Beispiel für einen Einsatzfall dieser Meta-Attribute sind die Sichtbarkeitsmarkierungen (*public*, *private*, ...) in objektorientierten Modellen.

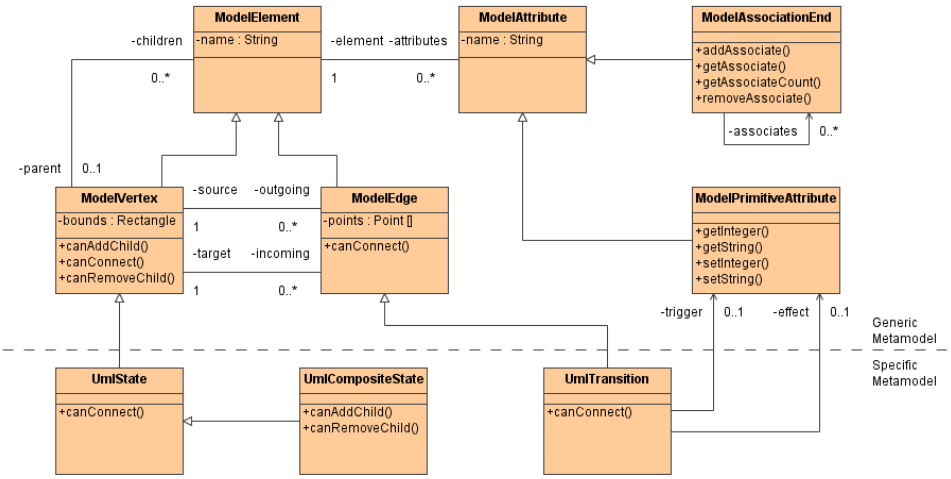


Abbildung 11: Vereinfachtes generisches und spezifisches Metamodell

- Assoziationsenden bieten Anknüpfungspunkte für Assoziationen. Letztere werden verwendet, um Elemente des Metamodells zu verbinden. Ähnlich wie in der UML, können Assoziationsenden eine Inverse und eine Multiplizität besitzen. Da sie vollwertige Mitglieder des Metamodells sind, kann die Implementierung problemlos beide Enden konsistent halten. Der Unterschied zwischen Assoziationen und den Kanten des Graphen liegt darin, dass erstere üblicherweise keine graphische Repräsentation besitzen.

Die Elemente des generischen Metamodells stellen bereits eine Menge von Methoden bereit, die zum Einhalten der Syntaxregeln einer speziellen Modellierungssprache eingesetzt werden können. Sowohl Knoten- als auch Kantenelemente besitzen etwa eine Methode *canConnect()*, die bestimmt, ob eine gegebene Kante zwei gegebene Knoten verbinden kann. Die Verbindung wird nur dann hergestellt, wenn alle beteiligten Elemente zustimmen. Diese und ähnliche Methoden werden üblicherweise in den applikationsspezifischen Metamodellen überschrieben.

Abbildung 11 gibt einen groben Eindruck vom Aufbau des Metamodells. Einige der grundlegenden Klassen des generischen Metamodells sind im oberen Bereich dargestellt. Der untere Bereich zeigt, wie Teile eines Metamodells für *UML*-Zustandsübergangsdiagramme von den generischen Klassen abgeleitet werden können. *UmlState* und *UmlCompositeState*, die beide offensichtlich einen Knoten-artigen Charakter besitzen, erben von *ModelVertex*. Die *UmlTransition* wird entsprechend ein Nachkomme von *ModelEdge*. Alle drei Klassen überschreiben verschiedene Methoden, die zur Einhaltung der Syntaxregeln benötigt werden: Zustände dürfen nur durch Transitionen verbunden werden (und umgekehrt), und zusammengesetzte Zustände können andere Zustände enthalten.

Der zuvor beschriebene fixe Teil des Frameworks arbeitet mit beliebigen applikationsspezifischen Metamodellen zusammen, ohne daß Änderungen etwa an der Benutzerschnittstelle

nötig wären. Dies wurde erreicht, indem bereits die grundlegenden Elemente des Metamodells mit Mechanismen zur Introspektion ausgestattet wurden. Diese erlauben es unter anderem, alle Attribute und Assoziationen eines Elementes zu erfragen und zu modifizieren. Auf dieser Basis ließen sich sowohl das Laden, Speichern und Drucken von Modellen auf eine allgemeine Weise realisieren wie auch die komplexeren Komponenten der Benutzerschnittstelle, also Inspektor und Navigator.

4.3 Hypertext

Am rechten Bildschirmrand des Anwendungsfensters, befindet sich ein Bereich in dem Hypertext auf Basis der Hypertext Markup Language (HTML) angezeigt werden kann (siehe 12). Dieser wird innerhalb des Werkzeugs zu verschiedenen Zwecken verwendet:

- Es besteht die Möglichkeit, zusammen mit dem Werkzeug eine feste Menge von HTML-Seiten auszuliefern. Dies wird innerhalb von DAVE und SAM sowohl zur Realisierung einer *Online-Hilfe* zur Verwendung des Programms als auch zur Integration des eingangs erwähnten Lehrstoffs genutzt. Es gibt eine feste Einstiegsseite (die übliche Datei `index.html`), die das Werkzeug initial anzeigt. Diese Seite sollte also ein Inhaltsverzeichnis oder Verweise zu weiteren Seiten enthalten. Sonst besteht zunächst keinerlei Beschränkung bezüglich des Lehrstoffes, der in ein Werkzeug integriert werden kann.

- Ein zweiter Reiter innerhalb des Hypertext-Bereiches zeigt Annotationen zu Modellelementen. Jedes Modellelement kann über einen Hyperlink an eine beliebige HTML-Seite gebunden werden. Diese wird im Hypertext-Bereich angezeigt, wenn das Modellelement ausgewählt wird. Eine mögliche Anwendung hierfür sind komplexe Beispielmodelle, die mit Annotationen versehen werden und von den Studierenden im Sinne entdeckenden Lernens erkundet werden sollen. Unterhalb der HTML-Seite wird auch die aus dem Inspektor stammende Freitextnotiz eines Elementes in Form einer gelben Haftnotiz angezeigt. Dies gibt den Studierenden die Möglichkeit, die Ergebnisse ihrer Erkundsaufträge an den relevanten Stellen zu vermerken.

In beiden Reitern des Hypertext-Bereiches können neben den üblichen HTML-Verweisen zwischen einzelnen Seiten spezielle Hyperlinks mit dem Präfix `limo:` verwendet werden. Diese führen beim Anklicken Aktionen aus, die das Modell oder die gesamte Anwenduntg betreffen. So ist es zum Beispiel möglich, über einen Hyperlink im Lehrstoff gezielt ein Modellelement zu fokussieren und zu selektieren, Veränderungen am Modell vorzunehmen oder sogar ein komplett neues Modell zu laden. Dies ermöglicht ein sehr eng verzahntes Zusammenspiel zwischen Hypertext und Modell und kann zum Beispiel zur Realisierung von *Guided Tours* innerhalb des Lehrstoffes dienen.

Ein anderes potentielles Einsatzszenario für den Hypertext ist ein papierloser Übungsbetrieb: Aufgaben können den Studierenden in Form von Hypertext innerhalb des Werkzeugs zur Verfügung gestellt werden, und die Studierenden erstellen und annotieren ihre Lösung mit den gleichen Mitteln. Anschließend kann die Lösung in digitaler Form an einen Tutor versandt werden, der ebenfalls das Werkzeug benutzt, um sie zu bewerten. Dieses Szenario ist in abgeschwächter Form beim Einsatz von DAVE bereits erprobt worden.

Abbildung 12: Hypertext-Bereich

4.4 Bibliotheken

Zur Reduzierung des Entwicklungs- und Wartungsaufwandes innerhalb des Frameworks und der konkreten Anwendungen DAVE und SAM wurde an einigen Stellen auf existierende Java-Bibliotheken zurückgegriffen.

Bei der Entwicklung der Benutzerschnittstelle des Frameworks war die Verwendung von JHotDraw [Kai01] hilfreich, einer Klassenbibliothek für Applikationen, die sich mit technischen Zeichnungen im weiteren Sinne auseinandersetzen. JHotDraw baut wesentlich auf dem Begriff einer *Zeichnung* auf (dargestellt in unserem Arbeitsbereich), die eine beliebige Anzahl von sogenannten *Figuren* enthält (graphischen Repräsentaten unserer Modellelemente).

Die Bibliothek wurde ursprünglich von Gamma und Beck entwickelt, um die Verwendung von Entwurfsmustern [GHJV96] zu verdeutlichen. Trotzdem besitzt sie interessanterweise keine besonders ausgeprägte Trennung von Modell und Views (im Sinne von MVC), d.h. es existiert kein echtes Modell, das unter den Figuren liegt. Somit war es notwendig, die existierenden Figuren der Bibliothek mit unserem (Meta-) Modell zu unterfüttern. JHotDraw wird als *Open Source* auf SourceForge (http://www.sourceforge.net) gepflegt und ist zur Nutzung unter der GNU Lesser General Public License (LGPL) verfügbar.

Weiterhin kam die Bibliothek *Kilobyte XML* (KXML) [Hau04] zum Einsatz, die Funktionalität zum Lesen und Schreiben von XML-Dateien (Modelle und Konfigurationsdateien) bereitstellt. KXML wurde anstelle der in neueren Versionen von Java enthaltenen XML-Unterstützung gewählt, weil der Parser nicht ereignisbasiert arbeitet, sondern als XML-Pull-Parser [SH02] einen Strom von Token erzeugt. Die Verarbeitung dieser Token – und damit der Kontrollfluß – verbleiben bei der eigentlichen Anwendung, was dem Umgang mit den XML-Daten erheblich vereinfacht. KXML wird ebenfalls auf SourceForge gepflegt und unterliegt wie JHotDraw der LGPL.

Schließlich wurde noch für die Werkzeuge DAVE und SAM eine Simulationskomponente für Zustandsdiagramme eingesetzt, die auf das UVM-Projekt *Virtuelle Welt* zurückgeht. Diese Komponente unterstützt im wesentlichen UML-Zustandsdiagramme, wie sie in Version 1.5 bzw. der neueren Version 2.0 der UML-Spezifikation beschrieben sind. Die Anbindung der Simulation an ein anderes Projekt geschieht (auf Seiten der nutzenden Anwendung) über das Implementieren einiger weniger Java-Schnittstellen, die von der Simulation vorausgesetzt werden. Durch diese lose Anbindung ist sichergestellt, daß sowohl die Anwendungen als auch die Simulationskomponente unabhängig voneinander gepflegt werden können und insbesondere erstere von den kontinuierlichen Verbesserungen an letzterer profitiert.

5 Erfahrungen

Im Sommersemester 2003 war die Entwicklung des Werkzeugs DAVE so weit fortgeschritten, daß ein erster evaluierter Einsatz im Vorlesungsbetrieb möglich war. Die Evaluation wurde vom Hochschuldidaktischen Zentrum (HDZ) der Universität Dortmund unterstützt. Sie war in eine umfangreichere Erhebung eingebettet, innerhalb derer auch andere Fragen untersucht wurden. Dazu zählten zum Beispiel das allgemeine Lernklima im Studiengang Informatik und der Einfluß des Geschlechts auf die Einstellung zu und die Nutzung von E-Learning-

Angeboten. Details der Untersuchung sind in [MGKS+04] nachzulesen. Die hier genannten Zahlen stammen ebenfalls aus diesem Bericht.

5.1 Vorgehen

Aufgrund der thematischen Nähe bot sich für die Evaluation von DAVE die Vorlesung „Softwaretechnik“ an, die in Dortmund zu diesem Zeitpunkt im Hauptstudium (sechstes Semester) angeboten wurde. Im Rahmen dieser Veranstaltung werden neben anderen Spezifikations- und Beschreibungssprachen UML-Zustandsdiagramme behandelt. Die Vorlesung wurde von etwa 100 Studierenden besucht, 80 davon haben an den Übungen teilgenommen. Einsatz und Evaluation von DAVE im Rahmen der Veranstaltung gestalteten sich wie folgt:

- Im Anschluß an die üblichen einführenden Vorlesungen zu Zustandsdiagrammen fand eine etwa einstündige Vorlesung zur Verwendung des Werkzeugs statt. Diese Vorlesung wurde von mehreren Mitarbeitern des HDZ beobachtet.

- Zur Bearbeitung der vorlesungsbegleitenden Übungszettel zu Zustandsdiagrammen (zwei Aufgabenzettel mit jeweils zwei umfangreicheren Aufgaben) haben die Studierenden das Werkzeug verwendet. Eine Gruppe von Studierenden wurde bei ihrem Erstkontakt mit dem Werkzeug durch das HDZ beobachtet und anschließend befragt.

- Bei der Besprechung des zweiten Aufgabenzettels in den Übungsgruppen wurde ein umfangreicher, mit Hilfe des HDZ erarbeiteter Fragebogen an die Studierenden ausgehändigt. Mit einer einzigen Ausnahme haben alle Studierenden diesen Fragebogen ausgefüllt, so daß die Rücklaufquote der Befragung bei nahezu 100% lag.

- Schließlich fanden noch Interviews mit den zwei Mitarbeitern statt, die für die Betreuung der Übungsgruppen verantwortlich waren.

5.2 Feedback der Studierenden

Die informalen und formalen Befragungen der Studierenden ergaben ein weitgehend positives Bild des Werkzeugs. Die folgenden Abschnitte geben das Feedback der Studierenden in verschiedenen Bereichen, die durch den Fragebogen berührt wurden, wieder.

5.2.1 Erwartungshaltung

Die Studierenden standen dem Werkzeugseinsatz nach der einführenden Vorlesung überwiegend positiv gegenüber (65% Zustimmung, 27% Ablehnung, 8% unentschieden) und sahen den Mehrwert, den das Werkzeug mit sich bringt (76% Zustimmung, 20% Ablehnung, 4% unentschieden). Eine Studierende oder ein Studierender äußerste auf dem Fragebogen, daß der Aufgabenzettel zu DAVE der erste sei, auf den sie oder er sich gefreut habe. Es ist also davon auszugehen, daß bei den Studierenden prinzipiell eine große Offenheit für den Einsatz neuer Medien als unterstützendes Element in einer Vorlesung bzw. im Übungsbetrieb vorhanden ist.

5.2.2 Installation

Die Installation von DAVE bestand im wesentlichen darin, eine ZIP-Datei von der Web-Seite des Lehrstuhls zu laden, die Datei zu entpacken und anschließend das Programm zu starten. Dies stellte für die meisten Studierenden keine größere Hürde dar (86% hatten keine Probleme, 13% hatten lösbare Probleme, 1% schaffte die Installation gar nicht). Die Probleme bestanden fast ausschließlich darin, daß zunächst eine aktuelle Java-Laufzeitumgebung eingerichtet werden mußte (34% mußten Java erstmalig installieren oder aktualisieren, 63% besaßen bereits ein aktuelles Java, 3% machten keine Angabe). Diejenigen Studierenden, die zunächst Java installieren mußten, hatten damit im wesentlichen keine Schwierigkeiten (81% hatten keine Probleme, 19% hatten lösbare Probleme).

Man würde bei Studierenden der Informatik vermuten, daß sie die Installation einer Software nicht vor unlösbare Probleme stellt, und die Zahlen bestätigen dies. Um jedoch den Vorgang für Studierende mit geringerem technischen Vorwissen (etwa außerhalb der Informatik) zu erleichtern, wird überlegt, künftige Versionen entweder mit einem automatischen Installationsprogramm oder über den WebStart-Mechanismus von Java auszuliefern. Letzterer erlaubt durch einen speziellen Hyperlink in einer Web-Seite die automatische Integration der Software in die Arbeitsoberfläche des Nutzers (inklusive der Erstellung eines Symbols und regelmäßiger Prüfungen auf aktualisierte Versionen).

5.2.3 Verwendung

Die Verwendung des Werkzeugs wurde von den Studierenden positiv beurteilt. Die Oberflächengestaltung wurde als ansprechend empfunden (89% Zustimmung, 10% Ablehnung, 1% unentschlossen), was sicherlich auch der externen Beratung in Gestaltungsfragen zu verdanken ist, die MuSofT im Laufe des Jahres 2002 erfahren hat. Die Ausführungsgeschwindigkeit war offenbar angenehm (79% Zustimmung, 21% Ablehnung), was bei der Verwendung von Java nicht selbstverständlich ist. Der Ressourcenbedarf scheint also im Rahmen dessen zu liegen, was die Rechner der Studierenden (fast alle nutzten das Werkzeug zuhause) bereitstellen. Es wäre interessant, an dieser Stelle eine vergleichende Untersuchung mit professionellen Werkzeugen anzustellen. Dies kann jedoch schlecht im Rahmen einer Vorlesung bzw. des Übungsbetriebes geschehen, da dann möglicherweise ein Teil der Studierenden bewußt benachteiligt würde. Hier wäre ein kontrolliertes Experiment mit Kleingruppen sinnvoller.

5.2.4 Simulation

Ein wesentliches Alleinstellungsmerkmal des Werkzeugs ist die Fähigkeit, Modelle zu simulieren. Diese Funktion wurde von allen Studierenden genutzt (75% ständig, 25% ein- oder mehrmals). Die Studierenden schätzten das frühzeitige Feedback, das sie durch diese Simulationsfunktion erhielten (90% Zustimmung, 8% Ablehnung, 2% unentschlossen) und sahen die Simulation als Hilfe beim Lösen der Aufgaben (94% Zustimmung, 6% Ablehnung). Die Mehrheit der Studierenden glaubt durch die Simulation ein besseres Verständnis für den zugrunde liegenden Formalismus der UML-Zustandsdiagramme erhalten zu haben (81% Zustimmung, 18% Ablehnung, 1% keine Angabe), was ja eines der erklärten didaktischen Ziele war. Kein einziger Studierender möchte auf die Simulationsfunktion zugunsten eines ausschließlichen

späteren Feedbacks durch den Übungsgruppenleiter verzichten (99% Ablehnung, 1% unentschlossen).

Ein großer Teil der Studierenden ist der Meinung, daß die Simulationsfunktion dazu verleitet, die Aufgaben nach dem Prinzip „Trial and Error“ zu lösen (52% Zustimmung, 47% Ablehnung, 1% unentschlossen). Dies muß aber nicht unbedingt negativ gesehen werden, da aus jedem Modellierungsfehler, der im Rahmen einer Simulation aufgedeckt wird, gelernt werden kann.

5.2.5 Visualisierung

Von den insgesamt vier Aufgaben waren die zwei, die durch multimediale Darstellungen echter Geräte visualisiert und unterstützt wurden, am beliebtesten. Die Möglichkeit zur Visualisierung wurde von den meisten Studierenden angenommen (38% bei jeder Aufgabe, 49% ein- oder mehrmals, 10% niemals)[4]. Mit dem für sie neuen Konzept, ein Modell nicht „auf der grünen Wiese“, sondern zur Steuerung eines gedachten Gerätes zu erstellen, hatten die Studierenden keine Schwierigkeiten. Nur wenigen Studierenden war unklar, wie ihr Modell technisch mit der Visualisierung der Waschmaschine bzw. Kaffeemaschine zusammenspielt (13% Zustimmung, 85% Ablehnung, 2% unentschlossen). Es liegt nahe zu vermuten, daß sich die Studierenden, die hier Probleme hatten, mit denen decken, die die Visualisierung niemals genutzt haben. Diese Korrelation wurde aber nicht geprüft.

Die meisten Studierenden fühlten sich durch die anschaulichen Beispiele angesprochen (75% Zustimmung, 22% Ablehnung, 3% unentschlossen) und sind der Meinung, daß diese dazu beitrugen, die abstrakten Probleme deutlicher zu machen (63% Zustimmung, 30% Ablehnung, 7% unentschlossen). Dies kann als Bestätigung für die didaktische Idee der Brücke zwischen abstraktem Formalismus und Realität gewertet werden. Nur wenige Studierende waren der Meinung, daß die Beispiele nicht hilfreich waren (14% Zustimmung, 79% Ablehnung, 7% unentschlossen). Trotzdem war nur etwas mehr als die Hälfte der Studierenden der Meinung, daß durch die Beispiele der praktische Nutzen des zugrundeliegenden Formalismus deutlich wurde (51% Zustimmung, 46% Ablehnung, 3% unentschieden). Hier besteht also noch Verbesserungsbedarf, zum Beispiel derart, daß die Einbettung von Zustandsdiagrammen in den gesamten Entwicklungsprozeß deutlicher gemacht wird.

5.2.6 Gesamteindruck

Insgesamt wurde das Werkzeug von den Studierenden positiv beurteilt. Die meisten würden das Werkzeug anderen Studierenden weiterempfehlen (75% Zustimmung, 25% Ablehnung). Ein sehr großer Teil der Studierenden wünscht sich ähnliche Werkzeuge für andere Themenbereiche der Vorlesung (86% Zustimmung, 14% Ablehnung). Hierfür bestehen bereits konkrete Pläne.

Kritisch angemerkt wurde von vielen Studierenden, daß sie sich während der Arbeit mit dem Werkzeug beeinträchtigt gefühlt haben (40% fühlten sich beeinträchtigt, 47% nicht, 13% waren unentschlossen oder haben keine Angabe gemacht). Als Grund wurde hier fast ausschließlich die Unausgereiftheit des Programms angegeben, was angesichts der Tatsache, daß

[4]Die Frage war ungünstig formuliert. Da nur die Hälfte der Aufgaben überhaupt die multimediale Visualisierung unterstützte, war es eigentlich nicht möglich, sie bei jeder Aufgabe zu nutzen

dies der erste Einsatz war, nicht verwunderlich ist. Die Studierenden zeigten eine Reihe von Schwachstellen und Fehlern auf, die im Anschluß an den Einsatz korrigiert werden konnten. Außerdem wurden einige Funktionen gewünscht, die in der ersten Fassung von DAVE nicht vorhanden waren. Dazu zählten zum Beispiel eine Online-Hilfe mit einem Einführungsbeispiel, ein Fangraster zum leichteren Ausrichten von Zeichnungselementen, eine bessere Behandlung von Stützpunkten in komplexen Transitionen, sowie eine „Undo“-Funktion und eine Unterstützung der Zwischenablage. Vieles davon dies wurde dem Werkzeug ebenfalls in den ersten Wochen nach dem Einsatz hinzugefügt.

5.3 Feedback der Betreuer

Das Feedback der beiden Übungsgruppenbetreuer war ebenfalls positiv. Hervorgehoben wurde die Erleichterung beim Korrigieren der Aufgaben, da hier ebenfalls die Simulationsfunktion des Werkzeugs zum Einsatz kommen konnte.

Außerdem ergab sich in der Diskussion, daß mit dem Werkzeug neue Aufgabentypen möglich sind, die sich bei einer Bearbeitung mit Papier und Bleistift nicht anbieten: Die Aufgaben können, da sie durch die Simulation unterstützt werden, insgesamt komplexer und damit realistischer werden. Wo traditionelle Übungszettel zu Zustandsdiagrammen oft im wesentlichen die Syntax abfragen, erlaubt die Simulation die gewünschten Einblicke in deren (Laufzeit-) Semantik. Die multimedial unterstützten Beispiele tragen zudem zur Motivation der Studierenden bei, indem sie die erwähnte Brücke zwischen abstraktem Vorlesungsstoff und einer praktischen Anwendung schlagen. Dies deckt sich mit den Aussagen der Studierenden im Fragebogen.

5.4 Weitere Planungen

Die beschriebenen Ergebnisse beziehen sich zunächst nur auf das Zustandsdiagramm-Werkzeug DAVE. Die Entwicklung des Architektur-Werkzeugs SAM war zwar im Sommersemester 2003 ebenfalls weitgehend abgeschlossen, aber das Werkzeug war nach Einschätzung der Entwickler noch nicht stabil genug, um damit einen ernsthaften Übungsbetrieb durchzuführen. Eine vollständige Evaluation von SAM wird erst im Frühjahr 2004 in Kleingruppen bzw. im Übungsbetrieb einer erneuten Iteration der oben erwähnten Veranstaltung durchgeführt werden.

Die bei der Evaluation von DAVE erzielten Ergebnisse sind jedoch bereits jetzt auf SAM übertragbar: Beide Werkzeuge besitzen in Form des LIMO-Framworks die gleiche technische Basis, so daß Aussagen über Benutzerschnittstelle, Verwendbarkeit und mögliche Fehler unmittelbar für alle Werkzeuge der Familie gelten. Insbesondere profitiert auch SAM von den im Anschluß an die Evaluation durchgeführten Korrekturen bzw. Erweiterungen. Da Zustandsdiagramme zudem auch inhaltlich eine Untermenge dessen darstellen, was von SAM abgedeckt wird – dort kommt noch die strukturelle Komponente hinzu –, gilt hier ebenfalls eine Übertragbarkeit der Ergebnisse.

DAVE wird im Wintersemester 2003/2004 in Dortmund erneut in einer Grundstudiumsveranstaltung zur UML-Modellierung eingesetzt. Hier nutzen etwa 350 Studierende das Werkzeug zum Lösen der Übungsaufgaben. Auch dieser Einsatz wird durch einen Fragebogen evaluiert werden.

6 Zusammenfassung

Ausgangspunkt von Teilprojekt 2.1 war die Überlegung, daß die Architektur von Software-Systemen ein sehr praxisorientiertes Thema innerhalb der Softwaretechnik darstellt und auf eine entsprechende Weise gelehrt werden sollte. Die rein theoretisch-instruktive Vermittlung der Eigenschaften bestimmter Architekturen oder architektureller Stile in Vorlesungen befähigt die Studierenden nicht automatisch zum Entwurf komplexer Software-Systeme – genau wie das reine Wissen um *if*-, *for*- und *while*-Konstrukte sie nicht dazu befähigt, strukturiert zu programmieren. In beiden Fällen ist das praktische Einüben des vermittelten Wissens nötig.

Die im Teilprojekt entstandene Lerneinheit unterstützt die Studierenden bei diesem praktischen Einüben. Dazu stellt sie als primäres Ergebnis das Werkzeug SAM bereit, das zum Modellieren von Software-Architekturen auf Basis einer Untermenge UML dient. In erster Näherung ist SAM ein CASE-Tool, das in besonderem Maße auf die Bedürfnisse der Lehre in der Softwaretechnik ausgerichtet ist. Neben einer generellen Beschränkung des Funktionsumfangs auf das, was in der Lehre wirklich benötigt wird, ist die Möglichkeit der Simulation des modellierten Systems ein zentrales Alleinstellungsmerkmal von SAM. Die Simulation erlaubt Einblicke in das Verhalten des Systems und trägt gleichzeitig zu einem vertieften Verständnis der Semantik der Beschreibungssprache bei.

Ein weiteres Ergebnis des Teilprojektes, das sich in vielerlei Hinsicht als sehr wertvoll erwiesen hat, ist das Werkzeug DAVE. DAVE ermöglicht die Modellierung und Simulation von UML-Zustandsdiagrammen. Obwohl ursprünglich nur als Prototyp von SAM geplant, wurde DAVE im Verlauf des Projektes zu einem eigenständigen Produkt, das für praktisch alle Veranstaltungen geeignet ist, in denen die UML und/oder Zustandsdiagramme Berücksichtigung finden. Insbesondere die multimediale Untermalung der Simulation macht DAVE zu einer Bereicherung gegenüber der klassischen Lehre von Zustandsdiagrammen auf der Basis von Papier und Bleistift. Die Evaluation des Einsatzes von DAVE im Sommersemester 2003 bestätigt dies.

Das vielleicht nachhaltigste Ergebnis des Teilprojektes ist das Framework LIMO zur Realisierung lehretauglicher graphischer Editoren. Auf Basis von LIMO wurden sowohl DAVE und SAM realisiert als auch die beiden Werkzeuge zur Prozeßmodellierung, die in Teilprojekt 3.3 entstanden sind. Weitere Werkzeuge auf Basis des Frameworks sind am Lehrstuhl geplant bzw. bereits in Arbeit: Im Rahmen einer Diplomarbeit entsteht zur Zeit ein Werkzeug zur Modellierung und Simulation von Petrinetzen [Pet62], das insbesondere die Erfahrungen mit der multimedialen Untermalung einer Simulation auf diesen Formalismus zu übertragen versucht. Im Laufe des Jahres 2004 soll zudem ein UML-Werkzeug entstehen, das Zustands-, Klassen-, Objekt- und Sequenzdiagramme integriert und sich zur Unterstützung von einführenden Softwaretechnik-Veranstaltungen eignet. Auch hier werden die bereits existierenden Simulationskomponenten eine zentrale Rolle spielen.

Die Ergebnisse des Teilprojektes konnten an verschiedenen Stellen [Ple03, APS03, APS04a, APS04b] erfolgreich publiziert werden.

7 Evaluierung

Zur Evaluierung der Ergebnisse des Teilprojektes fand im Sommersemester 2003 ein Einsatz des Werkzeugs DAVE statt, der durch das Hochschuldidaktische Zentrum (HDZ) der Universität Dortmund begleitet wurde. Aufgrund der thematischen Nähe bot sich für die Evaluation die Vorlesung „Softwaretechnik“ an, die in Dortmund zu diesem Zeitpunkt im Hauptstudium (sechstes Semester) angeboten wurde. Die Vorlesung wurde von etwa 100 Studierenden besucht, 80 davon haben an den Übungen teilgenommen. Das Werkzeug wurde durch den Autor in einer Vorlesung präsentiert und von den Studierenden für die Dauer von zwei Wochen zur Bearbeitung von Übungsaufgaben verwendet.

Neben einer Reihe von Beobachtungen von und Interviews mit den Studierenden stützt sich die Evaluation wesentlich auf einen formalen Fragebogen, der im Anschluß an diesen Einsatz von den Übungsteilnehmern ausgefüllt wurde. 79 von 80 Teilnehmern haben diesen Fragebogen ausgefüllt, so daß die Rücklaufquote praktisch bei 100% liegt. Die vollständigen Ergebnisse des Fragebogens wie auch der weiteren Maßnahmen finden sich in [MGKS+04].

Folgende Ergebnisse wurde auf Basis des Fragebogens nachgewiesen:

- **Erwartungshaltung:** Die Erwartungshaltung der Studierenden dem Werkzeug gegenüber war positiv (65% Zustimmung, 27% Ablehnung, 8% unentschieden). Die Studierenden sahen den Mehrwert, den die Nutzung des Werkzeugs mit sich bringen würde (76% Zustimmung, 20% Ablehnung, 4% unentschieden).
- **Installation:** Das Herunterladen von DAVE aus dem Internet und die anschließende Installation stellten für die Studierenden keine größere Hürde dar (86% hatten keine Probleme, 13% hatten lösbare Probleme, 1% schaffte die Installation gar nicht). Eine Reihe von Studierenden mußte für DAVE eine aktuelle Version der Java-Laufzeitumgebung einrichten.
- **Leichtgewichtiger Ansatz:** Die Idee eines leichtgewichtigen Modellierungswerkzeugs, das spezielle Akzente auf einfache Benutzerführung und einen an die Lehre angepaßten Funktionsumfang setzt, war erfolgreich. Die Oberflächengestaltung wurde als ansprechend empfunden (89% Zustimmung, 10% Ablehnung, 1% unentschlossen). Die Ausführungsgeschwindigkeit war offenbar weitgehend angenehm (79% Zustimmung, 21% Ablehnung).
- **Simulationsfunktion:** Die Simulationsfunktion liefert didaktisch wichtiges, frühzeitiges Feedback. Dies sahen auch die meisten Studierenden so (90% Zustimmung, 8% Ablehnung, 2% unentschlossen). Außerdem trägt sie nach Selbsteinschätzung der Studierenden zu einem besseren Verständnis des Formalismus bei (81% Zustimmung, 18% Ablehnung, 1% keine Angabe). Entsprechend wurde sie von allen Studierenden genutzt (75% ständig, 25% ein- oder mehrmals).
- **Multimediale Visualisierung:** Die multimediale Visualisierung der Simulation macht abstrakte Probleme anschaulicher (63% Zustimmung, 30% Ablehnung, 7% unentschlossen). Die gewählten Beispiele waren ansprechend (75% Zustimmung, 22% Ablehnung, 3% unentschlossen). Die Möglichkeit zur Visualisierung wurde von den meisten Studierenden angenommen (38% bei jeder Aufgabe, 49% ein- oder mehrmals, 10% niemals).

- **Gesamteindruck:** Insgesamt wurde das Werkzeug von den Studierenden positiv beurteilt. Die meisten würden das Werkzeug anderen Studierenden weiterempfehlen (75% Zustimmung, 25% Ablehnung). Ein sehr großer Teil der Studierenden wünscht sich ähnliche Werkzeuge für andere Themenbereiche der Vorlesung (86% Zustimmung, 14% Ablehnung).

Die Ergebnisse beziehen sich zunächst nur auf das Zustandsdiagramm-Werkzeug DAVE. Die Entwicklung des Architektur-Werkzeugs SAM war zwar im Sommersemester 2003 ebenfalls weitgehend abgeschlossen, aber das Werkzeug war nach Einschätzung der Entwickler noch nicht stabil genug, um damit einen ernsthaften Übungsbetrieb durchzuführen. Eine vollständige Evaluation von SAM wird erst im Frühjahr 2004 in Kleingruppen bzw. im Übungsbetrieb einer erneuten Iteration der oben erwähnten Veranstaltung durchgeführt werden.

Die bei der Evaluation von DAVE erzielten Ergebnisse sind jedoch bereits jetzt auf SAM übertragbar: Beide Werkzeuge besitzen in Form des zugrunde liegenden Framworks die gleiche technische Basis, so daß Aussagen über Benutzerschnittstelle, Verwendbarkeit und mögliche Fehler unmittelbar für alle Werkzeuge der Familie gelten. Insbesondere profitiert auch SAM von den im Anschluß an die Evaluation durchgeführten Korrekturen bzw. Erweiterungen. Da Zustandsdiagramme zudem auch inhaltlich eine Untermenge dessen darstellen, was von SAM abgedeckt wird – dort kommt noch die strukturelle Komponente hinzu –, gilt hier ebenfalls eine Übertragbarkeit der Ergebnisse.

DAVE wird im Wintersemester 2003/2004 in Dortmund erneut in einer Grundstudiumsveranstaltung zur UML-Modellierung eingesetzt. Hier nutzen etwa 350 Studierende das Werkzeug zum Lösen der Übungsaufgaben. Auch dieser Einsatz wird durch einen Fragebogen evaluiert werden.

Literatur

[ADE02] ALFERT, KLAUS, ERNST-ERICH DOBERKAT und GREGOR ENGELS: *Ergebnisbericht des Jahres 2002 des Projektes „MuSofT – Multimedia in der SoftwareTechnik“*, 2002. MuSofT-Bericht Nr. 2.

[APS03] ALFERT, KLAUS, JÖRG PLEUMANN und JENS SCHRÖDER: *A Framework for Lightweight Object-Oriented Design Tools*, 2003. Seventh ECCOP Workshop on Pedagogies and Tools for Learning Object-Oriented Concepts.

[APS04a] ALFERT, KLAUS, JÖRG PLEUMANN und JENS SCHRÖDER: *Eine Familie didaktischer Modellierungswerkzeuge für den Softwaretechnik-Unterricht*. In: FELLBAUM, KLAUS, KLAUS REBENSBURG und ANDREAS SCHWILL (Herausgeber): *Proceedings des zweiten Workshops Grundfragen multimedialer Lehre (GML)*, 2004.

[APS04b] ALFERT, KLAUS, JÖRG PLEUMANN und JENS SCHRÖDER: *Software Engineering Education Needs Adequate Modeling Tools*. In: HORTON, TOM und ANN SOBEL (Herausgeber): *Proceedings of the Seventeenth Conference on Software Engineering Education & Training (CSEE&T)*, 2004.

[BMR+96] BUSCHMANN, FRANK, REGINE MEUNIER, HANS ROHNERT, PETER SOMMERLAD und MICHAEL STAL: *Pattern-Oriented Sofware Architecture*. John Wiley and Sons, 1996.

[BRJ99] BOOCH, GRADY, JAMES RUMBAUGH und IVAR JACOBSON: *The Unified Modeling Language User Guide*. Addison Wesley Longman, 1999.

[DE01] DOBERKAT, ERNST-ERICH und GREGOR ENGELS: *Ergebnisbericht des Jahres 2001 des Projektes „MuSofT – Multimedia in der SoftwareTechnik“*, 2001. MuSofT-Bericht Nr. 1.

[GHJV96] GAMMA, ERICH, RICHARD HELM, RALPH JOHNSON und JOHN VLISSIDES: *Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, 1996.

[GS94] GARLAN, DAVID und MARY SHAW: *An Introduction to Software Architecture*. Internal Report CMU-CS-94-166, 1994.

[Har87] HAREL, DAVID: *Statecharts: A Visual Formalism for Complex Systems*. Science of Computer Programming, 8(3):231–274, June 1987.

[Hau04] HAUSTEIN, STEFAN: *KObjects Homepage*, 2004. http://www.kobjects.org (zuletzt gesichtet 19.01.2004).

[HN96] HAREL, DAVID und AMNON NAAMAD: *The STATEMATE Semantics of Statecharts*. ACM Transactions on Software Engineering and Methodology, 5(4):293–333, 1996.

[Kai01] KAISER, WOLFRAM: *Become a programming Picasso with JHotDraw*, 2001. http://www.javaworld.com/javaworld/jw-02-2001/jw-0216-jhotdraw.html (zuletzt gesichtet 19.01.2004).

[Lyo98] LYONS, ANDREW: *UML for Real-Time Overview*. 1998. http://www.rational.com/products/whitepapers/100463.jsp (zuletzt gesichtet am 06.01.2003).

[MGKS+04] METZ-GÖCKEL, SIGRID, MARION KAMPHANS, ELLEN SCHRÖDER, ANNA DRAG und ANJA TIGGES: *Evaluation des Projektes MuSofT – Multimedia in der SoftwareTechnik*, 2004. MuSofT-Bericht Nr. ??? (erscheint noch).

[Obj02] OBJECT MANAGEMENT GROUP: *Meta Object Facility (MOF) Specification 1.5*, 2002. http://www.omg.org/cgi-bin/doc?formal/2002-04-03 (zuletzt gesichtet 19.01.2004).

[Obj03] OBJECT MANAGEMENT GROUP: *Unified Modeling Language (UML) Specification 1.5*, 2003. http://www.omg.org/cgi-bin/doc?formal/03-03-01 (zuletzt gesichtet 19.01.2004).

[Pet62] PETRI, CARL ADAM: *Kommunikation mit Automaten.* Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[Ple03] PLEUMANN, JÖRG: *A Framework for Lightweight Graphical Modeling Applications*. In: CALLAOS, N., M. MARGENSTERN, J. ZHANG, O. CASTILLO und E.-E. DOBERKAT (Herausgeber): *SCI 2003 Proceedings*, Seiten 440 – 445, 2003.

[SGW94] SELIC, BRAN, GARTH GULLEKSON und PAUL T. WARD: *Real-Time Object-Oriented Modeling*. John Wiley and Sons, 1994.

[SH02] SLOMINSKI, ALEKSANDER und STEFAN HAUSTEIN: *XML Pull Parsing*. interChange, Seiten 13–17, December 2002.

[SR98] SELIC, BRAN und JIM RUMBAUGH: *Using UML for Modeling Complex Real-Time Systems*. 1998. http://www.rational.com/products/whitepapers/442.jsp (zuletzt gesichtet am 06.01.2003).

Lerneinheit Entwurfsmuster - Bericht 2003 MuSofT Teilprojekt 2.2

S. Seehusen, D. Irmscher-Lecon

Das inhaltliche und das didaktische Konzept der im Projekt MuSofT entwickelten Lerneinheit Entwurfsmuster und deren Umsetzung wird vorgestellt. Die in der technischen Realisierung verwendeten Werkzeuge zur Erstellung der verschiedenen Medienobjekte sowie die Medienobjekte werden beschrieben. Die Ergebnisse des ersten Teils der Evaluation der entwickelten Lehr-/und Lernmaterialien werden dargestellt.

Inhaltsverzeichnis

1 Einleitung

Entwurfsmuster sind ein wichtiges Konzept der Softwaretechnik. Sie dienen insbesondere der Vermittlung von Lösungen von häufig wiederkehrenden Entwurfs- und Implementierungsproblemen. Deshalb kommt diesem Lehrstoff große Bedeutung in der praktischen Softwaretechnik zu. Entwurfsmuster sollen didaktisch für die Präsenzlehre konzipiert und die Präsentation einzelner Muster sowie deren Eingliederung in ein größeres Softwareprodukt multimedial aufbereitet werden. Es werden die Phasen Entwurf und Implementierung abgedeckt. Die Beschreibung eines Entwurfsmusters besteht standardmäßig aus Text und Diagrammen sowohl für den statischen als auch für den dynamischen Aspekt. Gerade der dynamische Aspekt soll unter anderem mit Animationen visualisiert werden, um die Dynamik besser vermitteln zu können. Die Vermittlung dieser Aspekte wird durch herkömmliche Techniken nach unserer Erfahrung nicht hinreichend unterstützt und führt bei den Studierenden zu unnötigen Verständnisproblemen. Gerade die Überführung des Entwurfs in die Implementierung wird durch Einsatz von Hypermediaverweistechniken sichtbar und nachvollziehbar gemacht, in beide Richtungen: vom Entwurf zur Implementierung und zurück.

2 Vorstellung der Lerneinheit

In diesem Kapitel werden die Konzeption und die Struktur der Lerneinheit Entwurfsmuster vorgestellt.

2.1 Didaktisches Konzept und Lernziele

Das Ziel der Lerneinheit besteht in der Vermittlung des allgemeinen Konzeptes von Entwurfsmustern und in der Vermittlung von ausgewählten Entwurfsmustern. Ein wichtiges Lernziel besteht darin, dass die Studierenden selbständig weitere Entwurfsmuster bei Bedarf auswählen, sich aneignen und umsetzen können. Standardwerke, aber keine Lehrbücher, zu Entwurfsmustern sind unter anderem [EGV95] und [FBS96]. Es gibt weiterhin diverse Bücher, auch keine Lehrbücher, in denen weitere Entwurfsmuster beschrieben und kategorisiert sind, unter anderem [DSB00]. Daneben gibt es diverse Literatur zu programmiersprachspezifischen Anwendungen, zum Beispiel in [Gra98] oder [Coo00] und entsprechende Web-Sites.
Die Lerneinheit Entwurfsmuster soll eine allgemeine Einführung in Entwurfsmuster und in die Beschreibungsstruktur von Entwurfsmustern geben. Das Konzept von Entwurfsmustern

unterstützt den Entwurf, die Beschreibung und insbesondere auch die Diskussion von Softwareentwürfen und -implementierungen. Die Studierenden sollen befähigt werden, solche Diskussionen konstruktiv zu führen, die den Entwicklungs- und Qualitätssicherungsprozess fördern. Des Weiteren werden wichtige Kategorien von Entwurfsmustern wie Architektur-, Erzeugungs-, Struktur- und Verhaltensmuster eingeführt. Durch die Kenntnis von Entwurfsmustern sollen die Studierenden komplexe Bibliotheken wie zum Beispiel einige Java-APIs einfach nutzen können.
Zu diesen ausgewählten Kategorien werden repräsentative und häufig verwendete Entwurfsmuster präsentiert. Zu jedem Entwurfsmuster wird ein vollständiges Anwendungsbeispiel einschließlich Klassen und Sequenzdiagramm gegeben. Zudem wird der Quellcode in Java bereitgestellt, in dem Programmbeispiele, in denen Entwurfsmuster eingesetzt wurden, entworfen und implementiert wurden. Auch die Kombination von verschiedenen Entwurfsmustern wird anhand eines durchgängigen größeren Beispiels diskutiert.
Die Lerneinheit kann in das Studienfach Softwareentwicklung eingegliedert werden. Falls ein solches Fach explizit nicht vorgesehen ist, kann die Lerneinheit zum Studienfach Programmierung oder zum Studienfach Softwaretechnik gehören, je nach Aufteilung der Studienfächer im jeweiligen Curriculum. Die Lerneinheit stellt ein Bindeglied zwischen Softwaretechnik und Programmierung dar. Die in der Lerneinheit erworbenen Kenntnisse werden in der praktischen Softwareentwicklung angewendet, wie sie eine Softwareentwicklerin oder ein Softwareentwickler in der Anwendungsentwicklung oder in der Systementwicklung durchführt.

2.2 Individuelle kognitive und methodische Vorkenntnisse

Es werden grundlegende Kenntnisse in objektorientiertem Entwurf und objektorientierter Programmierung, einschließlich UML und Java, vorausgesetzt. Wird die Lerneinheit in der Weiterbildung eingesetzt, wird die Fähigkeit zum Selbststudium vorausgesetzt.

2.3 Lernszenario

Die erstellten Lernmodule können im Wesentlichen in zwei unterschiedlichen Szenarien eingesetzt werden.

Präsenzlehrveranstaltung

Im ersten Szenario werden die erstellten Materialien, insbesondere die Animationen, in der Präsenzlehrveranstaltung zur Erläuterung der Lehrinhalte eingesetzt. In der Übung und in einer seminaristischen Vorlesung werden besonders die interaktiven Elemente der Animationen eingesetzt, um in Rückkopplung mit den Studierenden bestimmte Elemente zu vertiefen. Des Weiteren werden die Lernmodule zur Nachbereitung und zum Nachschlagen bei der Bearbeitung von Aufgaben von den Studierenden eingesetzt. Es werden somit Vorlesung, Übung und Praktikum explizit unterstützt. Ein Teil der Lernmodule wird zum Beispiel in Lübeck im 4. Semester in der Lehrveranstaltung „Programmiertechniken“ der Studienrichtung Informatik des Studiengangs Elektrotechnik zum Thema Entwurfsmuster eingesetzt. Ein weiterer Teil der Lernmodule kann zum Beispiel in einer entsprechenden Vertiefungsveranstaltung angeboten werden.

Lernszenario Weiterbildung

Im zweiten Lernszenario dienen die erstellten Materialien zur Weiterbildung in einer Kombination von Selbststudium, Online-Betreuung und Präsenzseminar. Die Animationen dienen der Erläuterung der beschreibenden Texte. Die Hypertextstrukturen werden insbesondere zur Auswahl der stofflichen Inhalte eingesetzt, die von besonderem Interesse für die lernende Person ist. Da es pro präsentiertem Entwurfsmuster eine ausführliche Einführung und Beschreibung sowie im Referenzteil eine konzentrierte Beschreibung gibt, kann die lernende Person je nach Vorkenntnissen die entsprechende Präsentation auswählen. Ein Teil der Lehr-/Lernmaterialien wird zum Beispiel im Online-Studiengang Medieninformatik im Kurs „Objektorientierte Programmierung mit Java“ an der FH Lübeck eingesetzt.

2.4 Lehr-/Lernmaterial

Das Lehr- und Lernmaterial besteht für die Vorlesung aus Folien, Texten, Bildern, Grafiken und Animationen. Einige Beispielanwendungen bieten auch interaktive Elemente zur Demonstration von Nutzungsabläufen. Im Lehr- und Lernmaterial sind jeweils einfache Tests zur Lernfortschrittskontrolle integriert, deren Antworten vom Rechner selbst ausgewertet werden können. Diese Tests werden in der Regel von jeweils einer Einzelperson durchgeführt. Des Weiteren werden größere Projekt-Aufgaben angeboten, die in der Regel eine konstruktive Entwicklungsarbeit umfassen und eine Gruppenarbeit von Studierenden erfordern. Dies ist ein Lernarrangement, das für Studierende mit Vorkenntnissen ab dem 4. Semester geeignet ist [BS03].

2.5 Die Lerneinheit Entwurfsmuster

Die Lerneinheit wird in 5 Teile gegliedert, die jeweils aus Lernmodulen bestehen. Die zu lernenden Inhalte werden in den ersten zwei Teilen inhaltlich aufeinander aufgebaut und sind stark mit dem Referenzteil und auch mit der zusammenhängenden Beschreibung des durchgängigen Beispiels vernetzt. Vom Referenzteil gibt es entsprechende Rückverweise auf die einführenden Teile. In den einzelnen Teilen, insbesondere im Teil Ausblick, wird auf vertiefende Inhalte verwiesen, die teilweise auch im WWW verfügbar sind. Insbesondere soll auch auf andere Lernmodule verwiesen werden, die im Projekt MuSofT entwickelt wurden.

2.5.1 Teil 1: Einführung in Entwurfsmuster

- Allgemeine Einführung
- Einführendes Entwurfsmuster

Anhand eines ausgewählten Entwurfsmusters, Filter (Pipes-and-Filters), wird das Konzept, die Darstellungsmethode, das Beschreibungsschema und die UML-Notation von Entwurfsmustern eingeführt

2.5.2 Teil 2: Entwurfsmuster

Es wird eine Auswahl weiterer Entwurfsmuster vorgestellt, wobei jeweils eins oder mehrere aus verschiedenen Kategorien wie Architektur-, Erzeugungs-, Struktur- und Verhaltensmuster gewählt werden. Ein weiteres Auswahlkriterium ist die Verwendung für interaktive Programme.
Jedes Entwurfsmuster wird in einem eigenen Lernmodul dargestellt. Folgende Muster werden behandelt:

- Filter
- Singleton
- Strategie (strategy)
- Beobachter (observer)
- MVC
- Adapter
- Brücke (bridge)
- Kompositum (composite)
- Befehl (command)
- Iterator

Zu der Beschreibung jedes Entwurfsmusters gehört:

- ein Einführungsbeispiel,
- Beschreibung nach dem eingeführten Beschreibungsschema,
- Klassendiagramme, Sequenzdiagramme, evtl. Kollaborationsdiagramme, evtl. Zustandsdiagramme,
- Animationen, orientiert an Objekt- und Sequenzdiagrammen,
- ein Anwendungsbeispiel (mit konkreten Klassendiagrammen), das komplett (in Java) verfügbar ist und das möglichst ein Teil des durchgängigen Beispiels aus Teil 3 darstellt,
- wenn vorhanden, die Verwendung des Entwurfsmusters in einer Java-API, und
- Aufgaben.

2.5.3 Teil 3: Durchgängiges Beispiel

So weit wie möglich wird ein durchgängiges Beispiel zur Demonstration der Anwendung der Entwurfsmuster einschließlich Implementierung in Java eingesetzt.

2.5.4 Teil 4: Ausblick

Im Teil 4 folgt ein Überblick über weiterführende Verweise und Literatur. Ein Lernziel besteht darin, dass die Studierenden lernen, selbständig weitere Entwurfsmuster bei Bedarf auswählen, sich aneignen und umsetzen können.

2.5.5 Teil 5: Referenzteil

Während in Teil 2 die einzelnen Entwurfsmuster einführend beschrieben werden, wird in diesem Referenzteil jedes Entwurfsmuster noch einmal nach einem festen Beschreibungsschema kurz beschrieben. Durch den Referenzcharakter ist es kein Lernmodul im engeren Sinne, es ist jedoch ein Bestandteil der Lerneinheit und kann insbesondere auch zur Wiederholung oder zur Auswahl von Entwurfsmustern genutzt werden.

2.5.6 Aufgaben

Aufgaben sollen die Studierenden zum aktiven Lernen motivieren. Einige Aufgaben dienen der Selbsteinschätzung der Studierenden (Lernfortschrittskontrolle), andere Aufgaben dienen als Anreiz, sich tiefer mit dem Stoff zu beschäftigen. Die Aufgaben können zum Beispiel wiederholend sein, es können Analyseaufgaben sein, sie können eine Reflexion des Stoffes und/oder die Konstruktion einer eigenen Lösung eines vorgegebenen Problems (die Aufgabe im engeren Sinne) erfordern. In allen Lernmodulen aus Teil 1 und 2 werden Aufgaben verschiedener Art gestellt. Es werden sowohl geschlossene Fragen gestellt, bei denen die richtige Lösung unter falschen angeboten wird, als auch offene Fragen, bei denen die Studierenden selbständig die Antworten finden und frei beschreiben muss.

2.6 Anforderungen an die Lehr/Lernumgebung

Studiengang, Abschnitt des Studiengangs: Die Lerneinheit ist für einen Studiengang Informatik geeignet, wobei es hier auch anwendungsorientierte Studiengänge wie zum Beispiel Medieninformatik oder auch Studienrichtungen wie zum Beispiel Informatik in der Elektrotechnik sein können, in denen das Erlernen der Programmierung größerer Softwarepakete zum Studienumfang gehört.

Art der Ausbildungsinstitution: Die Lerneinheit kann an einer Universität oder einer Fachhochschule im 3. bis 5. Semester angesiedelt werden, je nach Vermittlung der Vorkenntnisse. Sie eignet sich auch für eine Weiterbildung für Personen, die über entsprechende Vorkenntnisse verfügen und eine Einführung in Entwurfsmuster genießen wollen.
Die Lehr- und Lernumgebung besteht aus einer verteilten Umgebung, die aus verschiedenen Mengen von Werkzeugen bestehen kann, die jeweils eine unterschiedliche Nutzung unterstützen.

Lernraum Für den Einsatz der Lerneinheit sollte pro Institution, die die Lerneinheit anbietet, ein Lernraum oder der Zugang zu einem solchen gemeinsamen Lernraum eingerichtet werden. Der Lernraum stellt die Lerneinheit den Lehrenden und Lernenden zur Verfügung

und unterstützt allgemein des Lehren und Lernen mit der Lerneinheit. Diese Funktion kann zumindest teilweise von der in MuSofT entwickelten Plattform übernommen werden. Es müssen aber pro Lerngruppe auch spezifische Arbeitsumgebungen zum Beispiel zur Unterstützung der virtuellen Gruppenarbeit (zum Beispiel durch BSCW) zur Verfügung gestellt werden. Die Auswahl der Werkzeuge ist kontextabhängig und sollte vorhandene Kenntnisse und eine einfache Integration in den Studienablauf mit einbeziehen.

Arbeitsumgebung der Lehrenden Zur Arbeitsumgebung der Lehrenden gehören alle Werkzeuge, um die einzelnen Lernmodule und Medienobjekte anzuzeigen, sowohl im Lehrveranstaltungsraum als auch auf einem Rechnerarbeitsplatz. Im Lehrveranstaltungsraum wird eine hinreichende Projektionsmöglichkeit (Beamer, Leinwand, Lautsprecher) mit Laptop (oder Standrechner) mit einem Anschluss ans Campusnetz vorausgesetzt. Als Präsentationswerkzeuge sollten Web-Browser, zum Beispiel Netscape, mit entsprechenden Plugins für Animationen, Audio, Video und evtl. 3D ausreichen. Zur Lehrumgebung gehört auch eine relativ einfache Möglichkeit, die Lerneinheit zu aktualisieren.

Arbeitsumgebung der Studierenden Für die Teile der Lerneinheit, die zum Selbststudium geeignet sind, und für die Lösung der Aufgaben wird den Studierenden eine Arbeitsumgebung empfohlen, die unter anderem folgende Werkzeuge enthält:

- Web-Browser, zum Beispiel Netscape, mit einigen Standard-Plugins, mit Java-Plugin,
- UML-Editor, möglichst mit XMI-Ausgabeformat, zum Beispiel Poseidon,
- JDK 2,
- Linux empfohlen, Windows auch möglich,
- HTML-Editor für Aufschreiben von Lösungen der Aufgaben,
- evtl. Softwareentwicklungsumgebung wie zum Beispiel Together,
- Werkzeuge zur asynchronen Kommunikation empfohlen wie email, news und Bulletin Board und
- Werkzeuge zur synchronen Kommunikation empfohlen wie zum Beispiel chat, Netmeeting, CUseeMe und elektronische WhiteBoards.

Für jeden Werkzeugtyp muss es mindestens ein Werkzeug geben, das entweder public domain oder dessen Nutzung zumindest für Ausbildungszwecke kostenfrei ist. Die Arbeitsumgebung sollte in entsprechenden Rechner-Pools an der Hochschule zur Verfügung stehen. Die Arbeitsumgebung sollte aber auch einfach auf dem heimischen Rechner ausführbar sein. Das zeitlich und räumlich asynchrone Arbeiten der Studierenden soll explizit unterstützt werden.

3 Technische Realisierung

Zur Vermittlung der Grundlagen von Entwurfsmustern zur Unterstützung des Entwurfs und der Implementierung von objektorientierten Systemen kommen folgende Lehr- und Lernmaterialien zum Einsatz:

- Einsatz von Text, Grafiken, Bildern, Animationen und Verweistechniken,
- Visualisierung der statischen Aspekte durch Komponenten-, Klassen- und Sequenzdiagramme,
- Visualisierung der dynamischen Aspekte durch Animationen (siehe z.B. Abbildung 4),
- unterschiedliche Lehr- und Lernansichten des Lehrmaterials entsprechend den unterschiedlichen Anforderungen der jeweiligen Nutzer und Nutzerinnen (Lehrende/Studierende),
- interaktive Aufgaben zum Entwurf von objektorientierten Systemen mit Entwurfsmustern und
- verfügbare Lernmaterialien im Online/Offline-Format, sowie als Druckversion.

Folgende Techniken werden bei Entwicklung und Einsatz der Materialien verwendet:

- Beschreibung der Struktur und der Integration der Medienobjekte in XML (L3- XML),
- Generierung der Lehr- und Lernansichten durch ein Werkzeug,
- Generierung von Übersichtsdiagrammen zur Unterstützung der Navigation,
- Navigationshilfen durch Integration einer Volltext- und Metadatensuche und
- Bereitstellung unterschiedlicher Lernpfade für die benutzerspezifische Navigation.

Bei der Entwicklung der Lerneinheit werden eine Reihe von Techniken und Werkzeugen eingesetzt. Die Entwicklung von multimedialen Einheiten erfordert zur Zeit immer noch, dass für unterschiedliche Medien unterschiedliche Werkzeuge verwendet werden müssen. Darüber hinaus müssen für einige Medien je nach Einsatzeigenschaften verschiedene Werkzeuge eingesetzt werden. Sobald firmeneigene Formate verwendet werden, ist die Nachhaltigkeit nicht gewährleistet.

3.1 Entwicklungsprozess

Für die Realisierung der Lerneinheit wurde ein Entwicklungsprozess weiterentwickelt, der in Abbildung 1 dargestellt wird. Die Entwicklung der Lerneinheit ist eine iterative Entwicklung, die dennoch in Phasen strukturiert werden kann [SL01]. Die Abbildung 1 zeigt, wie die Phasen Konzept, Entwurf, Realisierung (Implementierung), Test, Einsatz und Aktualisierung zueinander in Beziehung stehen. In den einzelnen Phasen werden die Entwicklungsdokumente erstellt,

wie in der Abbildung skizziert. Zur Beschreibung des Konzeptes gehören das inhaltliche Konzept (Stoffplan, Methoden, Kompetenzen), die didaktischen Vorüberlegungen (Leitbild, Lernziele (des gesamten Kurses), Lernszenarien und didaktischer Ansatz) sowie die Festlegung allgemeiner Eigenschaften (z.B. Nachhaltigkeit, Plattformneutralität). Der Entwurf kann aus einem Drehbuch bestehen oder nach einer schrittweisen Verfeinerung der Lernmodule beschrieben werden. Die Realisierung umfasst die Texte und die weiteren Medienobjekte sowie deren Integration. Es werden in der Regel verschiedene Tests durchgeführt, die entsprechend geplant, durchgeführt und dokumentiert werden. Während des Einsatzes wird die Lerneinheit wiederholt in verschiedenen Lehrveranstaltungen genutzt. Eine begleitende Evaluation wird durch entsprechende Logs und durch statistische Auswertung unterstützt.

Nach oder sogar während jeden Durchlaufes wird der Kurs in der Regel aktualisiert. Zur Aktualisierung gehört neben der Korrektur von Fehlern, Verbessern der Darstellung auch die Aufnahme von neuen Forschungs- und Entwicklungsergebnissen aus dem vermittelten Fachgebiet. Die Aktualisierung kann im Extremfall eine fast komplette Neuentwicklung beinhalten.

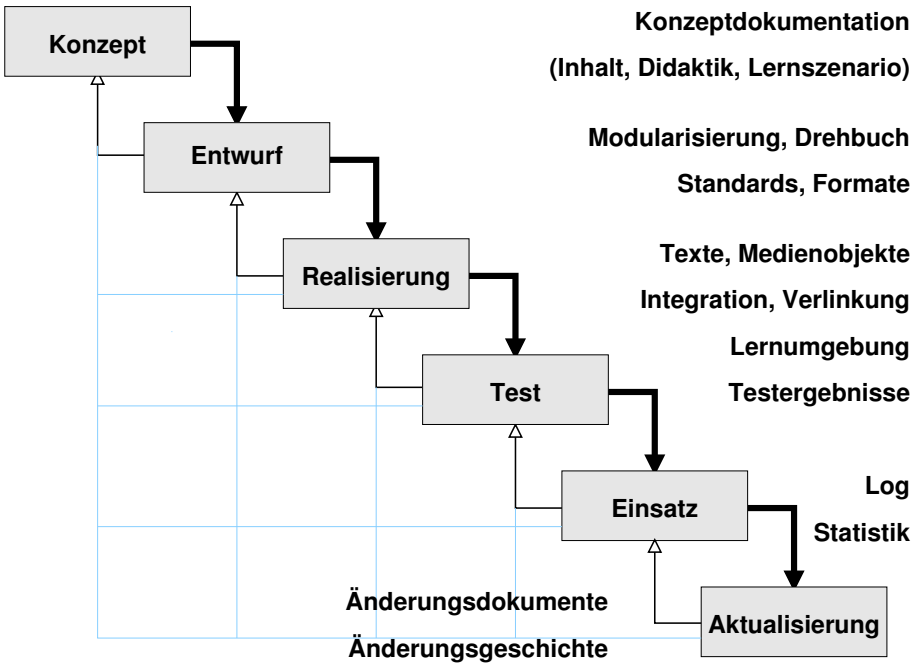


Abbildung 1: Entwicklungsprozess.

3.2 XML-Produktionsprozess

Um die Entwicklung durchgängig zu unterstützen, werden alle Entwicklungsdokumente in einem Dokument vereinigt. Ein Entwicklungsdokument stellt dann eine spezielle Sicht des Gesamtdokumentes dar. Zur Spezifikation der Struktur der Lerneinheit wird XML [W3C] verwendet, um insbesondere auch die Interoperabilität von verschiedenen Werkzeugen zu unterstützen. Es wurde L3-XML [SLK00], eine XML-DTD, weiterentwickelt, die die einzelnen Phasen der Entwicklung und des Einsatzes unterstützt. Die einzelnen Medienobjekte wie Text, Bild, Animation, Ton und Interaktion werden durch Verweise bzw. Attribute in die Struktur eingeordnet. Aus einer L3-XML-Beschreibung wird mit dem an der FH Lübeck entwickelten Werkzeug xml2html [SL01] eine Menge von vernetzten HTML-Seiten generiert. In Abbildung 2 wird der Produktionsprozess der Lerneinheit dargestellt.

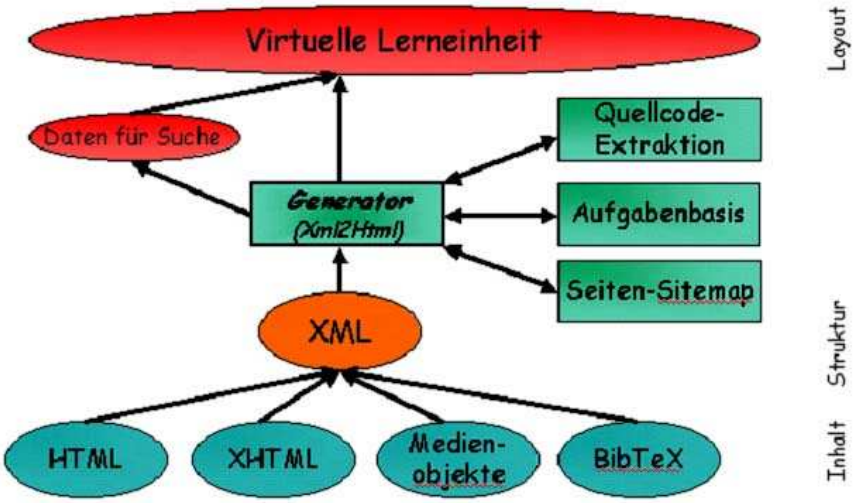


Abbildung 2: Produktionsprozess

3.3 Lehr- und Lernansicht

Aufgrund der bisherigen Erfahrungen mit den ersten Kapiteln der Lerneinheit im WS 2001/02 wurden mehrere Sichten definiert. Insbesondere wurden eine Lehr- und eine Lernansicht von den Funktionsmöglichkeiten und dem grafischen Entwurf festgelegt. Die Lehransicht ist die Ansicht zur Präsentation der Lehr-/Lerninhalte in einer Vorlesung, Übung oder Praktikum, siehe z.B. Abbildung 3. Diese Sicht ist für den Einsatz von Lehrenden konzipiert.
Die Lernansicht ist die Ansicht, die den Studierenden zum Nachschlagen, Vertiefen oder generell zum selbstorganisierten Studium bereitgestellt wird, siehe z.B. Abbildung 4. Die Anforderungen an diese Sichten unterscheiden sich sehr stark, obwohl die dargestellten inhaltlichen Konzepte gleich sind. Neben Schriftgrößen, Formatierung, Marginalien liegt der wesentliche

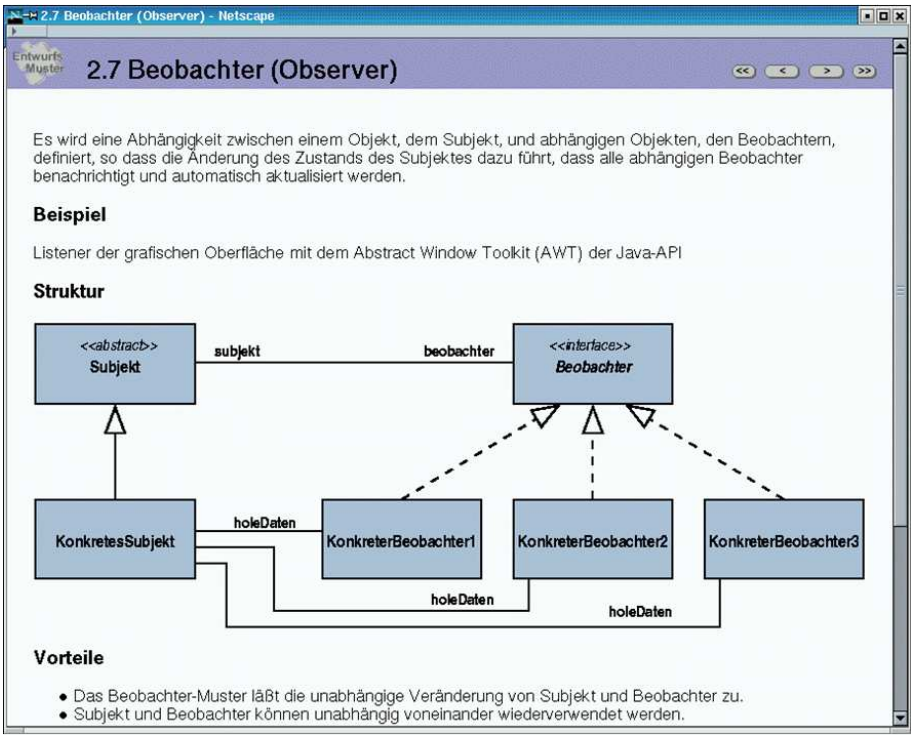


Abbildung 3: Lehransicht

Unterschied in dem Ausblenden von Details in der Lehransicht, den unterschiedlichen Navigationsmöglichkeiten, sowie dem Zugang zur Navigation und deren Darstellung.

3.4 Navigationen und Lernpfade

In der Lerneinheit werden verschiedene Arten der Navigation bereitgestellt, die durch die Erstellung der Struktur, Integration in XML und die Angabe von Metadaten ermöglicht werden. Aus diesen Informationen werden in den jeweiligen Sichten Verweisstrukturen und Verweislisten generiert.
Zu den Navigationsmöglichkeiten gehören:

- Inhaltsverzeichnis und Medienverzeichnisse: Die Verzeichnisse, die uns aus gedruckten Dokumenten bekannt sind, werden auch in einer Lerneinheit zur Verfügung gestellt und durch interaktive Verweismöglichkeiten ergänzt. Zu den Verzeichnissen gehören das Inhaltsverzeichnis, Aufgabenverzeichnis und Medienverzeichnisse von z.B. Applets.

- Lernpfade: Die Definition von verschiedenen Lernpfaden (Lernpfade siehe [SLK00]) wird in L3-XML explizit angegeben. Der Standardlernpfad (empfohlene Lernsequenz)

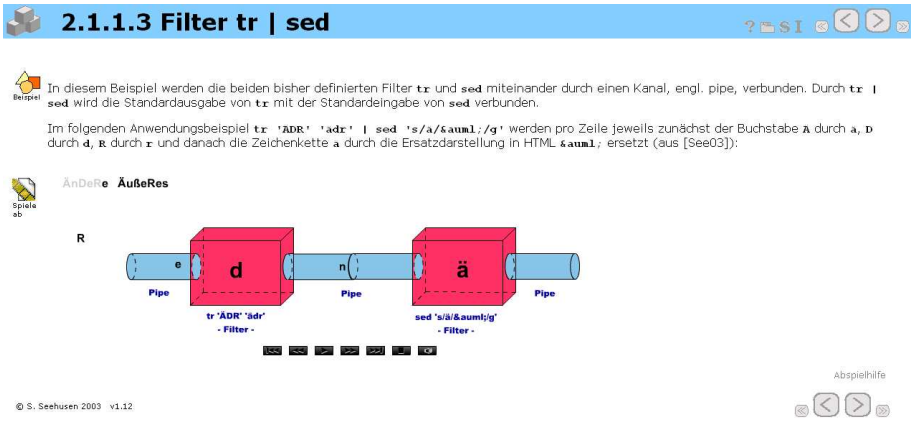


Abbildung 4: Lernansicht

wird hervorgehoben. In der Lehrversion entspricht diesem Lernpfad die normale Reihenfolge der Präsentation.

- Positionsübersicht: Zur Unterstützung der Navigation und der Übersicht der aktuellen Position in der Lerneinheit wird eine Übersichtskarte über die ganze Lerneinheit mit entsprechenden Verweisen in einer sensitiven Map auf jeder Seite dargestellt.

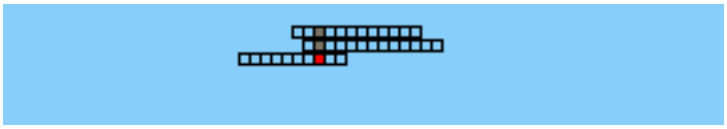


Abbildung 5: Übersichtskarte

Die Karte in Abbildung 5 zeigt z.B. an, dass man sich im 3. Kapitel, 2. Abschnitt, 8. Unterabschnitt befindet. Es wird visualisiert, wo in Kapitel oder Abschnitt man sich befindet, wieviel hinter und wieviel vor einem liegt. Die Karte ist auch sensitiv, so dass sie zum Springen in einen anderen Abschnitt dienen kann. Insbesondere werden dadurch weite Sprünge unterstützt, ohne über das Inhaltsverzeichnis gehen zu müssen.

- Glossar: Innerhalb der Lerneinheit werden Verweise auf das Glossar bei den einzelnen Begriffen hinterlegt.

- Suchfunktion: Über eine Suchfunktion [LS02], die in Java als Applet implementiert ist, wird eine Volltextsuche auf der Lerneinheit und eine Suche nach den Metadaten zur Verfügung gestellt.

3.5 Entwicklungswerkzeuge

Für die Beschreibung der Entwurfsmuster werden im Wesentlichen die Medien Text, Bilder, Animationen, Video und Audio verwendet, wobei der Schwerpunkt auf Text, Bilder, Animationen und Verweise gelegt wird. Es wurden Entwicklungswerkzeuge für die einzelnen Medien ausgewählt, die möglichst herstellerneutrale Formate oder Formate für die einzelnen Medien erzeugen können, so dass die Medienobjekte mit Standard-Werkzeugen präsentiert werden können.
Für die einzelnen Medien wurden verschiedene Werkzeuge eingesetzt. Für Texte, die in XHTML geschrieben werden, wurde im wesentlichen emacs eingesetzt. Für Bilder wurden Gimp, xfig und Illustrator eingesetzt, je nach Art des Bildes. Wichtig ist dabei, dass das erstellte Bild in der Ansicht ohne großen Qualitätsverlust skalierbar ist. Als Formate eignen sich dazu Vektorgrafikformate wie svg, PostScript und, eingeschränkt, JPEG. Da in einer Lerneinheit genormte grafische Beschreibungsmittel wie z.B. Klassendiagramme oft in Bildern vorkommen, muss die Darstellung von z.B. Klassen in allen Bildern sehr ähnlich sein. Deshalb muss dazu eine Art Bildbibliothek angelegt werden. Die Animationen wurden mit Flash oder mit einem svg-Editor erstellt. Eine Animation soll von dem Betrachtenden gesteuert werden können. Dazu wurde eine entsprechende Bedienungsleiste realisiert, die zu jeder Animation zur Verfügung gestellt wird. Der Entwicklungsaufwand für Animationen mit Flash ist relativ hoch. Deshalb werden zukünftige Animationen in svg erstellt.

Entwicklungswerkzeuge für Text

Das Medium Text ist nach wie vor das meist verwendete Medium in einer Lerneinheit. In der Lerneinheit Entwurfsmuster werden hauptsächlich die Ausgabeformate XHTML, PDF und PostScript verwendet. Diese Formate sind zurzeit weit verbreitet. Die Studierenden benötigen zur Betrachtung dieser Texte einen XHTML-fähigen Browser, den Acrobat-Reader und eventuell GhostView. Diese Programme sind kostenlos. PDF und PostScript sind für Druckversionen vorgesehen, die eine notwendige Ergänzung jeder Lerneinheit sind. Zur Erstellung der Texte werden Standardeditoren wie zum Beispiel xemacs verwendet. Des Weiteren werden Acrobat und LaTeX zur Erzeugung der PDF- und PostScript- Dateien verwendet. Zur Integration der Medienobjekte zu der Lerneinheit wird xml2html eingesetzt (siehe oben).

Entwicklungswerkzeuge für Bilder

Bei der Erstellung der Bilder wurden mehrere Entwicklungswerkzeuge verwendet, unter anderem die Macromedia-Produkte Freehand 10 und Fireworks 4 und xfig. Freehand 10 dient der Erstellung von vektorbasierten Illustrationen für die spätere Integration in Flash, um sie dort in Animationen weiterzuverarbeiten oder direkt als Bilder im gif-, jpeg- oder png-Format zu verwenden. Ebenso werden Graphiken mit Fireworks 4 erstellt, bearbeitet und animiert. Grafiken können optimiert und mit erweiterten Interaktivitätselementen versehen werden.
Für die Bearbeitung von Photos wird Photoshop von Adobe verwendet, das für Photobearbeitung gut geeignet ist. Die Studierenden benötigen für die Darstellung der Bilder selbst keine zusätzlichen Werkzeuge, außer den ohnehin verwendeten XHTML-fähigen Browser.

Entwicklungswerkzeuge für Animationen

Für die Erstellung von Flash-Animationen wurde Flash 5 verwendet. Für die Entwicklung

von svg-Animationen wird ein Texteditor verwendet. Zur Unterstützung der Entwicklung von Objektdiagrammanimationen wird ein einfaches Werkzeug entwickelt, das in solchen Diagrammen wiederkehrende Elemente bereitstellt.

3.6 Anmerkungen in Programmtexten

Ein wichtiges Merkmal der Lerneinheit besteht auch darin, dass zu einem Entwurfsmuster auch jeweils ein lauffähiges Beispiel in Java präsentiert wird. Die Präsentation muss neben z.B. einem Klassendiagramm auch fast immer die Darstellung von Programmtext vorsehen, da nur so eine konkrete Umsetzung gezeigt werden kann. Da Programmtexte von Beispielen für die Anwendung von Entwurfsmustern relativ lang sind, wird oft auf erklärende Kommentare im dargestellten Text verzichtet.

Um jedem Studierenden die Möglichkeit zu geben, Details der Erklärung des Programmtextes leicht nachzuschauen, wurde das Werkzeug acceNun im Teilvorhaben 2.2 entwickelt, das aus dem Programmtext eine Ansichtsversion generiert, in denen markierte Texte maus-sensitiv mit erklärendem Text hinterlegt werden können (siehe z.B. Abbildung 6). Das Werkzeug generiert aus einem Java-Programmtext eine XHTML-Seite mit JavaScript-Funktionsaufrufen. Auf diese Seite kann zur farbigen Markierung der Java-Schlüsselworte das Werkzeug Java2Html angewendet werden [Geb], das als Open source unter der GNU General Public License (GPL) verfügbar ist, so dass eine visuell ansprechende Darstellung des Quelltextes entsteht. Diese Darstellung kann in HTML- Seiten eingebunden werden. Das Werkzeug acceNun kann eigenständig oder aus einem anderen Programm wie bei der Generierung der vernetzten HTML-Seiten aufgerufen werden. Da der Text, der Programmteile erklärt, direkt im Programmtext als Kommentar eingefügt ist, wird eine konsistente Aktualisierung von Programm und Dokumentation unterstützt, so wie prinzipiell auch die JavaDoc-Kommentare diese Konsistenz unterstützen.

3.7 Interaktive Aufgaben

Auf dem Gebiet der Entwurfsmuster gibt es diverse Aufgaben, die darin bestehen, einen vorgegebenen Software-Entwurf, der als Klassendiagramm dargestellt ist, gemäß der Aufgabenstellung zu ergänzen und zu verändern. Um diese Art der Aufgaben für die Studierenden direkt am Rechner bearbeitbar und dann vom System überprüfbar zu machen, wurde im Teilvorhaben 2.2 ein Klient/Serversystem X2ex zur Unterstützung dieser Aufgaben entworfen und implementiert. Das System stellt für den Autor und die Autorin einen Editor für einfache Entwurfs-Aufgaben zur Verfügung, mit dem die Aufgabenstellung, das vorgegebene Klassendiagramm und die möglichen Lösungen erstellt werden können. Die Klassendiagramme werden in einem vereinfachten UML-Editor erstellt, wobei der Umgang und das Verhalten des Editors wie z.B. in Poseidon [Gen04] gestaltet ist. Die Aufgaben und die Lösungen werden in XML abgelegt. Bei der Präsentation einer Aufgabe gibt die lernende Person die Lösung an, indem sie wie mit einem einfachen UML-Editor das vorgegebene Klassendiagramm verändert. Die Lösung wird dann über RMI am Server, von dem die Aufgabe geladen wurde, überprüft.

Abbildung 6: Mouse-over-Annotation.

3.8 Lehr- und Lernmittel

Aus Mitteln des Projektes wurden für den Ersteinsatz als elektronische Lehr- und Lernmittel SmartClass und ein SmartBoard beschafft und der Einsatz begleitet.

SmartClass

Das SmartClass-System besteht aus einer zuverlässigen Schalttechnologie und verbindet jeden Rechner in einem Ausbildungsrechnerraum mit einer zentralen Kontrolleinheit, die vom Lehrenden bedient wird. Die Kontrolleinheit hat eine Matrixanordnung von Stationstasten, die es dem Lehrenden auf einfache Art erlaubt, den Systemstatus zu überwachen, Studierenden-Stationen auszuwählen und die Bildschirmausgabe und auch die Tastatur- und Mauseingabe zu übernehmen. Insbesondere ist auch eine Projektion eines Bildschirms auf alle anderen Bildschirme möglich. Im normalen Praktikumsbetrieb hat sich das System sehr gut bewährt, weil es für die Lehrenden intuitiv und leicht erlernbar ist. Eine kurze Demonstration der Bedienung von wenigen Minuten reichte. Da das System den Praktikumsbetrieb sehr gut unterstützt, wurde es von den verschiedenen Lehrenden, die den Praktikumsraum für Lehrveranstaltungen nutzen, eingesetzt.
Aus Projektmitteln wurden nur 6 Plätze mit SmartClass ausgestattet, um die Einsatzmöglichkeiten erst einmal auszutesten. Aufgrund der positiven Erfahrungen wurde auch die übrigen Rechner im Praktikumsraum aus Hochschulmitteln 2002 um SmartClass ergänzt.

SmartBoard

SmartBoard ist ein interaktives Whiteboard, das die Präsentation und die Interaktion von An-

wendungen direkt am Whiteboard erlaubt, ohne dass der Dozent oder die Dozentin sich hinter einem Rechner verstecken muss. Da wir eine mobile Version des SmartBoard gewählt haben, muss vor jedem Einsatz die Entfernung zum Beamer kalibriert werden. Eine solche Rüstzeit von ca. 5 Minuten ist für den Regelbetrieb nicht geeignet. War das SmartBoard einmal eingerichtet, so haben sich in einem Versuch und auch in Projektpräsentationen die Studierenden sehr schnell die Funktionalitäten erarbeitet und genutzt. Zur besseren Ausnutzung wurde aus Hochschulmitteln ein Beamer mit deutlich höhere Auflösung beschafft und es ist geplant, die Rüstzeit durch eine veränderte technische Organisation deutlich zu verkürzen.

4 Evaluation

Die entwickelten Lehr-/Lernmaterialien werden seit zwei Semestern an der Fachhochschule Lübeck im 4. Semester in der Lehrveranstaltung „Programmiertechniken“ der Studienrichtung Informatik des Studiengangs Elektrotechnik eingesetzt. In der Präsenzveranstaltung waren neben den entwickelten Lernmodulen auch Programmierung in Java für C++-Kundige, Parallelprogrammierung und Komponententechnologie Gegenstand der Lehrveranstaltung. Den inhaltlichen Kern des ersten Teils der Lehrveranstaltung bilden allgemeine Programmiertechniken, und hier insbesondere die Techniken, die als Entwurfsmuster beschrieben sind. Im Online-Kurs „Objektorientierte Programmierung mit Java“ im 4. Semester des Online-Studienganges kommt einige Lehr-/Lernmaterialien ebenfalls seit zwei Semestern zum Einsatz.
In beiden Semestern (Sommersemester2003 und Wintersemester 2003/2004) wurde eine Evaluation der im Projekt MuSofT entwickelten Materialien durchgeführt.
Die interessierenden Evaluationsinhalte waren der Einsatz digitaler Medien in der Lehre (Animationen, digital präsentierte Folien, Online-Hilfen, Online-Skripte, Online-Beispiele). Entsprechend einem im Vorfeld entwickelten Evaluationskonzept sollten Mehrfachbefragungen mit Fragebogen sowie Interviews die Haupterhebungsmittel sein. Über die Mehrfachbefragung sollte das Umfeld der Studierenden, Arbeitsweisen und Einstellungen erfasst werden, um darüber Informationen über den Nutzen von digitalen Medien in den Lehrmaterialien zu ermitteln.
In beiden Lehrveranstaltungen sollten jeweils drei Fragebogen zum Einsatz kommen, am Anfang, in der Mitte und am Ende des Semesters. Am Ende des Semesters sollte es eine Gruppendiskussion zum Thema geben, um die bereits durch die Fragebogen erhobenen Ergebnisse zu ergänzen und zu vervollständigen.

4.1 Entwicklung der Fragebögen

Für die Durchführung der Befragungen wurden als erstes die Fragebögen entwickelt, die als Online-Fragebogen am Rechner ausgefüllt und abgeschickt werden sollten. Die Entscheidung für einen Online-Fragebogen fiel, da zum einen eine höhere Akzeptanz beim Ausfüllen seitens der Befragten angenommen wurde (Ausfüllmotivation). Zum anderen wird durch den Einsatz dieser Technik die Datensammlung vereinfacht:
Die Daten liegen in digitaler Form vor, was die Weiterverarbeitung erleichtert. Ein weiterer Vorteil liegt in der Nachhaltigkeit der Fragebögen: einmal entwickelt können die Fragebögen

immer wieder eingesetzt werden. Über eine Verschlüsselung personenidentifizierender Daten ist es möglich, die abgegebenen Fragebögen der drei Befragungszeitpunkte wieder denselben Personen zuzuordnen, sofern dieselben Nummerncodes (wie z.B. Matrikelnummer) eingegeben wurden.

Das Layout wurde für alle entwickelten Fragebögen einheitlich gehalten. In den Fragebögen wurden unterschiedliche Fragestile eingesetzt: Es gab einfache Ankreuzfragen („Haben Sie einen Rechner“), offene Fragen in denen ein freier Text eingegeben werden konnte und Rating-Fragen (siehe z.B. Abbildung 7), bei denen die Befragten auf einer 5-stufigen Skala Angaben zu ihrer Einstellung machen sollten.

Abbildung 7: Fragebogen, Rating-Fragen

Die einzelnen Fragen waren teilweise zu allen drei Befragungszeitpunkten identisch. Dies sollte die Möglichkeit geben, eine Einstellungsänderung über das Semester zu erfassen, auf der einen Seite über alle Studierenden gesehen, aber auch bezogen auf die Angaben einer einzelnen Person. Unterschiedliche Fragen in den drei Fragebögen betrafen Angaben über die

Person (nur Semesteranfang), oder z.B. inhaltliche Fragen, die sich im Laufe der Lehrveranstaltung erst ergaben.
Zudem haben sich die Fragebögen der Präsenzveranstaltung und des Online-Kurses inhaltlich unterschieden: der Online-Kurs „OOPJ“ ist ein anderes Studienfach, die Studierenden haben einen anderen Hintergrund und andere Voraussetzungen, allein dadurch, dass sie in einem Studiengang studieren, in dem der Online-Kurs selbst schon über digitale Medien gelehrt wird. Es wurden aber auch identische Fragen eingesetzt, zum Beispiel in Hinsicht auf bestimmte digitale Medien wie Animationen oder interaktive Elemente - um die beiden Lehrveranstaltungsarten, in denen Materialien zum Thema Entwurfsmuster eingesetzt wurden, direkt vergleichen zu können.

4.2 Durchführung der Erhebung

Die verschiedenen Fragebogen wurden von den Studierenden am Rechner ausgefüllt und online abgeschickt. Alle Daten konnten problemlos empfangen werden. Im Sommersemester 2003 waren in der Präsenzveranstaltung „Programmiertechniken“ die Rücklaufquote zu allen Befragungszeitpunkten 100%, was wahrscheinlich auf den guten Kontakt zu den teilnehmenden Studierenden zurückzuführen ist, die die Fragebögen in ihrem Praktikum ausfüllen sollten. Am Ende des Semesters fand eine Gruppendiskussion für diese Lehrveranstaltung zum Thema „Digitale Medien in der Lehre“ statt, an der wiederum alle Studierenden teilnahmen. Insgesamt liegen damit zwar nur wenige Daten vor - bedingt durch die geringe Studierendenzahl im Sommersemester 2003, die Daten sind dafür aber vollständig.
Beim Online-Kurs „Objektorientierte Programmierung mit Java“ zeigte sich leider, dass die für den Kurs eingetragenen Studierenden nicht genügend motiviert werden konnten, an der Evaluation teilzunehmen. Aufgrund der sehr schlechten Rücklaufquote wurde auf das Einstellen eines dritten Fragebogens zum Semesterende verzichtet, da die Erreichbarkeit der Studierenden nicht gegeben war. Eine Auswertung wurde deshalb nicht vorgenommen.
Bedingt durch den Standort Lübeck, war die Anzahl der an der Erhebung teilnehmenden Studierenden recht klein, für die Präsenzlehrveranstaltung „Programmiertechniken“ zeichneten sich im Sommersemester 2003 jedoch folgende Erkenntnisse ab:

- Die Studierenden besitzen unterschiedliche Vorkenntnisse und unterschiedliche Lernereigenschaften.
- Gruppenarbeit und Interaktion allgemein ist wichtig für besseres Lernen.
- Lernen über Beispiele und Abbildungen ist wichtig und geht am besten.
- Animationen machen Spaß und sind nützlich für das Verständnis.
- Studierende nutzen Lehrmaterialien in Druckversion und in digitaler Form (Internet, Online-Referenzen, Online-Skript).
- Digitale Medien werden gerne angenommen; gute Mischung der Medien, Vielfalt und auch gute Qualität der angebotenen Medien ist den Studierenden wichtig.
- Einstellungen bezüglich erhobenem Lernverhalten und bezüglich digitalen Medien verändern sich über das Semester nur wenig.

Die Ergebnisse aus der Evaluation im Sommersemester 2003 sind durch die kleine Datenbasis nur als Trend einzustufen. Dennoch waren die Erkenntnisse aus mehrfacher Sicht hilfreich:

- Die Ergebnisse aus der Präsenzlehrveranstaltung zeigten sich insgesamt homogen, so dass ausgehend von dieser Basis der Einsatz digitaler Medien weitergestaltet werden konnte.
- Die Online-Fragebögen als Erhebungsinstrument haben sich bewährt. Das Ausfüllen, Abschicken und Empfangen hat sehr gut funktioniert, und die Motivation war bei den Ausfüllenden der Präsenzveranstaltung sehr hoch (zu entnehmen aus den gleichmäßig hohen Ausfüllzeiten und den oft umfangreichen und aufschlussreichen Antworten der Studierenden).

Im Wintersemester 2003/2004 fand erneut eine Evaluation statt, wiederum in beiden Lehrveranstaltungen. In der Erhebung wurden die Items der Fragebögen noch direkter auf die im Projekt entwickelten Materialien bezogen, so dass hier noch deutlichere Aussagen bezüglich der eingesetzten digitalen Materialien erwartet werden. Die erhobenen Daten in diesem Semester sind diesmal in beiden Lehrveranstaltungen nahezu vollständig, wenn auch die Datenbasis klein bleibt, bedingt durch die anzahlmäßig kleinen Studiengänge am Standort Lübeck. Die Auswertung der Evaluation des jetzigen Wintersemesters ist derzeit allerdings noch nicht beendet. In einem Gesamtabschlussbericht werden alle erhobenen und evaluierten Daten zusammengefasst und dann auch in Beziehung zu den Ergebnissen aus dem Sommersemester gesetzt. Die Ergebnisse gehen in die Planung weiterer entsprechender Lehrangebote zum Thema Entwurfsmuster ein.

5 Zusammenfassung

Das Ziel der Lerneinheit Entwurfsmuster besteht in der Vermittlung des allgemeinen Konzeptes von Entwurfsmustern und in der Vermittlung von ausgewählten Entwurfsmustern. Ein wichtiges Lernziel besteht darin, dass die Studierenden selbständig weitere Entwurfsmuster bei Bedarf auswählen, sich aneignen und umsetzen können. Die Lerneinheit besteht aus den Teilen Einführung in Entwurfsmuster, Entwurfsmuster, Durchgängiges Beispiel, Ausblick und dem Referenzteil. Die erstellten Lernmodule werden im Wesentlichen in zwei unterschiedlichen Szenarien eingesetzt. Im ersten Szenario werden die erstellten Materialien, insbesondere die Animationen, in der Präsenzvorlesung zur Erläuterung der Lehrinhalte eingesetzt. Im zweiten Lernszenario dienen die erstellten Materialien dem Selbststudium.
Zur Entwicklung der Lerneinheit wird eine Vielzahl von Werkzeugen eingesetzt. Die Erstellung von multimedialen Lehr-/Lernmaterialien ist im Vergleich zur Erstellung von herkömmlichen Materialien sehr aufwändig. Aus den Mitteln, die normalerweise für die Lehre zur Verfügung stehen, können multimediale Materialien nicht entwickelt werden.
Die Arbeitsumgebung der Studierenden besteht aus Standard- Werkzeugen, die für die Ausbildung kostenlos zur Verfügung stehen und einfach installierbar sein sollten.
Evaluationen der entwickelten Materialien wurden durchgeführt. Die an der Evaluation im Sommersemester 2003 teilnehmenden Studierenden bewerteten Gruppenarbeit und Interaktion als wichtig, Lernen über Beispiele und Abbildungen funktioniert bei (fast) allen am besten.

Digitale Medien werden gerne angenommen, wie z.B. Animationen, die das Verständnis erleichtern und motivierend sind. Wenn verschiedene Formate der Lehrmaterialen angeboten werden, so werden diese von den Studierenden auch angenommen, wobei auf Qualität der Lehrmaterialien großer Wert gelegt wird. Die erhobenen Einstellungen der Studierenden zum Einsatz von digitalen Medien verändern sich –nach mehr oder weniger erstmaliger Unterstützung der Lehre durch diese Angebote– über das gesamte Semester nur wenig.

Literatur

[BS03] Werner Beuschel and Silke Seehusen. Lehr- und Lernformen für web-basierte Studiengänge - Erfahrungen aus E-Learning-Projekten. In A. Schwill, editor, *1. Workshop Grundfragen multimedialer Lehre*, 2003.

[Coo00] James W. Cooper. *Java Design Patterns, A Tutoria*. Addison-Wesley, 2000.

[DSB00] Hans Rohnert Douglas Schmidt, Michael Stal and Frank Buschmann. *Pattern-Oriented Software Architecture - Patterns for Concurrent and Networked Objects*, volume 2. John Wiley, 2000.

[EGV95] Ralph Johnson Erich Gamma, Richard Helm and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

[FBS96] Hans Rohnert Peter Sommerlad Frank Buschmann, Regine Meunier and Michael Stal. *Pattern-Oriented Software Architecture - A System of Patterns*. John Wiley and Sons, Chichester, 1996.

[Geb] Markus Gebhard. Java2html. http://www.java2html.de.

[Gen04] Poseidon. http://www.gentleware.org, 2004.

[Gra98] Mark Grand. *Patterns in Java*, volume 1. Wiley, 1998.

[LS02] C. Lecon and S. Seehusen. Combining Structure Search and Content Search for Online Courses. In A.M. Tjoa and R.R. Wagner, editors, *13th International Workshop on Database and Expert Systems Applications: MIW'2002 - The 3rd International Workshop on Management of Information on the Web - Web-Based Teaching and Learning*, pages 366–370, Aix-en-Provence, France, September 2-6 2002. IEEE Computer Society.

[SL01] S. Seehusen and C. Lecon. Entwicklung von Online-Kursen für den längerfristigen Einsatz. In K. Bauknecht, W. Brauer, and T. Müeck, editors, *Informatik 2001 (Workshop Virtuelle Lernräume)*, Vienna, Austria, September 26-28 2001. Oesterreichische Computer Gesellschaft.

[SLK00] S. Seehusen, C. Lecon, and C. Kaben. Specification of Learning Paths in Virtual Courses. In *Frontiers in Education Conference (FIE'2000)*, pages S3D–11, Kansas City, USA, October 18-21 2000. http://www.informatik.fh-luebeck.de/ti/Seehusen/Publications/FIE00/FIE2000Seehusen.pdf.

[W3C] 2003. http://www.w3.org/XML.

Der MuSofT-Abschlussbericht

Peter Aschenbrenner, Andy Schürr

peter@informatik.unibw-muenchen.de, andy.schuerr@es.tu-darmstadt.de
Real-Time Systems Lab (FG EchtzeitSysteme)
Darmstadt University of Technology

Inhaltsverzeichnis

1 Einleitung

Im Rahmen des MuSofT-Projektes hat das Teilprojekt 2.3 eine multimedial aufbereitete Lerneinheit zum Thema “Algorithmen und Datenstrukturen” entwickelt, die in verschiedenen Umfängen für ganz unterschiedliche Studiengänge eingesetzt werden kann. Will man dieses klassische Vorlesungsthema der Informatik multimedial unterstützen, so bietet sich insbesondere die Animation und Visualisierung der behandelten Standardalgorithmen und -datenstrukturen an. Wie bereits in [EE02] und [KDE03] beschrieben findet man zwar heute - besonders im Internet - eine große Anzahl von fertigen Animationen und Werkzeugen, die solche Visualisierungen für einfache Datenstrukturen und Algorithmen anbieten oder deren Erstellung vereinfachen (z.B. [CCA01], [Ani01], [Sor01], [Bub01]).
Allerdings bestehen in nahezu allen Fällen Einschränkungen in der Ausnutzung des möglichen Potentials der multimedialen Lehre von Algorithmen und Datenstrukturen (wie fest vorgegebener Ablauf ohne flüssige Animationen oder völlig ohne Interaktion; fehlende Konfigurierbarkeit für neue Datenstrukturen).
Im Rahmen der Lerneinheit 2.3 wird deshalb ein neues Konzept für die Gestaltung von Lernmodulen im Bereich “Algorithmen und Datenstrukturen” entwickelt, das folgende Punkte unterstützt:

- Standard-Datenstrukturen und Algorithmen werden nach softwaretechnischen Gesichtspunkten mit graphischen Notationen (eine Teilmenge der UML) modelliert.
- Aus den erarbeiteten Modellen werden lauffähige Animationen generiert, die entweder als reine Präsentationen in Vorlesungen integriert ablaufen oder interaktives Arbeiten im Rahmen von Übungen oder dem Selbststudium erlauben.
- Das graphische Debugging selbst erstellter Programme von Studenten wird mithilfe von Visualisierungsbausteinen und automatischer Überprüfung von Invarianten von Datenstrukturen unterstützt.

Die Tätigkeiten im ersten Jahr umfassten u.a. den Test verschiedenster Animationskonzepte, fertiger Animationen und Animationsplattformen. Im Jahr darauf wurde auf der Basis der ausgewählten Plattformen das Rahmenwerk VIDEA (Visual Interactive Data structure Environment for Animations; [PS03] liefert eine Beschreibung) aufgebaut und das AVL-Beispiel in diesem Rahmenwerk erstellt. Außerdem wurde in diesem zweiten Projektjahr die Vorlesung als modularer, auf verschiedene Arten zusammensetzbarer Foliensatz (nach internem Sprachgebrauch also in ’Level1-Version’) entwickelt, eingesetzt und evaluiert, vorerst in der Programmiersprache Ada. Im dritten Jahr wurde das Rahmenwerk vervollständigt und es kamen weitere Animationen hinzu. Außerdem fand im dritten Jahr ein Praktikum an der TU Darmstadt statt, in dem konfigurierbare Flash-Animationen eingesetzt wurden, die in den Jahren 2002 und 2003 entwickelt worden waren.

Drei Varianten der Lerneinheit 2.3 wurden ins MuSofT-Portal ([Por03]) eingestellt, die aus konfigurierten

- Foliensätzen

- Flash-Animationen und
- Instanzen der interaktiven Visualisierungsumgebung VIDEA bestehen.

2 Vorstellung der Lerneinheit

2.1 Präsenzveranstaltung

Als erstes wurde die Lerneinheit 2.3 als Vorlesung realisiert und zwar konfigurierbar für Studenten der Elektrotechnik oder der Informatik (zu finden unter [Ein03]). Den Studierenden werden im wesentlichen folgende Lehrinhalte vermittelt:

- eine Auswahl der wichtigsten Standardalgorithmen und -datenstrukturen (insbesondere für Suchen und Sortieren auf Listen, Bäumen und Graphen)
- Prinzipien des Entwurfs von Algorithmen (wie “divide and conquer”, “Greedy”-Algorithmen, dynamische Programmierung, . . .)
- Programmiersprachliche Konstrukte imperativer und objektorientierter Art (z.B. Umgang mit Zeigerstrukturen, Ausnahmebehandlung zur systematischen Fehlerbehandlung etc.)
- Objekt- und Klassendiagramme der UML als graphische Hilfsmittel zum Entwurf von dynamischen Datenstrukturen
- die Idee der Datenabstraktion (Pakete, Schnittstellen etc.) und der Entwicklung generischer Programmeinheiten
- Invarianten und einfache Vor- und Nachbedingungen zur Beschreibung von Schnittstellen
- systematischer Test entwickelter Algorithmen und Datenstrukturen
- Floyd-Hoare-Kalkül zur Programmverifikation
- O-Kalkül zur Charakterisierung von Laufzeit- und Speicherplatzverhalten

Die Verwendung der zu unterrichtenden Datenstrukturen wird mithilfe einer fiktiven Speditionsfirma “Blitz AG” motiviert und demonstriert (siehe [Ein03] und [Por03]).

2.2 VIDEA

Ergänzend dazu werden in der entwickelten Visualisierungsumgebung VIDEA graphische Lernszenarien und interaktive Übungsbeispiele für AVL-Bäume, Graphenalgorithmen (die kürzeste Wegesuche und das Spannbaum-Problem) und die doppelt verkettete Liste zur Verfügung gestellt. Für den AVL-Baum wurde dieses Ziel - inklusive aller notwendigen Implementierungen für die Laufzeitumgebung - bereits im Jahr 2002 erreicht und der MuSofT-Projektleitung präsentiert.

So sieht sich der Lernende etwa einem bestehenden binären Baum gegenüber, der z.B. die Suchbaum-Eigenschaft oder die Balancierungs-Eigenschaft verletzt und muss diese mit Löschen und Wiedereinfügen geeigneter Knoten oder durch Rotationsoperationen wiederherstellen. Dabei unterstützen ihn Fehlermeldungen der Laufzeitumgebung und farbliche Markierung der Wurzelknoten fehlerhafter Teilbäume. Abbildung 1 zeigt eine Beispielsitzung mit einem AVL-Baum, der die Balancierungs-Eigenschaft verletzt; der Wurzelknoten des unbalancierten Teilbaums erscheint deswegen farblich markiert (im Bild etwas dunkler).

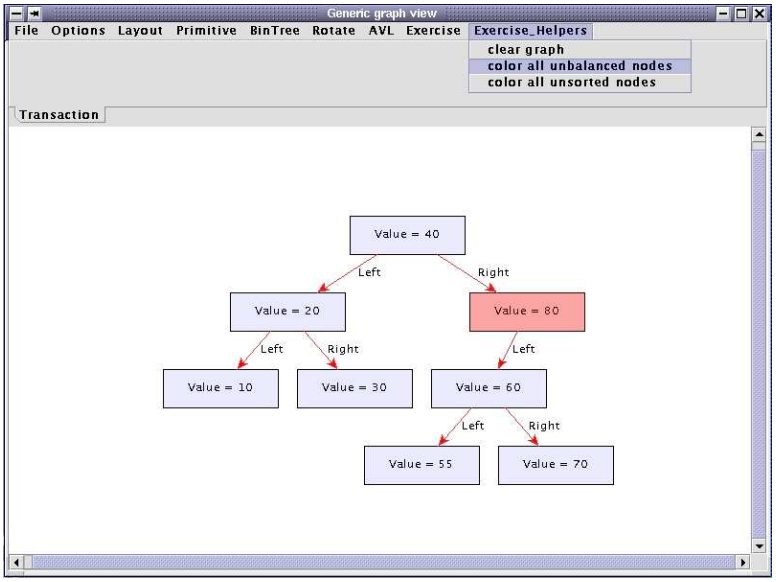


Abbildung 1: AVL-Baum, der die Balancierungs-Eigenschaft verletzt

Oder der Lernende baut sich mit Hilfe mehr oder weniger mächtiger Einfügeoperationen (die weniger mächtigen erlauben es, Invarianten der Datenstruktur zu verletzen, um auch Fehlersituationen entstehen zu lassen) selbst einen AVL-Baum zusammen und lernt dabei anschaulich dessen Verhalten kennen.
Weitere interaktive Aufgaben für die restlichen erwähnten Datenstrukturen kamen im Jahr 2003 (zusammen mit der dafür benötigten Funktionalität der Visualisierungsumgebung VIDEA) hinzu.
Eine besonders hervorzuhebende Eigenschaft von VIDEA ist die Anpassbarkeit an neue Datenstrukturen (durch die Verwendung von Graphgrammatiken, vgl. Abschnitt 3.2.1). Dadurch soll insbesondere der spätere Einsatz auch nach dem Ende von MuSofT gewährleistet werden. Als weitere Besonderheit ist es möglich, die Laufzeitumgebung als graphischen Debugger für selbstgeschriebene Java-Programme der Lernenden einzusetzen. Dabei verwenden diese in ihrem Java-Code, der z.B. eine Linksrotation in einem AVL-Baum (siehe Abbildung 2) durchführen soll, Primitive wie setLeft und setRight, die bereits Corba-Aufrufe enthalten und dadurch zu entsprechender Veränderung des visualisierten Graphen und dazu passenden Meldungen bzw. Fehlermeldungen führen.

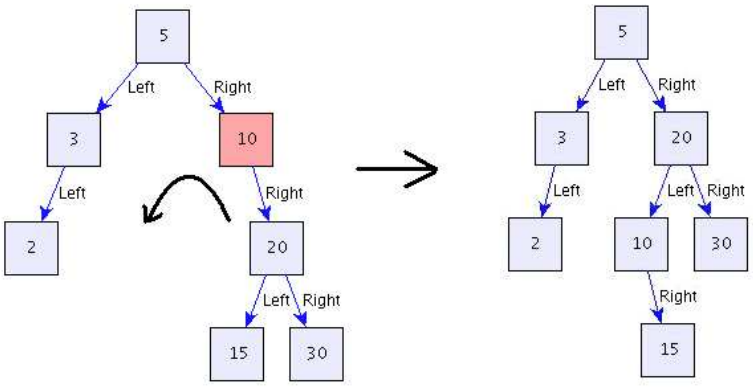


Abbildung 2: Klassische Darstellung der Linksrotation ein einem AVL-Baum

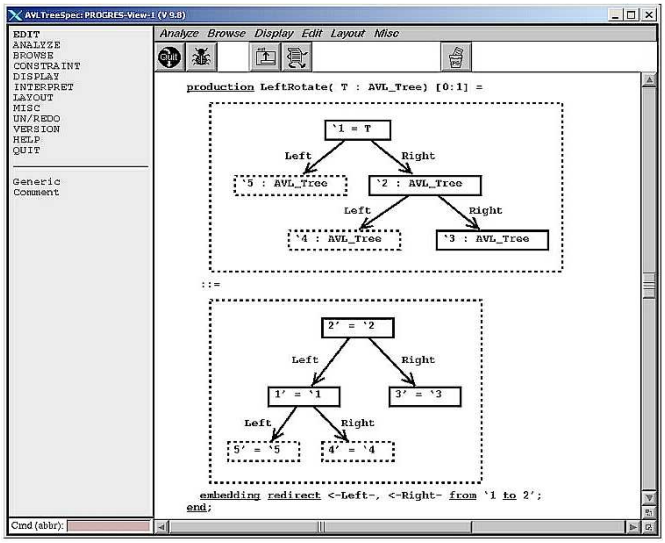


Abbildung 3: Spezifikation der Linksrotation in PROGRES

Wir betrachten ein Beispiel mit zwei fiktiven Gruppen von Studenten, die jeweils ein Java-Programm zur Lösung der Linksrotation auf einem AVL-Baum geschrieben haben.

Abbildung 4 zeigt ein Szenario, in dem man auf der linken Seite den Javacode-Debugger (hier beispielhaft Borland Together) sieht und gleichzeitig auf der rechten Seite einen kleinen (ehemaligen) Baum, auf den bereits ein Teil der programmierten Links-Rotation ausgeführt wurde.

Die erste Gruppe hat dabei einen kleinen Fehler gemacht, der dazu führt, dass ein Zeiger falsch umgehängt wird. Solche Fehler sind nicht immer leicht zu finden. Da die fiktive Gruppe

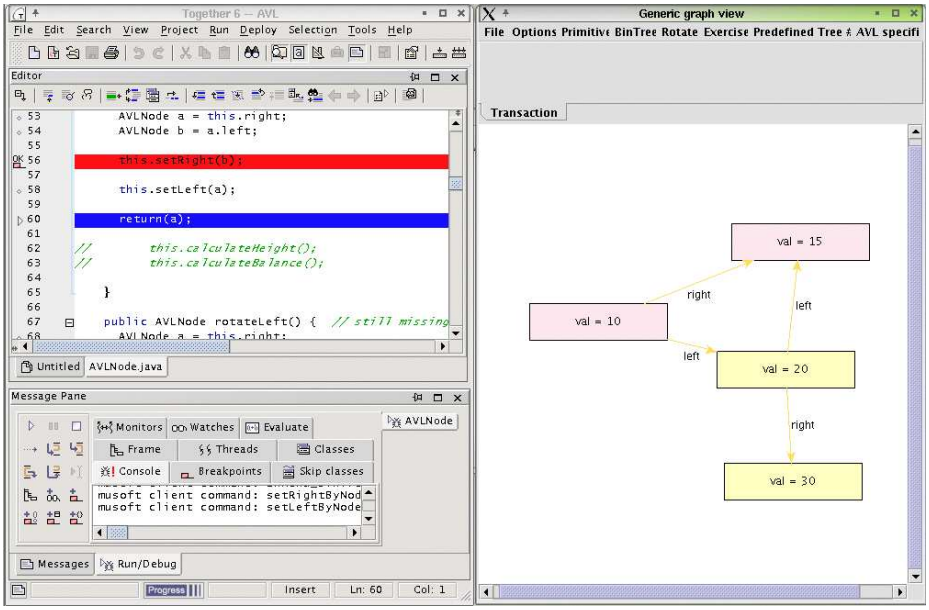


Abbildung 4: Visuelles Debugging mit Together und VIDEA: Zwischenzustand der Linksrotation. Die Knoten ’10’ und ’15’ sind wegen temporär verletzter Baum- und AVL-Eigenschaften rot; das ist im Schwarzweiß-Druck leider kaum zu sehen.

aber ihr Programm wie in Abbildung 4 gezeigt mit VIDEA graphisch debugged, können die Studierenden leicht sehen, an welcher Stelle was genau schief geht.
Die zweite Gruppe hat korrekten Code geschrieben, so dass die Rotation - wie in Abbildung 5 gezeigt - reibungslos funktioniert.
Zusammenfassend kann VIDEA also eingesetzt werden:

- Zur Unterstützung der Präsenzlehre (etwa um den Aufbau eines AVL-Baums in der Vorlesung zu illustrieren).
- Zur Unterstützung des Übungsbetriebes (etwa um Beispiele am Beamer “durchzuspielen” anstatt an einer Tafel die verschiedenen Schritte etwa bei der Linksrotation eines AVL-Baums nacheinander hinzuschreiben und wegzuwischen).
- Als Umgebung für die Durchführung des interaktiven Übungsbetriebs selbst (unterstützend durch Fehlermeldungen, Tips, . . .) für ausgewählte Aufgaben.
- Zum (interaktiven) Selbststudium der Studierenden.
- Zum visuellen Debuggen eigener Programme.

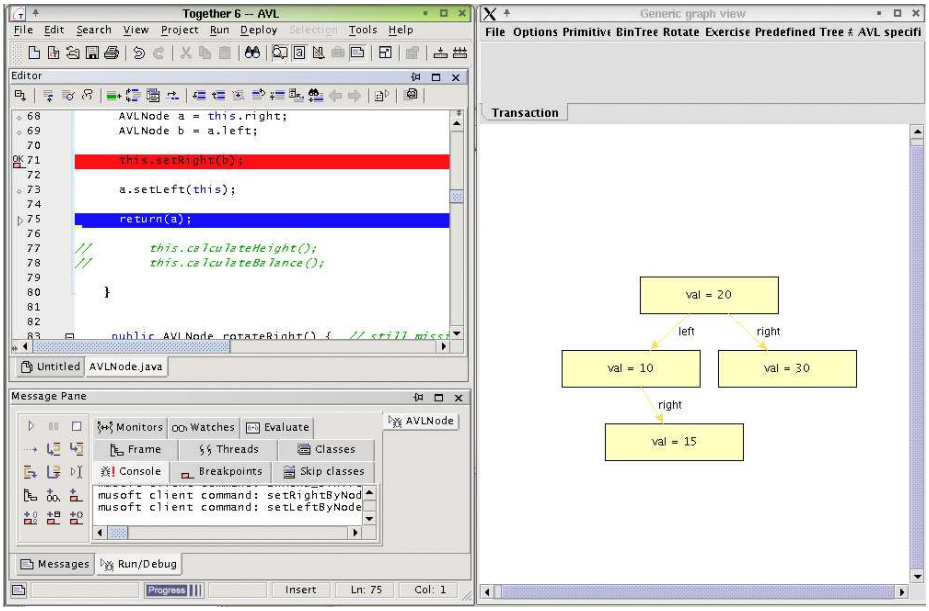


Abbildung 5: Visuelles Debugging mit Together und VIDEA: Ende der Linksrotation

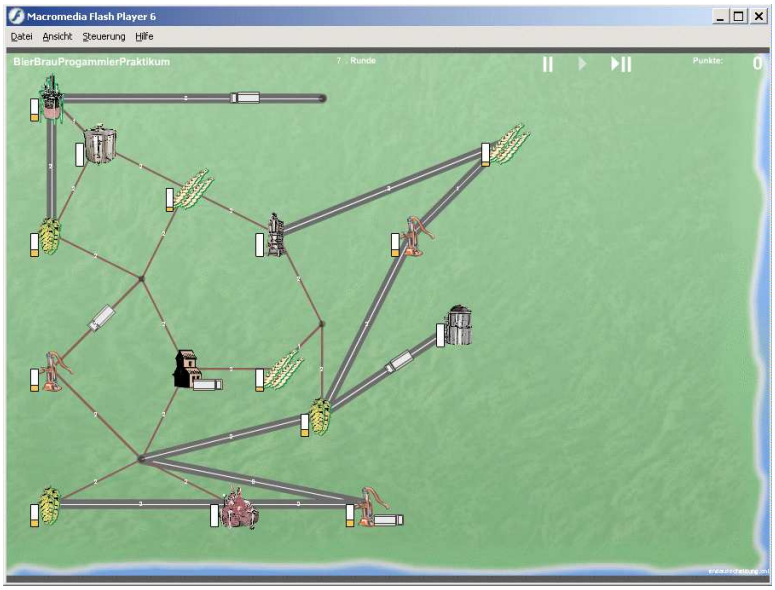


Abbildung 6: Einer der Siegeralgorithem des Jahres 2003 beim “Bier erzeugen”

2.3 Praktikum

Das Software Praktikum ist als Ergänzung zu den Vorlesungen gedacht. Die erste Zielgruppe waren Studenten der Elektrotechnik der TU Darmstadt, die vor dem Jahr 2003 an den Veranstaltungen “Informatik I” und “Informatik II” teilgenommen hatten und dort das Programmieren in der Sprache Java erlernt haben. Das Praktikum besteht aus einem Spiel, in dem es darum geht, in einer vorgegebenen Zeit eine möglichst große Menge Bier zu produzieren. Zu diesem Zweck ist ein Java-Rahmenwerk mit eingebetteter Flash-Animation zur Visualisierung gegeben. In diesem Rahmenwerk sind alle Regeln sowie die Produktionskette über 8 Produkte hinweg zu Bier enthalten. Aufgabe der Studenten ist es, am Ende über einen eigenen individuellen Gesamtalgorithmus zu verfügen, der auf beliebige Szenarien (Spielpläne) angewendet wird. Im Rahmen eines Wettbewerbs über zwei Ausscheidungsrunden werden schließlich die drei besten Algorithmen ermittelt und prämiert. Abbildung 6 zeigt den Ablauf eines der Siegeralgorithmen im Jahr 2003.

3 Technische Realisierung

3.1 Präsenzveranstaltung

Wie bereits in der Einleitung erwähnt wurde das Lehrmaterial des Teilprojektes 2.3 bereits in drei Ausprägungen (drei Lerneinheiten) ins MuSofT-Portal gestellt.
Um das speziell für die Vorlesung zu ermöglichen, wurden mit Adobe Framemaker Foliensätze erzeugt, in denen bestimmte Elemente - wie Quellcode - austauschbar sind. Aus diesen können Folien im Acrobat-Format pdf generiert werden, wie das auch schon mit den beispielhaften und modular in verschiedenen Kombinationen nutzbaren Foliensätzen geschah, die als Beispiel-Präsenzveranstaltung ins MuSofT-Portal eingestellt wurden.

3.2 VIDEA

Wie schon im Abschnitt 2 beschrieben ist die Konfigurierbarkeit - besonders im Hinblick auf die Austauschbarkeit der jeweiligen zu unterrichtenden Datenstruktur - in VIDEA ein ganz zentrales Anliegen. Das bedeutet aber, dass eine technische Repräsentation der Datenstruktur, ihrer Schnittstellenoperationen und ggf. des Algorithmus existieren muss, die möglichst entkoppelt ist vom Rest des Systems und die nach Möglichkeit nahe an den sowieso genutzten Repräsentationen für Datenstrukturen liegt.
In einer Evaluation verschiedener Paradigmen und Tools für die Erzeugung von passenden Visualisierungen in der ersten Projektphase zeigte sich der Einsatz einer Graphtransformationsumgebung diesen Anforderungen besonders gewachsen. Die Gründe werden im nächsten Abschnitt beschrieben.

3.2.1 Graphtransformationen

In der herkömmlichen Lehre von Datenstrukturen in Informatiklehrbüchern werden Schnittstellenoperationen wie z.B. die Linksrotation in einem AVL-Baum (siehe Abbildung 2) oft als Aufeinanderfolge zweier Zustände dieser Datenstruktur dargestellt. Wenn man nun sowohl

den Zustand der Datenstruktur vor der Schnittstellenoperation als auch den Zustand danach jeweils als Graph auffasst, kann man die Schnittstellenoperation auf ganz natürliche Weise als Graphtransformation interpretieren. Deswegen bieten sich Graphen und Graphtransformationen als Beschreibungsmittel für Datenstrukturen, Schnittstellenoperationen und Algorithmen und damit als Spezifikationsmittel für unsere Visualisierungsumgebung VIDEA an (bezüglich des Stichwortes der Spezifikation sei noch gesagt, dass man im Sinne der UML jede Graphtransformation als Paar von Objektdiagrammen oder als eine Menge von Paaren von Objektdiagrammen auffassen kann).

Um eine automatisierte Generierung von VIDEA-Instanzen aus diesen Spezifikationen zu ermöglichen, muss es eine Laufzeitumgebung geben, die aus einer solchen Spezifikation mithilfe von Namenskonventionen, weitergehender Konfigurierung (z.B. die Einstellung eines zu verwendenden Layoutalgorithmus) und vorgegebener Übungsbeispiele eine Art interaktiven Graphbrowser realisiert, in dem die Lernenden dann den jeweiligen Zustand der zu unterrichtenden Datenstruktur sehen und manuell oder programmatisch verändern und manipulieren können.

Das Java-Rahmenwerk UPGRADE2 ([UPG02]) unterstützt die Erstellung einer solchen Laufzeitumgebung basierend auf dem Graphtransformationssystem PROGRES ([PRO01]). Beide werden im folgenden Abschnitt kurz vorgestellt.

3.2.2 Rahmenwerk

Im gewählten Graphtransformationswerkzeug PROGRES können die Datenstrukturen, ihre Schnittstellenoperationen und ggf. die Algorithmen auf graphischem (oder auch textuellem) Wege wie im Abschnitt 3.2.1 gefordert spezifiziert werden (für eine Einführung in die PROGRES-Sprache siehe etwa [And96]). Eine Beispielspezifikation der Linksrotation war in Abbildung 3 zu sehen. Man beachte den Wiedererkennungswert zu einer “herkömmlichen” visuellen Beschreibung wie in Abbbildung 2.

Wie bereits beschrieben ist einer der Hauptgründe für die Verwendung eines Graphtransformationswerkzeugs die Erweiterbarkeit und Wiederverwendbarkeit unserer Lerneinheit dank der naheliegenden visuellen Notation bei der Definition neuer Datenstrukturen und Algorithmen. Selbstverständlich muss auch hierbei mit einem gewissen Einarbeitungsaufwand eines hierin evtl. noch unerfahrenen Dozenten in die gewählte Graphtransformationsumgebung, hier also PROGRES, gerechnet werden, der aber unserer Meinung nach wesentlich geringer ist als z.B. bei einer Erweiterung eines vergleichbaren reinen Java-Visualisierungssystems für neue Datenstrukturen; zudem erleichtert unser Ansatz eine Erweiterung oder Anpassung durch die saubere Trennung zwischen Spezifikation der Datenstruktur und Implementierung der Visualisierung. Für den Einsatz bereits bestehender Instanzen unserer Visualisierungsumgebung sind dagegen keinerlei Kenntnisse von Graphtransformationen oder eines entsprechenden Werkzeugs notwendig.

Als Java-Rahmenwerk für den Bau der Laufzeitumgebung wurde UPGRADE2 gewählt, weil hier schon ein automatischer Prozess der Generierung von Java-Datenstrukturen aus Graph-Spezifikationen implementiert ist. Außerdem stehen einige hilfreiche Basisoperationen wie die einfach konfigurierbare Ausführung definierter Graphtransformationen aus dem Menü und die automatische Anzeige von Parameterfenstern für Graphtransformationen zur Verfügung.

Für die letztendliche Darstellung der Graphen wird innerhalb von VIDEA noch eine Java-basierte Graphvisualisierungsbibliothek benötigt. Hier gab es in UPGRADE2 eine bereits ziemlich aufwendig ausprogrammierte Lösung, die auf ILog's JViews ([JVi01]) basiert. Aus verschiedenen Gründen (u.a. Lizenzgründe und die Notwendigkeit, kontinuierliche Animationen im Zusammenhang mit Layoutalgorithmen darstellen zu können) haben wir uns im Teilprojekt 2.3 entschieden, anstatt der JViews-Lösung eine eigene mithilfe der kommerziellen, ursprünglich an der Uni Tübingen entwickelten Graphenbibliothek yFiles ([yfi01]) zu entwickeln.

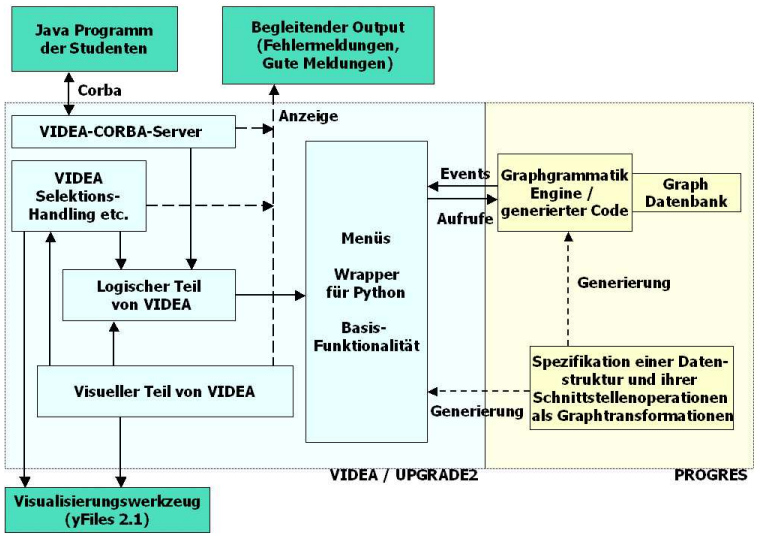


Abbildung 7: Aufbau von VIDEA

3.2.3 Aufbau

Das Zusammenspiel der bisher schon beschriebenen Komponenten wird in Abbildung 7 überblicksartig dargestellt.
Innerhalb des großen Kastens sind diejenigen Komponenten rechts dargestellt, die mit PROGRES realisiert sind, während die zu UPGRADE2/VIDEA gehörigen Teile sich links befinden.
Im folgenden soll anhand Abbildung 7 kurz das Prinzip des Konfigurationsprozesses einer neuen VIDEA-Instanz beschrieben werden: Als erstes spezifiziert der Dozent, der die Lerneinheit einsetzen will

- eine Datenstruktur wie etwa den AVL-Baum als PROGRES-Knotentyp,

- wichtige Schnittstellenoperationen (Einfügen eines neuen Elementes, Linksrotation, ...) als Graphtransformationen

- und für Übungsaufgaben benötigte Hilfsprozeduren auf dem AVL-Baum (wie z.B. den Test 'IsBalanced', der später verwendet werden kann, um die Lernenden die Balancierungs-Eigenschaft eines Teilbaums manuell überprüfen zu lassen) als PROGRES-Produktionen oder -Tests.

Aus dieser Spezifikation wird als nächstes in einem automatisierten Verfahren C-Code generiert, der sowohl die Basis für eine weitere Generierung von UPGRADE2-Datenstrukturen für Menüs etc. darstellt als auch die notwendige Basis für die PROGRES-Laufzeitumgebung, die später die Graphtransformationen ausführt, in der Graphdatenbank speichert, Aufrufe von VIDEA entgegennimmt und Events für VIDEA generiert.
Die restlichen Komponenten seien anhand eines kleinen Beispielablaufs charakterisiert: Wenn der Lernende etwa im Rahmen einer Übung zum manuellen Aufbau eines AVL-Baumes einen gerade neu erzeugten Knoten mithilfe einer Produktion 'AddLeft' als linken Sohn eines schon im Baum befindlichen Knotens einfügen will, kann er zuerst den zukünftigen Vater-Knoten mit der Maus markieren, wobei der Selektionsmanager über eine yFiles-Funktionalität den markierten Knoten identifiziert, mithilfe des logischen Teils von VIDEA in eine PROGRES- / UPGRADE2-Knotennummer umrechnet und für die weitere Verwendung in einer entsprechenden UPGRADE2-Datenstruktur vormerkt. Das gleiche passiert nun mit dem neuen linken Sohn. Unter der Bedingung, dass die vorher erwähnte PROGRES-Produktion 'AddLeft' zwei Knoten als Parameter erwartet und zwar zuerst den Vater und dann den neuen linken Sohn, kann der Lernende nun im Menu 'AddLeft' aufrufen. In diesem Fall verpackt die UPGRADE2-Basisfunktionalität von VIDEA diesen Aufruf in einen Aufruf der entsprechenden PROGRES-Produktion und übergibt gleich die zugehörigen Parameter. Die Graphgrammatik-Engine führt die Graphtransformation auf ihrer Datenbank aus und liefert dementsprechende Events zurück, etwa im Erfolgsfall ein 'AddEdge'-Event für die neu hinzukommende Kante. Der Layoutmanager (enthalten im visuellen Teil von VIDEA) erfährt davon mithilfe der Basisfunktionalität und führt mit Ausnutzung von yFiles-Funktionalität einen flüssigen Animationsschritt aus, in dem der neue linke Sohn an seinen ihm nun zustehenden Platz im Graphen (links unter seinem Vater) bewegt wird.
Im Fehlerfall wäre im Rahmen des begleitenden Outputs eine Fehlermeldung ausgegeben worden und der Graph hätte sich nicht verändert.
Die bisher nicht beschriebene Komponente 'Java-Programm der Studenten' bezieht sich auf die Möglichkeit, mithilfe von VIDEA ein graphisches Debugging eigener Programme der Lernenden durchzuführen.
Diese Funktionalität kam im Jahr 2003 hinzu und ist bereits im Abschnitt 2.2 beschrieben worden. Technisch basiert sie auf einem Corba-Server, der lediglich auf dem JDK1.4.1 aufbaut (und also möglichst plattformunabhängig ist) und einem Java-Wrapper, der den Corba-Client implementiert und dem Studenten-Programm Methoden liefert zur Fernsteuerung von VIDEA.
Ein Beispiel war bereits in Abbildung 4 zu sehen, wo die Studierenden sich auf existierende Methoden setLeft, setRight u.ä. abstützen dürfen (die bereits Corba-Aufrufe für VIDEA enthalten) und damit Rotationsoperationen für den AVL-Baum implementieren.

3.3 Praktikum

Die Realisierung des Rahmenwerks für das Praktikum geschah im Wesentlichen in Java. Die Studenten fertigen ihre Lösungen direkt im Java-Bestandteil des Rahmenwerks an festdefinierten Stellen ein. Dabei benutzen sie die integrierte Entwicklungsumgebung Together von Borland (siehe Abbildung 8 für die Paketstruktur).

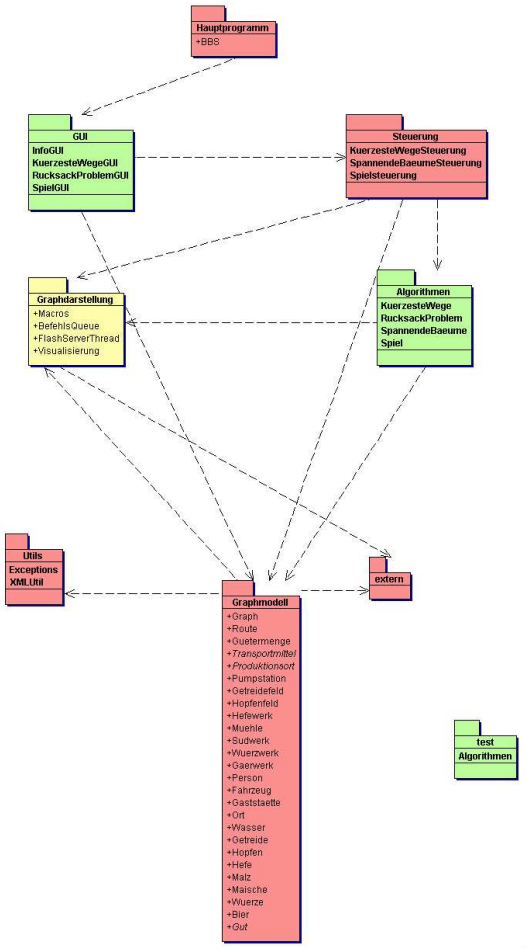


Abbildung 8: Paketstruktur im Praktikum

Als Format für die einzelnen Szenarien wurde das xml-Format gewählt. Dieses wird über den xml-Parser Xerces eingelesen und direkt vom Rahmenwerk geschrieben. Zur Erstellung von Szenarien enthält das Rahmenwerk einen Editor, der sich der graphischen Visualisierung, die auch für die Darstellung der Endalgorithmen verwendet wird, bedient.

Diese Visualisierung stellt, wie auch die übrigen Visualisierungen für die Teilprobleme, eine Flash-Anwendung dar. Sie wird vom Java-Rahmenwerk über eine Socket-Verbindung durch Befehle im xml-Format gesteuert. Es gibt keine Kommunikation von der Visualisierung zum Java-Rahmenwerk. Somit kann die Visualisierung auch nach dem Ende der Java-Applikation weiterlaufen und bereits übermittelte Befehle aus ihrer Warteschlange abarbeiten und visualisieren.
Um den Test-First Ansatz des eXtreme Programming umzusetzen, benutzen die Studenten die JUnit-Test-Unterstützung von Together.

4 Erfahrungen

4.1 Präsenzveranstaltung

Die erste Fassung der Lerneinheit 2.3 kam bereits im Wintertrimester 2002 an der UniBw München zum Einsatz. Anstelle von Java (der Sprache, auf die man sich in MuSofT geeinigt hat) wurde in der Vorlesung “Einführung in die Informatik II” noch die Programmiersprache ADA verwendet. Das war möglich, weil (wie schon beschrieben) die Lerneinheit so konzipiert ist, dass alle programmiersprachlichen Beispiele in einfacher Weise austauschbar sind. Als kleinster gemeinsamer Nenner der ADA und Java zu Grunde liegenden Programmierparadigmen wird deshalb eine objektbasierte Softwareentwicklung propagiert, die sich in beiden Programmiersprachen in sinnvoller Weise umsetzen lässt.
In der beschriebenen Vorlesung wurden bereits Animationen eingesetzt, die allerdings noch meistens passiv oder auf relativ einfache Interaktionsbeispiele mit Sortieralgorithmen beschränkt und im Internet frei verfügbar waren. Zusätzlich wurde dort das Konzept der Erläuterung von Datenstrukturen und Algorithmen mit UML- Klassen- und Kollaborationsdiagrammen erprobt. Die Ergebnisse einer einfachen Evaluation werden im Evaluations-Kapitel beschrieben.

4.2 VIDEA

Der erste praktische Einsatz von VIDEA erfolgte im Rahmen der Übungen zur Vorlesung “Einführung in die Informatik II” im Wintertrimester 2003 an der Universität der Bundeswehr München.
Dort wurde an zwei Tagen (mit Gruppen von ca. 15 Studenten) ein aus der Vorlesung bekanntes Thema jeweils ca. eine Stunde lang anhand einer interaktiven Übungsaufgabe vertieft. Parallel dazu wurde eine gleich große Kontrollgruppe mit einer Papier-Aufgabe des selben Lernstoffes konfrontiert. Die Betreuung und beobachtende Evaluierung erfolgte durch zwei Mitarbeiter des MuSofT-Projektes. Nach Beendigung der Aufgabe füllten beide Gruppen einen Evaluationsbogen aus, der den Lernfortschritt messen sollte. Die MuSofT-Gruppe füllte zusätzlich einen Fragebogen aus, der Selbsteinschätzung und Motivation vor und nach der Übung überprüfte.
Während der interaktiven Übung war es von den Studierenden u.a. gefordert, einen Beispielablauf von Einfüge-Operationen, die in flüssiger Animation alle Rotationen illustrierten, zu

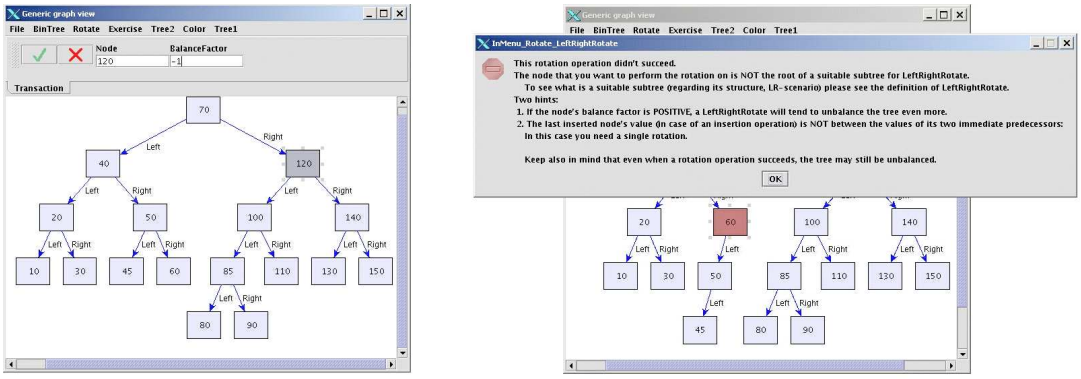


Abbildung 9: Screenshots der AVL-Instanz. Links: Der korrekte Balance-Faktor eines Knotens ist gesucht. Rechts: Eine falsche Rotation wurde gewählt.

beobachten, Höhe und Balancierungs-Faktor mehrerer innerer Knoten richtig anzugeben (siehe Abb. 9 - links), einen falsch einsortierten Knoten zu finden, den Baum durch Umhängen des fehlerhaften Knotens zu reparieren, die nun unbalancierte Stelle zu finden, durch manuellen Aufruf geeigneter Rotations-Operationen auch dieses Problem zu beheben (siehe Abb. 9 - rechts) und das Verhalten bei Einfügen weiterer Knoten zu testen. Die Ergebnisse werden im Detail im Evaluations-Kapitel beschrieben.
Die beobachtende Evaluation bei der AVL-Baum-Übung zeigte, dass die Studenten schnell lernten, das Tool zu bedienen, sie hatten Spaß und arbeiteten konzentriert. Andererseits lasen sie Fehlermeldungen nicht richtig und es bestand die Tendenz, das gewünschte Ergebnis unter Umgehung der geforderten Schritte zu erreichen, so wurde etwa versucht, den Baum durch Einfügen neuer Knoten zu rebalancieren anstatt durch explizite Rotationen.
Das Feedback der Studenten der MuSofT-Gruppe im Motivations-Fragebogen war: Macht Spaß, tolle Animation, ansprechendes Layout, intensiveres Lerngefühl Sie wünschten sich weitere Features (undo / redo, interne Knotennummern verbergen, zoom in /out, Speed-control), von denen ein Teil bereits implementiert ist (Speed-Control) und ein Teil gerade noch implementiert wird (undo/redo, interne Knotennummern verbergen).
Die zweite Übung behandelte Dijkstra's kürzeste Wegesuche, einen Algorithmus, der es durch eine Breitensuche erlaubt, kürzeste Pfade in einem (un-)gerichteten Graphen mit attributierten Kanten zu finden. In vielen Fassungen des Algorithmus wird mit drei Farben gearbeitet, die die Menge der Knoten in drei Gruppen aufteilen; diese Tatsache haben wir für die Visualisierung genutzt und mit den Ampelfarben rot-gelb-grün gearbeitet.

Übungsaufgaben waren u.a.

- das Finden kürzester Pfade zu gegebenen Knoten und
- Überlegen und nachmaliges Testen folgender Fragen:
- Welcher Knoten wird als erster betrachtet ?

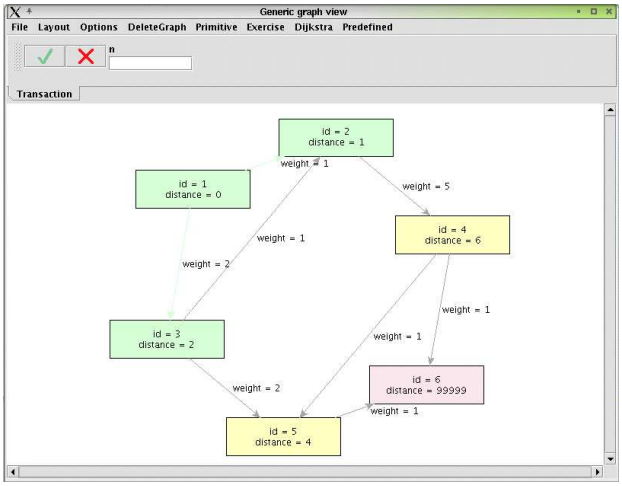


Abbildung 10: Screenshot der Dijkstra-Instanz. Die Knoten sind je nach Abarbeitungs-Stand des Algorithmus rot, gelb oder grün hinterlegt, die Kanten grau oder grün. Diese Nuancen sind leider im Schwarzweiß-Druck fast nicht zu sehen.

- Welche neue Farbgebung müsste nach dem nächsten Schritt entstehen, welcher Knoten ist nun der Ausgangsknoten des nächsten Schrittes ?
- Wann würde der Algorithmus genau stoppen, wenn der kürzeste Weg zu Knoten X (einem definierten Knoten) gesucht wäre ?

Unsere Beobachtung bei der Dijkstra-Übung war, dass die Studierenden durch die Aufgabenstellung besser geführt wurden als bei der AVL-Baum Übung. Zusätzliches Feedback war unter anderem: Schöne Einfärbefunktion, gute Überprüfungsfunktion. Für die Ergebnisse der inhaltlichen Evaluation verweisen wir wieder auf das Evaluations-Kapitel.
Unsere eigene Erfahrung mit VIDEA im Praxis-Einsatz ist, dass nach einem ersten Aufsetzen der notwendigen Umgebung (in unserem Fall auf einem Linux-Server, dessen Bilder über eine Windows-X-Emulation an die Windows-Clients verteilt wurde) der spontane Einsatz von VIDEA keinen echten Mehraufwand fordert, wirklich Spaß macht und oft die Erklärung der Detailfragen erleichtert. Schließlich ist es umständlich, verschiedenste Zustände etwa eines AVL-Baums an der Tafel durch abwechselndes Zeichnen und Wegwischen simulieren zu müssen.

4.3 Praktikum

Im Laufe des Praktikums galt es, drei Algorithmen in Java zu implementieren, die erst am Ende zu einem Gesamtalgorithmus zusammengefügt wurden. Im Einzelnen ging es dabei um:

1. Das Rucksackproblem: Es wurde eine Heuristik zur Bestimmung einer möglichst optimalen Beladung eines gegebenen Transportmittels mit beschränkter Kapazität mit Gü-

tern einer gegebenen Gütermenge, die i.a. die Kapazität des Transportmittels übersteigt, vorgestellt.

2. Minimal Spannende Bäume: Es wurde ein exakter Algorithmus zur Berechnung eines minimal spannenden Baumes zu einem gegebenen Graphen, der i.a. kein Baum ist, präsentiert.

3. Kürzeste Wege Suche: Es wurde ein exakter Algorithmus zur Berechnung des kürzesten Wegs zwischen zwei Knoten in einem Graphen erläutert.

Die einzelnen Algorithmen wurden umfassend dokumentiert und in Pseudocode angegeben. Zusammen mit der Information, an welchen Stellen die Implementierungen in das Rahmenwerk einzufügen sind, war die Umsetzung durch die Studenten kein Problem. Ferner stand den Studenten für jeden Algorithmus eine Visualisierung zur Verfügung, anhand derer sie sich das Problem besser verdeutlichen konnten. Dabei ging es nicht um die Visualisierung eines Algorithmus zur Lösung, sondern allein darum, dass die Studenten interaktiv durch Ausprobieren Lösungen suchen konnten. Dabei wurden sie stets über die Qualität ihrer Lösung durch die Visualisierungen informiert.
Für den Gesamtalgorithmus stand eine weitere Visualisierung zur Verfügung, die durch das Rahmenwerk gesteuert wurde. Anhand dieser konnten die Abläufe des Spiels gemäß des jeweiligen Gesamtalgorithmus nachvollzogen werden. Ein Punktestand sowie ein Info-Fenster gaben während des Wettkampfs Aufschluss über das Abschneiden der jeweiligen Gesamtlösung.
Die Ergebnisse einer durchgeführten Evaluation werden im Evaluations-Kapitel beschrieben.

5 Zusammenfassung

In unserem Teilprojekt 2.3 haben wir 3 Lerneinheiten im MuSofT-Portal der Öffentlichkeit zur Verfügung gestellt, die aus Foliensätzen, Flash-Animationen und Instanzen unserer Visualisierungsumgebung VIDEA bestehen. Alle diese Inhalte sind auf verschiedene Arten zusammenstellbar und teilweise hochgradig konfigurierbar und können so auf Lernstoff verschiedenen Detaillierungs- und Schwierigkeitsgrades angewandt werden - im Falle von VIDEA sogar auf andere Datenstrukturen.

6 Evaluierung

6.1 Präsenzveranstaltung

Die Evaluation der Vorlesung geschah durch die Verwendung eines Standardfragebogens, dessen Auswertung ungefähr folgendes Resultat ergab: Bei den Informatikstudenten herrschte eine positive Resonanz vor, die Elektrotechnikstudenten wurden selbst durch die einfachere Level1-Version überfordert und hätten gerne noch wesentlich mehr Animationen gehabt. Die Klausurergebnisse lassen sich mit denen der Vorjahre nicht vergleichen, da sich der Vorlesungsstoff stark geändert hatte.

Frage	MuSofT-Gruppe	Vergleichs-Gruppe
	richtige Antworten in %	
Ein AVL-Baum ist ein ...-...-Baum	74	91
Die Höhen zweier Teil-Bäume unterscheiden sich höchstens um ...	100	94
Wenn sich die Balance ändert, wird eine ...-Operation ausgeführt	94	94
Die Höhe des folgenden Baumes <1> ist...	94	75
Die Höhe des folgenden Baumes <2> ist...	94	88
Die Höhe des leeren Baumes ist...	65	75
Nach dem Einfügen eines Knotens sind höchstens ... Rotationen erforderlich	88	>0
Folgender Baum verletzt die Sortierungs-Eigenschaft. Bitte kurze Begründung, wo	82	100
Folgender Baum verletzt die Balancierungs-Eigenschaft. Bitte kurze Begründung, wo	94	88
Durch welche Rotation um welchen Knoten könnte der Baum aus der vorigen Frage rebalanciert werden ?	>47	>31
Geben Sie ein Beispiel eines Knoten-Wertes an, den man im folgenden Baum einfügen müsste, um eine sofortige Links-Rechts-Rotation zu erzwingen	71	75
Was passiert beim Löschen des Knotens 10 im AVL-Baum (Kurze Auflistung der Schritte):	>29	88
Inwieweit beeinflusst eine Rotations-Operation die Sortierungs-Eigenschaft des Baums ?	65	94
Wo werden AVL-Bäume Ihrer Meinung nach in der Praxis eingesetzt ?	47	56

Tabelle 1: Ergebnisse der Auswertung der Kontroll-Fragebögen zum inhaltlichen Verständnis beider Gruppen bezüglich AVL-Bäumen

6.2 VIDEA

Die Rahmenbedingungen der ersten zwei Evaluationen wurden bereits im Erfahrungsteil beschrieben. Wie dort schon erwähnt gab es an zwei Terminen jeweils eine MuSofT-Gruppe und eine Kontrollgruppe, wobei die MuSofT-Gruppe einen Motivationsfragebogen und beide Gruppen noch (zur Messung des Fortschritts) einen inhaltlichen Fragebogen ausfüllten. Uns war von Anfang an klar, dass die geringe Anzahl von Studenten, die in den Gruppen anwesend waren, keine signifikanten Ergebnisse und Interpretationen zulassen würden. Wir haben uns trotzdem entschieden, diese Art der Befragung und des Testes durchzuführen, weil es uns in der Arbeit während des Projektes wertvolles Feedback aus der Praxis bescherte und weil wir

Frage	MuSofT-Gruppe	Vergleichs-Gruppe
	richtige Antworten in %	
Wie ist der kürzeste Weg in folgendem Graphen zum Knoten 3 ?	89	71
zum Knoten 5 ?	89	79
In welcher Reihenfolge würden die Knoten des obigen Graphen im Rahmen der Berechnung des Dijkstra-Algorithmus (bis zum Ende) als 'U' besucht werden ?	100	79
Kann es passieren, dass der Algorithmus manche Knoten (und also auch Kanten) gar nicht besuchen muss, wenn die kürzesten Wege zu einem bestimmten Knoten gesucht sind ?	100	93
Bitte kurze Begründung:	>47	>7
Warum ist es nicht sinnvoll, die Prioritätswarteschlange in Ada als geordnete Liste zu implementieren ?	53	0
Nun soll der Algorithmus minimal erweitert werden, um bei der Suche nach einem kürzesten Pfad zu einem bestimmten Zielknoten Z bei sehr starker Vernetzung des Graphen nicht zu oft in die falsche Richtung zu laufen. Dazu soll jeder Knoten 2 zusätzliche Attribute XPosition und YPosition vom Typ Float bekommen, die zu Beginn initialisiert werden, und es sollen bevorzugt solche Kanten traversiert werden, die eher in Richtung auf die Koordinaten des Zielknotens Z hin führen. Welche Aktion in der Hauptschleife des Dijkstra-Algorithmus müsste dafür ein anderes Ergebnis liefern ?	63	36
Welche Funktion bzw. Aktion muss also geändert werden ?	>32	>14
Beschreiben Sie kurz die notwendige Änderung:	58	50
Ist es in Ihrer Lösung immer noch gewährleistet, dass der jeweils kürzeste Weg gefunden wird ?	47	21
Wenn ja, warum? Wenn nein, warum nicht ?	47	29

Tabelle 2: Ergebnisse der Auswertung der Kontroll-Fragebögen zum inhaltlichen Verständnis beider Gruppen bezüglich Dijkstra-Algorithmus

dadurch zumindest für uns selber ein Gefühl dafür bekamen, inwieweit wir auf dem richtigen Weg waren. Insofern sind die Ergebnisse in den Tabellen 1 und 2 mit Vorsicht zu genießen. Trotz allem haben wir uns über das deutlich bessere Abschneiden der MuSofT-Gruppe in unserer zweiten Evaluation gefreut, in der wir bereits die Erfahrung aus der AVL-Evaluation einbringen konnten. Die '>'-Zeichen, die an manchen Stellen in den Auswertungen auftauchen, bedeuten, dass die jeweilige Gruppe aus nachvollziehbaren Gründen (wie fehlende Bearbeitung einer zugehörigen Aufgabe, abweichende Erklärung des jeweiligen Tutors von der gewünschten Musterlösung oder halbwahre Antworten, die wir nicht eindeutig mit Punkten bewerten konnten) die jeweilige Frage besser beantworten hätte können als in den Prozent-Zahlen ausgedrückt.

6.2.1 AVL-Bäume

Die Auswertung der Fragebögen zeigte, dass beide Gruppen ein gutes Verständnis von AVL-Bäumen gewonnen hatten, wobei die MuSofT-Gruppe besser darin war, etwa die Höhe eines Beispiel-Baums anzugeben oder die Stelle, wo ein Beispiel-Baum unbalanciert ist. Die Stärken der Vergleichsgruppe lagen in Fragen wie die nach der Höhe des leeren Baums oder was beim Löschen eines Knotens passiert (ein Thema, das die MuSofT-Gruppe weitgehend ausgeklammert hatte). Auch wenn durch die sehr eng liegenden Resultate und die Kleinheit der Gruppen kein signifikantes Ergebnis hergeleitet werden kann, könnte man doch eine Stärke der MuSofT-Gruppe bei Fragen erahnen, die mit der visuellen Darstellung am Bildschirm zusammenhingen und eine Stärke der anderen Gruppe bei eher theoretischen Fragen. Ausnahmen davon gab es bei Teilthemen, die unterschiedlich intensiv geübt worden waren.
Die Auswertung des Motivationsfragebogens zeigte, dass die Studenten in unserer MuSofT-Gruppe die graphische Darstellung und die Animation ansprechend fanden, den Lernvorgang als intensiv empfanden, dass es Spaß machte und verständlich war.

6.2.2 Dijkstra

Die "beobachtende Evaluation" ergab (genau wie die Auswertung des Motivationsfragebogens) sehr ähnliche Erkenntnisse wie bei der AVL-Übung, allerdings eine genauere Bearbeitung der Vorgaben. Die Auswertung der Verständnis-Fragebögen zeigte ein leicht bis viel besseres Abschneiden der MuSofT-Gruppe bei den meisten Fragen.

6.3 Praktikum

Zur Evaluation wurde das Praktikum als Pflichtveranstaltung im Sommersemester 2003 zum ersten Mal durchgeführt. Es nahmen 180 Studenten teil (für das nächste Mal ist mit einer Teilnehmerzahl von ca. 250 Studenten zu rechnen). Diese wurden in Übungsgruppen mit ungefähr 24 Studenten eingeteilt. Innerhalb jeder Übungsgruppe bildeten 4 Studenten ein Team, welches gemeinsam an einer Lösung arbeitete. Neben den im Erfahrungsteil beschriebenen Aufgaben, lernten die Studenten Grundzüge der Softwaretechnik kennen und mussten diese auch umsetzen (Coding Guidelines, eXtreme Programming, JUnit Tests, JavaDoc, etc.). Das Praktikum war zeitlich in 5 Aufgabenblöcke untergliedert. In der Regel standen für jeden

Aufgabenblock 2 Wochen zur Verfügung. Für den Gesamtalgorithmus am Ende hatten die Studenten 3 Wochen Zeit.
Schließlich hatten die Studenten Gelegenheit, das Praktikum und dessen Inhalte über ein online-Formular annonym zu bewerten und ggf. Verbesserungsvorschläge zu machen. Die für diesen Bericht relevanten statischen Angaben sind in den Tabellen 3, 4, 5, 6 zusammengefasst.

sehr hoch	17%
hoch	34%
mittel	20%
wenig	12%
gar keiner	10%
keine Angabe	7%

Tabelle 3: Wie bewerten Sie den Nutzen der bereitgestellten Dokumentation? (93 Angaben)

sehr hoch	4%
hoch	42%
mittel	23%
schlecht	13%
miserabel	14%
keine Angabe	4%

Tabelle 4: Wie bewerten Sie die Qualität der bereitgestellten Dokumentation? (93 Angaben)

genial	20%
gut	30%
ganz nett	28%
schlecht	8%
miserabel	11%
keine Angabe	3%

Tabelle 5: Wie bewerten Sie die Idee des Bierbrauspiels? (93 Angaben)

Bewertung der Evaluation: Es hat sich gezeigt, dass das Praktikum die Studenten grob in zwei etwa gleich große Lager teilt: Eine Gruppe, die mit dem Praktikum sehr gut zurecht kam und es sehr gut bewertete, und eine Gruppe, die überfordert war und dem Praktikum somit eine schlechtere Bewertung gab.
Die Gruppe der Studierenden, die große Probleme mit dem Praktikum hatte, deckt sich in etwa mit der Gruppe von Studierenden, die auch in den vorangegangenen Vorlesungen “Allgemeine Informatik I und II” große Schwierigkeiten hatte.

spitze	5%
gut	27%
okay	23%
schlecht	15%
katastrophal	26%
keine Angabe	4%

Tabelle 6: Wie ist Ihr Gesamteindruck vom Praktikum? (93 Angaben)

Literatur

[And96] ANDY SCHÜRR: *Introduction to the Specification Language PROGRES*. In: *in: M. Nagl (ed.): Building Thightly-Integrated (Software) Development Environments: The IPSEN Approach, LNCS 1170*, Seiten 248–279. Springer Verlag Berlin, 1996.

[Ani01] `http://www.informatik.uni-siegen.de/db/animations.php3?lang=en`, Dez. 2001.

[Bub01] `http://olli.informatik.uni-oldenburg.de/fpsort/Animation.html`, Dez. 2001.

[CCA01] `http://www.cs.hope.edu/~alganim/ccaa/`, Dez. 2001.

[EE02] ERNST-ERICH DOBERKAT und GREGOR ENGELS (Herausgeber): *Ergebnisbericht des Jahres 2001 des Projektes "MuSofT – Multimedia in der SoftwareTechnik"*, Band 121 der Reihe *Softwaretechnik-Memo*. Lehrstuhl für Software-Technologie, Fachbereich Informatik, Universität Dortmund, März 2002.

[Ein03] http://www2.informatik.unibw-muenchen.de/Lectures/WT2002/INF2/index.html, Jan. 2003.

[JVi01] `http://www.ilog.com/products/jviews/`, Dez. 2001.

[KDE03] KLAUS ALFERT, ERNST-ERICH DOBERKAT und GREGOR ENGELS (Herausgeber): *Ergebnisbericht des Jahres 2002 des Projektes "MuSofT – Multimedia in der SoftwareTechnik"*, Band 133 der Reihe *Softwaretechnik-Memo*. Lehrstuhl für Software-Technologie, Fachbereich Informatik, Universität Dortmund, März 2003.

[Por03] `http://musoft.cs.uni-dortmund.de:8080/musoft/lehreinheiten.html`, Nov. 2003.

[PRO01] `http://www-i3.informatik.rwth-aachen.de/research/projects/progres/`, Dez. 2001.

[PS03] PETER ASCHENBRENNER und ANDY SCHÜRR: *Generating Interactive Animations from Visual Specifications*. In: *In 2003 IEEE Symposium on Visual Languages and Formal Methods*, Seiten 169–176, Auckland , New Zealand, Oct 2003.

[Sor01] http://www.db.fmi.uni-passau.de/Sommercamp2001/Unterlagen/SortDemo.html, Dez. 2001.

[UPG02] http://www-i3.informatik.rwth-aachen.de/upgrade/, Dez. 2002.

[yfi01] http://www.yworks.de/, Dez. 2001.

Der MuSofT-Abschlussbericht Arbeiten zu LE 3.1 “V-Modell” und LE 3.2 “Qualitätsmanagement”

Fritz Schmidt, Anita Böhm, Alexander Lurk,
Andreas Piater und Darko Sucic

Kurzfassung

Die Lerneinheiten 3.1 und 3.2 des Projektes MuSofT befassen sich mit Qualitätsmanagement (QM) bei der Entwicklung großer Softwaresysteme. Schwerpunkte sind prozessorientiertes (LE 3.1) und produktorientiertes (LE 3.2) QM. Als Produkte werden Softwaresysteme aus dem Ingenieurbereich mit stark simulationsorientierten Komponenten zugrundegelegt. Die Möglichkeiten multimedialen Arbeitens werden derart genutzt, dass die Lernmodule primär einführenden Charakter haben und durch Links mit Systemen verbunden werden, die mittels Tailoring auf aktuelle Projekte angepasst werden können. Dadurch erhalten die Lehrveranstaltungen verstärkt den Charakter von Erkundungsumgebungen in denen sich die Studierenden mit dem zu vermittelnden Stoff vertraut machen und entsprechend ihren Interessen ihre Kenntnisse vertiefen können. Die Lernmodule können darüber hinaus in den verschiedensten Kontexten wiederverwendet werden. Als Beispiel für eine besonders erfolgreichen Einsatz wird das Praktikum Simulation komplexer Systeme beschrieben. Mit diesem Praktikum gelang es uns nicht nur die Lehrinhalte auf attraktive Weise zu vermitteln, sondern auch selber zum Lerngegenstand zu machen. Das zu Lehrende bestimmte die Art zu Lehren und bot damit eine erste erfahrbare Bestätigung des Gelernten. Damit zeigt dieses Beispiel auch, wie sich durch den Einsatz multimedialer Elemente Lehrinhalte effektiver vermitteln lassen und welche Konsequenzen das für die universitäre Lehre haben kann.

Abstract

Learning units 3.1 and 3.2 of the project MuSofT deal with quality management (QM) in software engineering. Learning unit 3.1 concentrates on the process of software development. Learning unit 3.2 concentrates on testing. Products considered are primarily software systems which support engineers in performing their work. Therefore we include the development and testing of components for simulations based on mathematical models. The use of multimedia technologies allows to restrict the learning modules to introductions and to supplement these introductions through environments which allow students to explore software

engineering. As one consequence the learning modules can be reused in many different contexts. Links are provided to tools which can be tailored to handle concrete projects. They are supplemented by videos which explain the use of the tools. As an example how multimedia elements effect teaching we describe the exploration environment which we use for engineers to learn about developing simulations of complex systems.

Inhaltsverzeichnis

1 Einleitung

Die Lernmodule der Einheiten LE 3.1 “V-Modell” und LE 3.2 “Qualitätsmanagement” begreifen Software als technisches Produkt. Technische Produkte müssen Qualitätsforderungen genügen. Diese Forderungen müssen nicht nur aufgestellt, sondern auch erfüllt werden. Um dies zu erreichen, ist Qualitätsmanagement notwendig. Man unterscheidet prozessorientiertes (getrieben von Vorgehensmodell und darin beschriebenem Entwicklungsprozess) und produktorientiertes (getrieben von Qualitätsmerkmalen und Teststrategien) Qualitätsmanagement [ike].
Im Projekt wurden insgesamt 14 Lehrmodule entwickelt. Zielgruppe sind Ingenieure der Fächer “Angewandte Informatik”, “Maschinenbau” und “Simulation”. Die Lehrmodule können zum Teil einzeln verwendet und in andere Vorlesungen eingebaut werden oder aber als eigenständige Lerneinheit Verwendung finden. Sie sind zunächst für Präsenzveranstaltungen gedacht, sollen es aber auch erlauben, Kenntnisse aus vergangenen Vorlesungen aufzufrischen. Die ersten Versuche mit dem Einsatz einzelner Lehrmodule hat ergeben, dass bei Studierenden des Ingenieurwesens eine Kombination aus theoretischen Einführungen und praktischem Üben zu den besten Lehrerfolgen führt. Dem wurde in den Lehrveranstaltungen im WS 02/03

und verstärkt im WS 03/04 Rechnung getragen. Die Ergebnisse aus dem WS 03/04 werden im Frühjahr 2004 evaluiert und in einem Folgebericht dargestellt [Scha].
Die Inhalte der Lerneinheiten basieren primär auf unseren Erfahrungen bei der Durchführung von Qualitätsmanagement-Maßnahmen bei großen Industrieprojekten ([OS02], [DSV00], [Lee03], [Schb], [Schc]). Die Form orientiert sich dabei ebenfalls am Wunsch unserer Studierenden sich den Inhalt weniger über theoretische Präsentationen als vielmehr durch praktisches Erproben anzueignen. Die Präsentationen werden daher ergänzt um frei verfügbare Entwicklungsumgebungen, mittels derer im Rahmen von Praktika die QM Technologien erprobt und bei Studien- und Diplomarbeiten genutzt werden können. Weitere wichtige multimediale Elemente sind links auf weiterführendes Material, Multiple Chioce Elemente zur Überprüfung des Wissensstandes und dynamisierte Wissensvermittlungen in Form von Animationen, Filmen und Simulationen technischer Vorgänge.

2 Vorstellung der Lerneinheiten “V-Modell” und “Qualitätsmanagement”

Der Stoff wird in 2 Lerneinheiten mit insgesamt 14 Lermodulen präsentiert. Die Titel der Lernmodule findet man in Tabelle 1. Die Inhalte wurden in den Jahresberichten 2001 und 2002 beschrieben ([Dob02], [AD03], [ADE+02]) und können auch aus den Beschreibungen im MuSofT-Portal entnommen werden. Der aktuelle Stand der Entwicklung geht aus Tabelle 1 hervor und ist ausserdem auf unseren Webseiten([Schf], [Sche], [Schd], [Osm01], [BSS]) dokumentiert.
Im Sinne der MuSofT Typisierung sind die Lernmodule den Leveln 0 und 1 zuzuordnen, die Lernmodule haben also vor allem Übersichtscharakter und führen in die Themengebiete ein. Dabei stehen pro Lehrmodul ca. 50 Folien zur Verfügung, die teilweise untereinander vernetzt sind und häufig auf weiterführende Informationen verweisen. Da für eine Lerneinheit von 90 min in der Regel ca. 30 Folien ausreichend sind, ist der Lehrende gehalten entsprechend den Interessen der Lernenden Schwerpunkte zu setzen und eine Auswahl zu treffen.
Im Rahmen der Lerneinheiten von MuSofT setzen wir muldimediale Technologien eher spärlich ein. Insbesondere verzichten wir weitgehend auf selbsterklärende Animationen. (Ziel: Unterstützung in Präsenzveranstaltungen)
Unser methodisch - didaktisches Vorgehen ist zunächst sequentiell lehrgangsmäßig. Es wird ergänzt um eine hypermediale Erkundungsumgebung, die zum einen eine partielle Vertiefung und zum anderen eine problem- und prozessorientierte Annäherung an den Stoff ermöglicht.
Um dies zu erreichen, benötigen wir kleine, relativ gut gekapselte Lernmodule und ein Konzept zu ihrer Integration in größere Lerneinheiten. Die Kapselung der Lernmodule erreichen wir durch die schon erwähnte Beschränkung auf die Level 0 und 1. Die Vertiefung wird dann über Links zu Vorlesungen von Partnern aus dem Projekt MuSofT und zu Systemen, die sich auf aktuelle Probleme anwenden lassen, erreicht. Dies erfordert eine klare Trennung zwischen Lernmodulen und Anwendungsumgebungen auf der einen und Lehrinhalten und deren Präsentation auf der anderen Seite. Diese Trennungen ermöglichen es, die Lernmodule in verschiedenen Kontexten zu verwenden und durch Wahl der Beispiele oder gezielte Vertiefungen einzelner Unterthemen, an die Interessen und Möglichkeiten der Lernenden anzupassen. Zur

LE	LM	Titel	Status	Freigabe
3.1	1	Fehler und ihre Kosten	T1	Ende WS 03/04
3.1	2	Prozessqualität und Produktqualität	T1	Ende WS 03/04
3.1	3	Das Capability Maturity Modell	T1	Ende WS 03/04
3.1	4	Das V-Modell im Überblick	T1	Ende WS 03/04
3.1	5	V-Modell - Anwendungen	T1	Ende WS 03/04
3.1	6	Prozessbeschreibung SSDA	T1	Ende WS 03/04
3.1	7	Der Rational Unified Process im V-Modell	T1	Ende WS 03/04
3.2	8	ISO-Normen	T1	Ende WS 03/04
3.2	9	Prozessmodelle Übersicht	T1	Ende WS 03/04
3.2	10	Prüfung von SW-Komponenten - Einzeltests	T1	Ende WS 03/04
3.2	11	Prüfung von komponentenbasierten Systemen - Integrationstests	T1	Ende WS 03/04
3.2	12	Prüfung von Simulationsprogrammen - Funktionstests	T1	Ende WS 03/04
3.2	13	Testumgebungen - Abnahmetests	T1	Ende WS 03/04
3.2	14	Risikomanagement	T1	Ende WS 03/04

Tabelle 1: Stand der Arbeiten bei den Lernmodulen der Lerneinheiten LE 3-1 und LE 3.2 Status T1: Getestet in Iteration 1

Unterstützung der Selbstkontrolle des Lernfortschrittes der Lernenden werden multiple choice Fragebögen zur Verfügung gestellt.

2.1 Zusammenarbeit mit MuSofT Partnern

Im Rahmen des Projektes MuSofT haben wir vor allem mit den Partnern im Teilprojekt 3 “Querschnittsaufgaben im Entwicklungsprozess” zusammengearbeitet. Die in diesem Teilbereich entwickelten Lerneinheiten, decken wesentliche Aspekte der Systementwicklung ab und ergänzen sich daher. Entsprechend intensiv konnten wir an den Arbeiten der dort integrierten Projekte partizipieren. Das führte zunächst dazu, dass wir den Video über den Einsatz von CVS (Prof. Kelter Uni Siegen) mit in unser Praktikum Angebot aufnahmen. In einem 2. Schritt integrierten wir das Unified Process (UP) Lehrbuch (Prof. Doberkat Uni Dortmund), um schließlich das gesamte Praktikum im UP Modeller abzubilden und den Praktikumsfortschritt mit dem UP Tutor überwachen zu können. Damit wurde ein wichtiger Inhalt der Lerneinheiten selber Gegenstand des Lehrprozesses.

Die Abb. 1 zeigt das UP Praktikumsmodell mit seinen bisher durchgeführten, bzw. geplanten Zyklen. Man sieht daraus, dass alle drei Zyklen nach dem selben Schema abgelaufen sind. Die Unterschiede zwischen den Zyklen ergeben sich nur dann, wenn man die Verknüpfun-

gen zwischen den einzelnen Phasen und die diese Verknüpfungen beschreibenden Dokumente betrachtet.

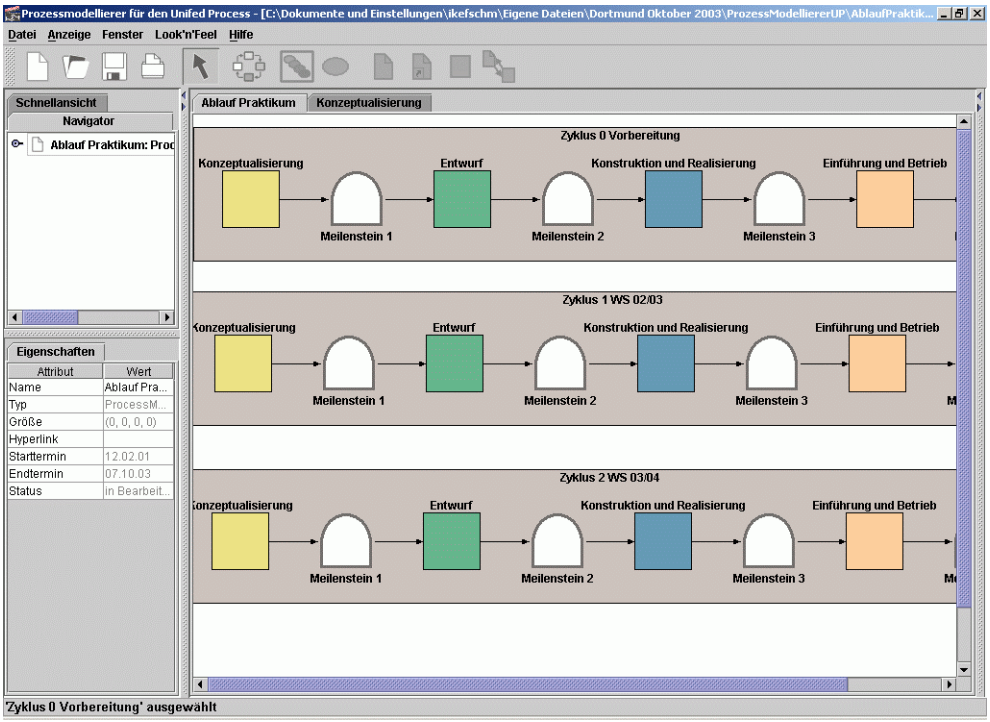


Abbildung 1: Modellierung des Praktikums Simulation komplexer Systeme im UM Modeller der Uni Dortmund

Neben diesem intensiveren Tausch von Erfahrungen und Materialien gab es auch mit anderen Projektpartnern außerhalb der regelmäßigen MuSofT Plenumsveranstaltungen hilfreiche Kontakte. Besonders zu erwähnen sind hier die Anregungen, die wir den Vorlesungen von A. Schürr in München und Darmstadt entnehmen durften [Schg], [Schh].

2.2 Zusammenarbeit mit Hochschulen im Raum Stuttgart

Besonders fruchtbar war die Zusammenarbeit mit der Hochschule für Medien Stuttgart und den Kollegen Göhner (Institut für Automatisierungs- und Softwaretechnik IAS) und Ludewig (Institut für Softwaretechnologie - IST - Abteilung Software Engineering).

Aus der Zusammenarbeit mit dem IAS resultierten nicht nur die schon erwähnten ersten Internet Präsentationen mit AIDA, sondern auch unser Ansatz Studien und Diplomarbeiten entsprechend den Vorgaben des V-Modells zu planen und durchzuführen. Darüber hinaus konnten wir manche Anregung aus den Vorlesungen des Institutes [Göha] in die MuSofT Lerneinheiten übernehmen.

Aus der Zusammenarbeit mit dem IST [Lud] resultiert der Zugang zu den Checklisten zur Überprüfung der Einhaltung der IST software development principles, was wesentlich zur Veranschaulichung des Ansatzes der testgetriebenen Entwicklung beiträgt. Mit der Hochschule der Medien und ihrer Ausgründung X_{term} sind wir über einen Werkvertrag zur Erstellung von Videos und eine Vereinbarung zum Tausch von Lerneinheiten verbunden. Basis der Zusammenarbeit ist das an der HdM entwickelte System [ter]. Das Programm erlaubt es, Audio, Video, Texte und Simulationen derart miteinander zu verbinden, dass optimale Lernerfolge damit erzielt werden können. Abb. 2 zeigt ein statisches Beispiel (Erklärung einer animierten Folie aus dem Video “Anleitung zum Einsatz von JUnit Tests”)

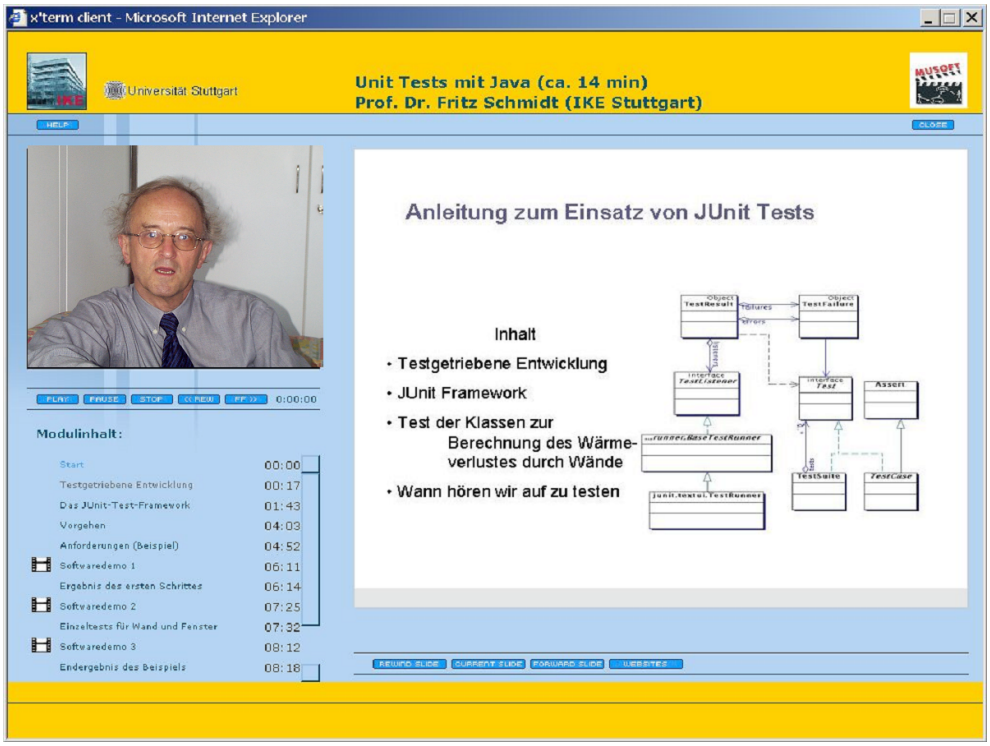


Abbildung 2: Screenschot vom Video “Anleitung zum Einsatz von JunitTest”, der in Zusammenarbeit mit der HdM Stuttgart entstanden ist

In der Tabelle 2 findet man eine Übersicht der in den Lerneinheiten verwendeten Videos.

2.3 Zusammenarbeit mit der Open Source Community

Für unsere Erkundungsumgebung greifen wir im Wesentlichen auf Open-Source-Produkte zurück.
Tabelle 3 gibt eine Übersicht über die wichtigsten Produkte.

Titel	Autoren	Webadresse
Anforderungsdefinition	IKE und HdM	Zugang über: http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/
Anleitung zum Einsatz von JunitTest	IKE und HdM	Zugang über: http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/
Anleitung zum Einsatz von Omondo	IKE und HdM	Zugang über: http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/
Anleitung zum Einsatz von Eclipse	IKE und HdM	Zugang über: http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/
Refactoring großer Systeme am Beispiel Gebäudesimulation	IKE und HdM	Zugang über: http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/
Einführung in das Concurrent Versioning System CVS	Uni Siegen	http://pi53.informatik.uni-siegen.de:8180/gCVS/index.html
Einführung in den Rational Unified Prozess	Rational USA	http://www.rational.com/media/products/clearcase/lifecycle.htm?from=inbrief&SMSESSION=NO
Erstellung einer Multimedia Produktion unter dem Werkzeug AIDA	IAS Uni Stuttgart	http://www.ias.uni-stuttgart.de/vortraege/rv-zukunft/html/index.htm

Tabelle 2: Videos für die Unterstützung der Lehre mittels der Kehreinheiten 3.1 und 3.2 und dem zugehörigen Praktikum

2.4 Zusammensetzung der Lernmodule zu Lerneinheiten

Als erste Konsequenz aus den Erfahrungen bei der Evaluierung multimedialer Lernmodule (siehe auch Abschnitt 5) haben wir die Vorlesung Simulation komplexer Anlagen und Systeme, in der wir den Studierenden Grundlagen des Softwareengineering im Hinblick auf Bau und Betrieb großer Systeme zur Simulation technischer Vorgänge vermitteln, recht radikal umgestellt.
Die Vorlesung bestand in den vergangenen Jahren aus ca. 14 Doppelstunden pro Semester. Sie wurde ergänzt durch ein Praktikum, das an drei Nachmittagen stattfand.
Die neue Struktur dreht diese Verhältnisse nahezu um. Die Vorlesung wurde auf 7 Doppelstunden zurückgefahren. (Modellierung, Wie reduzieren Ingenieure Komplexität, Standardisierung durch Prozessmodelle, Simplifizierung durch den UP, Spezialisierung und Wiederverwendung). Sie wird jetzt ergänzt durch ein Praktikum aus 10 Vortrags- und Übungssitzungen, für die sich die Studierenden zwei ganze Wochen freihalten. Dies entspricht einem Paradigmenwechsel in unserer Lehre. Das neue Motto lauted:

Erkundung statt Vermittlung

Elemente der Erkundungsumgebung sind

Produkt	Hersteller	Einsatzbereich	Bezugsquelle
V-Modell	Bund	Projekthandbuch	Zugang über `http://www.ansstand.de/frame.htm`
UP Modeller	Uni Dortmund	Prozessmodellierung	MuSofT Portal
UP Tutor	Uni Dortmund	Prozessüberwachung	MuSofT Portal
Eclipse	IBM Open Source	Software Entwicklung	http://www.eclipse.org
Omondo	Open Source	Modellierung	http://www.omodo.com
Junit Testframe	Open Source	Testgetriebene Entwicklung	http://www.junit.org
CVS	Open Source	Versions- und Konfigurations-management	http://www.cvshome.org
Renarch Framework	IKE	Gebäudesimulation	Zugang über: `http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/`

Tabelle 3: Produkte der Open Source Community, die im Rahmen von LE 3.1 und LE 3.2 zum Einsatz kommen

- Ausgewählte Lernobjekte in Form von Level 0 Lernmodulen
- Software Entwicklungsumgebung (SEU)
- Videofilme zur Einführung in die Tools der SEU
- Workflows für studentische Aktivitäten zur Softwareentwicklung
- Templates für wichtige Produkte
- Links zu den vollständigen Level 0 LMen der Lerneinheiten 3.1 und 3.2
- Links zu Level 1 LMen (wo aus anderen MuSofT Projekten verfügbar)
- Links zu weiterführenden Erfahrungsberichten und Dokumenten

Abb. 3 gibt eine Übersicht der für das Praktikum im WS 03/04 geplanten Versuche

2.5 Von der Theorie zur Praxis

Wir haben schon eingangs angedeutet (Abb. 1), dass wir das Praktikum selber im UP als iterativ inkrementellen Prozess abgebildet haben, der in jährlichen Zyklen abläuft. Damit gelang es uns nicht nur die Lehrinhalte auf attraktive Weise zu vermitteln, sondern sie auch selber zum Lerngegenstand zu machen. Das zu Lehrende bestimmte die Art zu Lehren und bot damit eine erste erfahrbare Bestätigung des Gelernten.

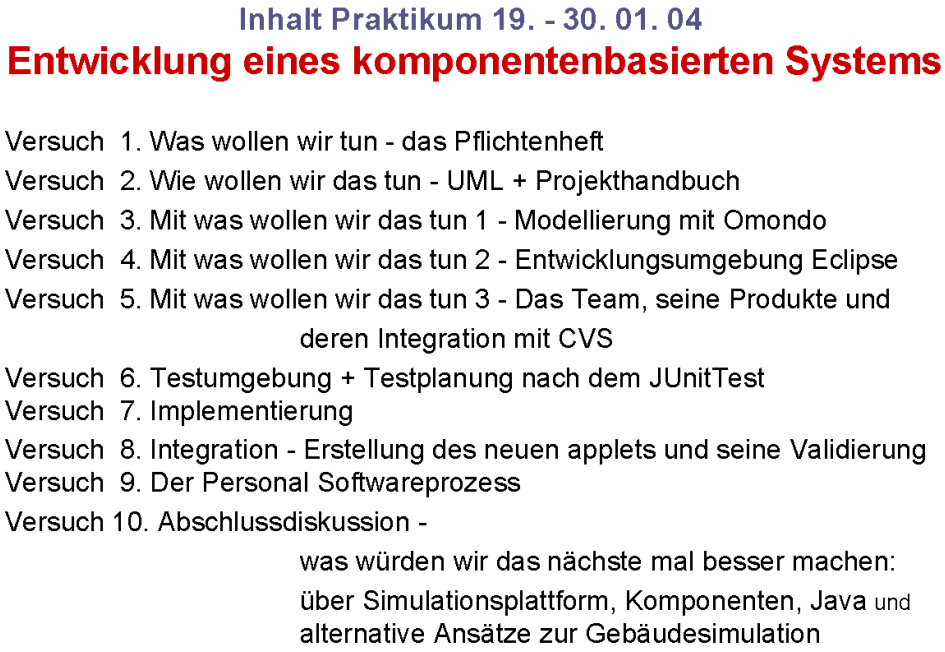


Abbildung 3: Versuchsplan für das WS 03/04

Abb. 4 zeigt den Zyklus wie er für das WS 03/04 geplant ist. Durch das geschilderte Vorgehen wird den Studierenden schnell klar, warum der zusätzliche Aufwand für Konfigurationsmanagement, Qualitätssicherung und Dokumentation sinnvoll ist. Insbesondere merken sie die Konsequenz, wenn Studierende des Vorsemesters nachlässig waren.
Durch das Arbeiten an einem längerlebigen Produkt wird auch vermittelbar, was es heißt, Anforderungen nicht richtig zu verstehen, nicht richtig umzusetzen und mit schlecht dokumentierten Schnittstellen weiterzugeben. Projektplanung, testgetriebenes Entwickeln und die Notwendigkeit des ständigen Verbesserns des eigenen Softwareentwicklungsprozesses werden so einsichtig.

3 Technische Realisierung

Bei der technischen Realisierung setzten wir Standardwerkzeuge sowohl zur Erstellung des Lehrmaterials als auch zum Zugriff darauf ein. Im wesentlichen sind das MS Office Tools und Web Browser. Mit der Plattform AIDA (Active Information Development Assistant des IAS der UNI Stuttgart [Göhb]) hatten wir eine Umgebung, die die Präsentation der Materialien für die Studierenden unterstützte.

- Sitzung 1-2 **Konzeption**:
  Definition der Aufgabe (Gebäudesimulation) und des Vorgehens
  - Einfluss der Modellierung - Anlegen des Pflichtenheftes
  - Tailoring des Vorgehensmodelles - Anlegen eines Projekthandbuches

- Sitzung 3-6 **Entwurf**:
  Detaillierung des Vorgehens
  - Einrichten der Projektumgebung
  - Erstellung der Spezifikation für eine Berechnungskomponente
  - Definition der Rollen
  - Definition der Aufgabenerfüllung

- Sitzung 7-8 **Implementierung**: Entwicklung einer Berechnungskomponente
  - Implementierung des Entwurfes
  - Integration in ein lauffähiges Programm

- Sitzung 9-10 **Einführung und Betrieb**:
  - Validierung des Programmes
  - Optimierung des Entwicklungsprozesses zwecks Qualitätsverbesserung

Abbildung 4: Gliederung der Praktikumsversuche nach den Phasen des Unified Prozess

AIDA wird im BMBF-Projekt ITO - Information Technology Online- weiterentwickelt und eingesetzt. Es ermöglicht die Erstellung von Multimedia-Vorlesungen, indem aus verschiedenen multimedialen Einzelelementen eine integrierte Vorlesungspräsentation erstellt wird. Über Internet kann der Lernende auf alle multimedialen Lehrmaterialien zugreifen, wobei er außer dem Web-Browser kein weiteres Werkzeug benötigt. Als Eingangsinformationen können u.a. mit Office-Werkzeugen erstellte Dokumente, Grafiken verschiedener Formate sowie Videos im AVI-Format verwendet werden. Unterstützende Dienste, wie etwa die Erstellung von Indizes, erleichtern dabei die Arbeit. Bei der Erstellung der Informationsapplikation werden die Eingangsinformationen ins HTML-Format (Hypertext Markup Language) transformiert und miteinander verknüpft. Leider ist dabei noch viel händische Arbeit nötig, so dass es zunehmend schwerer wurde, die Idee einer hörerangepassten Auswahl der Lehrmaterialien effektiv und zeitnah umzusetzen.

Wir haben auch gelernt Animationstechniken eher spärlich einzusetzen. Ein Zuviel an Animation ermüdet die Studierenden und lenkt ihre Konzentration statt auf die fachlichen Inhalte auf die speziellen Effekte.

Sehr stark setzen wir auf die Einbindung von Werkzeugen zur Erprobung des gelehrten Stoffes. Dies geschieht bevorzugt durch Links auf Demoumgebungen, Vorlesungen und Stoffsamm-

lungen anderer Kollegen und zu vertiefenden Texten. Details dazu haben wir im letzten Kapitel beschrieben.
Je mehr sich unser Lehrparadigma vom Vermitteln zum Erkunden entwickelte, desto mehr wurde uns klar, dass das neue Paradigma mit den Tools, die uns von AIDA angeboten wurden, nur unvollständig umgesetzt werden konnte. Eine Bewertung von AIDA im Rahmen der Vorarbeiten zum MuSofT Portal haben uns darin bestärkt, eigene Wege zu gehen. Die dafür entwickelte Architektur der IKE Lehr- und Lernumgebung wird in Abb. 5 gezeigt.
Die Lehr- und Lernumgebung stellt Lernobjekte unterschiedlichster Art (z.B. Dokumente, Präsentationen, Animationen, Videos, Simulationen, Datenbanken) als Dienstleistungen auf dafür geeigneten Servern, zur Verfügung. Dienstleistung meint dabei, dass nicht nur Inhalte sondern auch Methoden zu ihrer Präsentation angeboten werden. Der Zugriff auf die angebotenen Dienstleistungen kann sowohl über herkömmliche Klienten, die eigenständigen Anwendungen zugeordnet sind (im Bild GISterm aber auch Word oder Powerpoint), als auch über Web-Klienten (z.B. Browser, PDF-Reader usw.) erfolgen.
Web-Klienten sind schlanke Klienten. Um auf einem Web-Klienten einen Funktionsumfang in der Fülle einer eigenständigen Anwendung zur Verfügung zu stellen, wurde eine serverseitige Klientenschicht eingeführt. Diese stellt eine Mittelschicht basierend auf dem Publishing Framework Apache Cocoon [Coc] dar und erlaubt es, die für die Funktionen des Web-Klienten benötigte Logik in getrennten Komponenten zu realisieren. Der Web-Klient eröffnete durch die Möglichkeit Simulationen anzusprechen dem Lernenden viele neue Wege, das vermittelte Wissen durch Anwendung zu erkunden und zu vertiefen, ohne dazu spezielle Software installieren zu müssen.
Die Anwendung der Lehr- und Lernumgebung beginnt mit der Authentifizierung eines Benutzers an einem Klienten durch Eingabe seines Benutzernamens und Passworts. Damit ist grundsätzlichen Anforderungen des neuen Urheberrechtes Rechnung getragen. Fordert ein Benutzer eine Dienstleistung oder einen speziellen Inhalt an, wird nach der erfolgreichen Authorisierung anhand der Benutzerrolle die Dienstleistungen oder der Inhalt freigegeben und kann dann verwendet werden. Durch die Verwendung des Protokolles https ist es zusätzlich möglich, den Netzverkehr sicherer zu machen, als dies bei einer Verwendung anwendungsabhängiger Klienten in der Regel möglich ist. Diese beiden Vorteile gleichen in vielen Einsatzszenarien die Nachteile der Verwendung eines webbasierten Klienten aus.
In unserem Ansatz sind Lehren und Lernen kommunikative Prozesse kooperierender Partner. Kooperation muss aber organisiert - das meint vor allem koordiniert - werden. Deswegen arbeiten wir verstärkt an einer CSCW (Computer Supported Cooperative Work) Komponente. Sie wird es erlauben im Netz sowohl verteilte Projekte durchzuführen, als auch die Betreuung der Lerner durch die Lehrer zu intensivieren. Beides ist uns im Hinblick auf die geplante Einführung eines European Masters in Nuclear Engineering [NEP] wichtig.

4 Evaluierung

Zur Evaluierung der Lernmodule wurde ein mehrstufiges Verfahren eingesetzt. Nachdem die Lernmodule erstellt und vom Dozenten als fachlich richtig und dem Lernziel entsprechend bewertet werden, erfolgt im ersten Schritt eine Bewertung nach pädagogischen Gesichtspunkten. Dabei werden von einem unabhängigen Reviewer

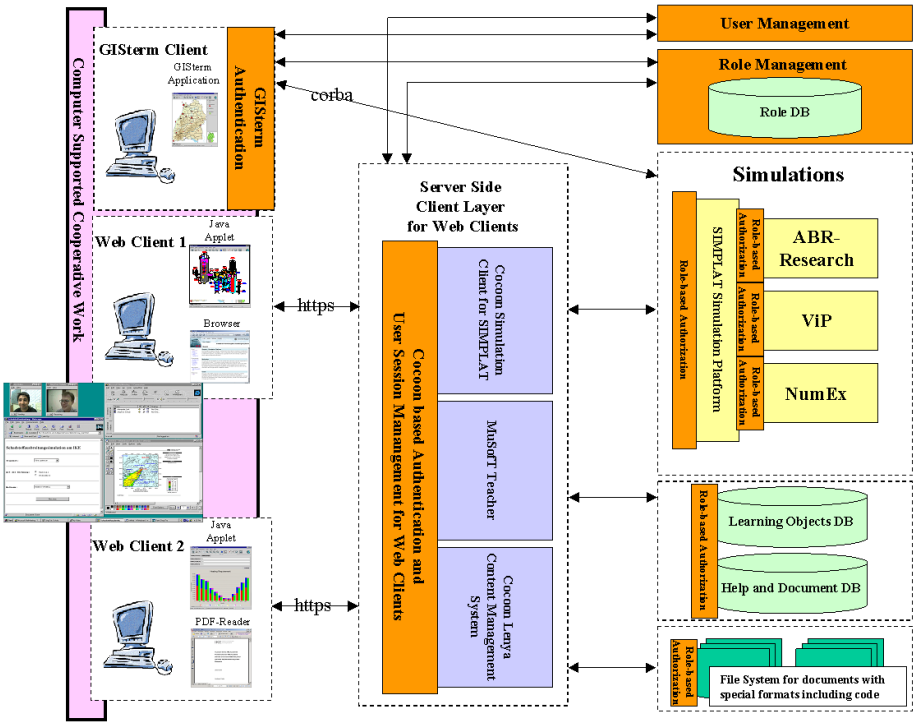


Abbildung 5: Die Lehr- und Lernumgebung des IKE in den Jahren nach MuSofT

- Formulierungen auf ihre Verständlichkeit überprüft
- Begriffe identifiziert, die gar nicht oder zu wenig erklärt werden und
- Beziehungen zu anderen Folien und Erklärungen aufgezeigt

Anschließend wurden dem aktuellen Lehrziel, mit dem der Lernmodul eingesetzt wird, entsprechende Fragen formuliert und zu jeder Frage eine html Seite generiert, über die die Studierenden ihr Verständnis an Hand unterschiedlicher Antwortalternativen überprüfen können. Beispiele dafür findet man auf der schon erwähnten Lehre homepage des IKE unter den dort veröffentlichten MuSofT Lernmodulen.
Sind die Lehrmaterialien so an die Ziele einer aktuellen Lehrveranstaltung angepasst, so werden sie dort eingesetzt und von den Studierenden bewertet. Im Maschinenbau ist die Zahl der Studierenden, die Vorlesungen aus dem Bereich der angewandten Informatik hören, zur Zeit eher gering. Wir haben daher auf statistische Erhebungen verzichtet und stützen die Bewertungen auf direkte Gespräche mit den Studierenden. Dabei haben sich vor allem zwei Dinge ergeben, die unsere Lehre entscheidend verändern.
Die erste Erfahrung ist die, dass die Studierenden zunächst mit einer recht allgemein gehaltenen und dafür leichter verständlichen Einführung in den Themenbereich zufrieden sind. Es ist

ihnen dann aber hilfreich, wenn sie sich gezielt in Teilbereichen, die sie besonders interessieren oder die sie für Arbeiten in anderem Umfeld benötigen, vertiefend weiterbilden können. Diese Weiterbildung erfolgt ähnlich wie bei Studien- und Diplomarbeiten weitgehend auf eigene Initiative. Links zu weiterführenden Angeboten sind daher erwünscht und werden auch genutzt. Die zweite Erfahrung ist die, dass die Studierenden die Übersichten aus den einführenden Vorlesungen durch eigenes Üben am Rechner vertiefen wollen. Es ist ihnen eher langweilig zu erfahren, wie man in der Theorie ein Vorgehensmodell auf ein abstraktes Problem zuschneidet, und sehr spannend, dies an Hand einer Aufgabe selbst durchzuführen. Praktika, die solches Üben ermöglichen, sind daher stärker als die Vorlesungen frequentiert. Dies nicht zuletzt auch deshalb, weil man sich ja den Vorlesungsstoff auf Grund der gut aufbereiteten Unterlagen im Netz selbst aufbereiten kann und es bei Bedarf möglich ist, sich weitere Informationen selbst zu beschaffen. Dies deutete sich schon nach dem ersten Praktikum, das im WS 01/02 durchgeführt wurde, an und verstärkte sich im Praktikum im WS 02/03.

5 Erfahrungen

Das Vorhaben hatte zum Ziel, die Entwicklung zweier Lehreinheiten und sie unterstützender Materialien, mit deren Hilfe Softwareengineering Studierenden der Ingenieurwissenschaften als Ingenieurdisziplin erfahrbar gemacht werden kann. Unsere Erwartungen an das Projekt waren von vorneherein sehr hoch, erhofften wir uns als Ingenieure doch eine nachhaltige Befruchtung unserer softwaretechnischen Lehranstrengungen durch die Vertreter der Informatik. Die Erfahrungen lassen sich wie folgt zusammenfassen:

- Die Bewertung der Lernmodule erfolgte über
  - Einzelgespräche
  - Bewertung des Engagements der Teilnehmer
  - Bewertung der erzielten Ergebnisse
- Dabei haben wir folgende positiven Erfahrungen gemacht
  - Zufriedenheit der Lernenden wurde gesteigert
  - Engagement wesentlich höher als bei Vorlesung
  - Hoher Einsatz um nicht nur Musskriterien sondern auch Wunschkriterien des Lastenheftes zu erfüllen
  - Ergebnisse übertrafen Erwartungen wesentlich
- Über die Potentiale des multimedialen Ansatzes haben wir gelernt
  - Theoretische Möglichkeiten können umgesetzt werden
  - Neue Lehrmethoden werden von den Studierenden angenommen
  - Es ist möglich Lehrgegenstand und Lehrinhalt zu einer Einheit zu verbinden, die ohne die multimedialen Techniken nicht möglich gewesen wäre.

Am Ende des Projektes lässt sich nun feststellen, dass diese Erwartungen nicht nur erfüllt, sondern weit übertroffen wurden. Als Konsequenz des Projektes hat sich im Laufe des Vorhabens eine dramatische Veränderung im didaktischen Konzept unserer Lehrveranstaltungen ergeben. Auf Grund der neuen Möglichkeiten einer multimedialen Unterstützung gelang es, den Lehrgegenstand (Vorgehen bei qualitätsgetriebenem Softwareengineering) mit der Durchführung der Lehrveranstaltung aufs innigste zu verknüpfen und so den Lehrgegenstand selber zum Rahmen zu machen, innerhalb dessen er gelehrt wird.

6 Zusammenfassung

Die Lerneinheiten 3.1 und 3.2 des Projektes MuSofT befassen sich mit Qualitätsmanagement (QM) bei der Entwicklung großer Softwaresysteme. Schwerpunkte sind prozessorientiertes (LE 3.1) und produktorientiertes (LE 3.2) QM. Als Produkte werden Softwaresysteme aus dem Ingenieurbereich mit stark simulationsorientierten Komponenten zugrundegelegt. Die Möglichkeiten multimedialen Arbeitens werden derart genutzt, dass die Lernmodule primär einführenden Charakter haben und durch Links mit Systemen verbunden werden, die mittels Tailoring auf aktuelle Projekte angepasst werden können. Dadurch erhalten die Lehrveranstaltungen verstärkt den Charakter von Erkundungsumgebungen in denen sich die Studierenden mit dem zu vermittelnden Stoff vertraut machen und entsprechend ihren Interessen ihre Kenntnisse vertiefen können. Außerdem können die Lernmodule in den verschiedensten Kontexten wiederverwendet werden.

Das Vorhaben hatte zum Ziel, die Entwicklung zweier Lehreinheiten und sie unterstützender Materialien, mit deren Hilfe Softwareengineering Studierenden der Ingenieurwissenschaften als Ingenieurdisziplin erfahrbar gemacht werden kann. Unsere Erwartungen an das Projekt waren von vorneherein sehr hoch, erhofften wir uns als Ingenieure doch eine nachhaltige Befruchtung unserer softwaretechnischen Lehranstrengungen durch die Vertreter der Informatik.

Am Ende des Projektes lässt sich nun feststellen, dass diese Erwartungen nicht nur erfüllt, sondern weit übertroffen wurden. Als Konsequenz des Projektes hat sich im Laufe des Vorhabens eine dramatische Veränderung im didaktischen Konzept unserer Lehrveranstaltungen ergeben. Auf Grund der neuen Möglichkeiten einer multimedialen Unterstützung gelang es, den Lehrgegenstand (Vorgehen bei qualitätsgetriebenem Softwareengineering) mit der Durchführung der Lehrveranstaltung aufs innigste zu verknüpfen und so den Lehrgegenstand selber zum Rahmen zu machen, innerhalb dessen er gelehrt wird.

Durch dieses Vorgehen konnten wir das Interesse am Inhalt und seiner Präsentation wesentlich steigern. Dies zeigte sich zunächst in einem sprunghaften Ansteigen der Zahl der Teilnehmer, dann aber vor allem in deren Präsenz, die nur selten unter 90% absank. Außer in unseren eigenen Lehrveranstaltungen konnten wir die in MuSoft gemachten Erfahrungen sowohl in verschiedene Projekte der Universität Stuttgart einbringen ([Stu] und [ssoP]) als auch zur Grundlage einer erfolgreichen Teilnahme an Anträgen im 6. Rahmenprogramm ([NEP] und [SAR]) verwenden.

Literatur

[AD03] ALFERT, K. und E.-E. DOBERKAT: *MuSofT-Bericht Nr. 2*. Memo Nr. 133 des Lehrstuhls für Softwaretechnologie, Fachbereich Informatik, Universität Dortmund, 133:58–68, 2003. Engels, G. (Hrsg.). ISSN: 0933-7725.

[ADE+02] ALFERT, K., E.-E. DOBERKAT, G. ENGELS, M. LOHMANN, J. MAGENHEIM und A. SCHÜRR: *Proc. SEUH 8 Software Engineerung im Unterricht der Hochschulen*. In: *MuSofT - Multimedia in der SoftwareTechnik*, Seiten 70–8. dPunkt-Verlag, 2002. Weber-Wulff (Hrsg.). ISBN: 3898642011.

[BSS] BÖHM, A., F. SCHMIDT und D. SUCIC. Video1 "Anforderungsdefinition" Zugang über: `http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/`.

[Coc] COCOON, APACHE. Apache Cocoon Homepage `http://cocoon.apache.org/`.

[Dob02] DOBERKAT, E.-E.: *MuSofT-Bericht Nr. 1*. Memo Nr. 131 des Lehrstuhls für Softwaretechnologie, Fachbereich Informatik, Universität Dortmund, 121, 2002. Engels, G. (Hrsg.). ISSN: 0933-7725.

[DSV00] DOBERKAT, E.-E., F. SCHMIDT und C. VELTMANN: *Re-engineering the German integrated System for measuring and assesing environmental radioactivity*. Environmental Modelling & Software, 15:267–278, 2000.

[Göha] GÖHNER, P. Lehrveranstaltungen am IAS `http://www.ias.uni-stuttgart.de/lehre/lehrveranstaltungen/index.html`.

[Göhb] GÖHNER, P. Neue Medien in der Aus- und Weiterbildung. Folien zu Vorträgen., `http://www.ias.uni-stuttgart.de/vortraege/rv-zukunft/html/index.htm`.

[ike] Allgemeine Informationen durch die ISO unter `http://www.iso.ch/iso/en/iso9000-14000/index.html`.

[Lee03] LEEB, H.: *IMIS-Migration*. In: *Projekt AJA, Anwendung JAVA-basierter Lösungen und anderer leistungsfähiger Lösungen in den Bereichen Umwelt, Verkehr und Verwaltung - Phase IV*, 2003. R. Mayer-Föll, A. Keitel, W. Geiger (Hrsg.).

[Lud] LUDEWIG, J. Software testing principles `http://www.iste.uni-stuttgart.de/se/service/checklists/download/Test_Pr%inciples.html`.

[NEP] NEPTUNO. NEPTUNO Nuclear European Platform for Training & UNiversity Organisations, Projekt im Rahmen des FP 6 unter negotiation, Nov. 2003.

[OS02] OBRECHT, R. und F. SCHMIDT: *Erneuerte Kernreaktorfernüberwachung in Baden-Württemberg*. In: *Projekt AJA, Anwendung JAVA-basierter Lösungen und anderer leistungsfähiger Lösungen in den Bereichen Umwelt, Verkehr und Verwaltung - Phase III*. Forschungszentrum Karlsruhe, Wissenschaftliche Berichte FZKA-6777, 2002. R. Mayer-Föll, A. Keitel, W. Geiger (Hrsg.); `http://www.lfu.baden-wuerttemberg.de/lfu/uis/aja3/index1.html`.

[Osm01] OSMANKOVIC, ALMA: *Möglichkeiten computerunterstützten Lernens im Bereich der beruflichen Weiterbildung - Untersuchung eines Lehrgangs des Technischen Hilfswerkes*, 2001. Die Arbeit ist unter `http://www.ike.uni-stuttgart.de/~www_wn/lehre` verfügbar.

[SAR] SARNET. SARNET: PROPOSAL OF NETWORK OF EXCELLENCE FOR A SUSTAINABLE INTEGRATION OF EUROPEAN RESEARCH ON SEVERE ACCIDENT PHENOMENOLOGY AND MANAGEMENT Projekt im Rahmen des FP 6 unter negotiation, Nov. 2003.

[Scha] SCHMIDT, F. Abschlussbericht zum Vorhaben MuSofT - Multimdia in der Software Technik Teil 2, Förderkennzeichen 01NM098C, erscheint 2004.

[Schb] SCHMIDT, F. European information/training on renewable energy in architecture, Abschlussbericht zum EU-ALTENER Contract 4.1030/Z/95-053 Deutsche Fassung gefördert von der DBU Az. 09394 24/2 IKE 4-FB-DBU 3 Juli 1999.

[Schc] SCHMIDT, F. Vorstudie für die Entwicklung PC-unterstützter Lernverfahren zur Akzeptanz- und Effektivitätssteigerung der Ausbildung im THW, Abschlussbericht zum Vorhaben 1025/99/1/BZS-XA2 IKE FB-THW 1 Juni 2001.

[Schd] SCHMIDT, F.: *WWW-Seite zu MuSofT mit allen Lerneinheiten und ergänzenden Lehrmodulen*. Zugang über: `http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/`.

[Sche] SCHMIDT, F.: *WWW-Seite zum Praktikum "Software engineering komplexer Systeme" an der Universität Stuttgart, Fakultät Maschinenbau*. Zugang über: `http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/`.

[Schf] SCHMIDT, F.: *WWW-Seite zur Vorlesung "Simulation komplexer Systeme" an der Universität Stuttgart, Fakultät Maschinenbau.* Zugang über: `http://ikeas1.ike.uni-stuttgart.de/~www_wn/lehre/`.

[Schg] SCHÜRR, A. WWW-Seite zur Vorlesung "Einführung in die Informatik II" an der Universität der Bundeswehr München, Fakultät für Informatik, `http://www2.informatik.unibw-muenchen.de/Lectures/WT2002/INF2/INF2.html%`.

[Schh] SCHÜRR, A. WWW-Seite zum Software-Praktikum an der TU Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, `http://www.es.tu-darmstadt.de` (Menüpunkt Lehre/SP).

[ssoP] PROJEKT SELF-STUDY ONLINE. NumEx-pDgl in der Fakultät 7 Details unter http://www.campus-online.uni-stuttgart.de/self-study/.

[Stu] STUTTGART, UNIVERSITÄT. Im April 2001 wurde das Programm 100-online an der Universität Stuttgart ausgeschrieben. Die Absicht war - auch der Name ist Programm - 100 Lehrveranstaltungen unter Einsatz multimedialer Techniken didaktisch neu aufbereitet im Intranet der Universität zur Verfügung zu stellen und so einen Beitrag zur Verbesserung der Lehr- und Lernsituation zu leisten. Das IKE beteiligte sich mit insgesamt 4 Teilprojekten (siehe unter Fakultät Energietechnik) an diesem Programm. Die einzelnen Informationen finden Sie auf den Projektseiten der Universität Stuttgart (www.uni-stuttgart.de/100online).

[ter] TERM. term Homepage: http://term.hdm-stuttgart.de/term2/login.jsp.

Der Abschlussbericht des MuSofT-Teilprojekts 3.3

Corina Kopka, Jens Schröder

Dieser Bericht gibt die abschließenden Ergebnisse der MuSofT-Lerneinheit 3.3 wieder, die als Thema die Lehre des Unified Software Process hat.

Inhaltsverzeichnis

1 Einleitung

Die Lerneinheit 3.3 *Durchführung von Softwareprojekten mit dem Unified Process* wurde im Rahmen des Projekts *MuSofT - Multimedia in der SoftwareTechnik* entwickelt und für die Lehre der Softwaretechnik eingesetzt. Sie bietet Unterstützung bei der Durchführung von Softwarepraktika. In Abgrenzung zu Programmierpraktika, bei der die Implementierung (z.B. von Algorithmen) im Vordergrund steht, handelt es sich hier um die Unterstützung umfassender Softwaretechnikpraktika, in denen Softwareentwicklungsprojekte durchgeführt werden. Die Studierenden sollen in Arbeitsgruppen neben dem Umgang mit einer Entwurfsnotation und mit den unterstützenden Werkzeugen, aufgrund einer konkreten Aufgabenstellung Software entwickeln. Der dabei zu beschreitende Entwicklungsprozess ist im Rahmen dieser Lerneinheit zu erlernen. Dazu wird der Ablauf eines Projektes auf der Grundlage des Unified Process visualisiert. Desweiteren kann ein Teil der Lerneinheit nicht nur zum Selbststudium, sondern

auch durch den Lehrenden (z.B. in Vorlesungen zur Softwaretechnik oder in vorbereitende Vorlesungen innerhalb des Softwarepraktikums) eingesetzt werden.
In der Industrie hat sich die iterative Softwareentwicklung durchgesetzt, da damit Risiken frühzeitig erkannt werden können und Fehler in frühen Phasen mit weniger Aufwand als in klassischen nichtiterativen Entwicklungsprozessen behoben werden können. Für eine praxisnahe Ausbildung wurde daher der Unified Process als Vertreter iterativer Softwareentwicklungsprozesse gewählt.

2 Vorstellung der Lerneinheit

Während Softwaretechnikvorlesungen eher darauf ausgerichtet sind, softwaretechnisches Wissen im Sinne eines Informationsvermittlungsprozesses weiterzugeben, eröffnen Softwarepraktika u.a. die Möglichkeit, softwaretechnisches Wissen im Zusammenhang eines konkreten Projektes anzuwenden und Erfahrungen hinsichtlich der Kommunikation und Zusammenarbeit in einem Projektteam zu sammeln.
Schwerpunkt in Softwarepraktika bilden i.d.R. der Einsatz bereits bekannter Sprachen und Methoden für Analyse, Entwurf, Implementierung und Test. Darüberhinaus werden aber auch genaue Kenntnisse über Softwareentwicklungsprozesse benötigt, die oft nicht explizit gelehrt werden. Typischerweise werden Prozessmodelle vorgegeben [Sch01], so dass keine der möglichen Prozessmodellalternativen betrachtet werden. In der MuSofT-Lerneinheit *Durchführung von Softwareprojekten mit dem Unified Process* hingegen wird der Entwicklungsprozess anhand des Beispiels *Unified Process* (UP) [JBR98, Kru99] zum Lerngegenstand erhoben [KA02].
Der Unified Process ist idealerweise für große Projekte geeignet und erweist sich daher als problematisch für die Anwendung in kleineren Praktika, wie etwa typischen Programmierpraktika. Die MuSofT-Lerneinheit unterstützt daher umfassendere Softwaretechnikpraktika und Projektgruppen, in denen kleinere und mittelgroße Softwareentwicklungsprojekte durchgeführt werden.
Die MuSofT-Lerneinheit *Durchführung von Softwareprojekten mit dem Unified Process* legt einen Schwerpunkt darauf, Studierenden die Gelegenheit zu geben, Gelerntes über den Softwareentwicklungsprozess konstruktiv anzuwenden und Erfahrungen zu sammeln. Dazu wird von den Studierenden ein solcher Entwicklungsprozess geplant. Darüberhinaus wird in einem konkreten Projekt auf Basis des geplanten Entwicklungsprozesses Software entwickelt, so dass Studierende Gelegenheit bekommen, ihre Planung zu evaluieren, über ihre Entscheidungen in der Planung zu reflektieren und Konsequenzen aufgrund ihrer neuen Erfahrungen zu ziehen. Lernen erfolgt also durch reflektiertes Handeln und folgt damit konstruktivistischen didaktischen Ansätzen. Diese Art von Lernprozess erinnert nicht zufällig an den *Personal Software Process* [Hum96], bei dem die Entwickler empirische Daten über ihre eigenen Arbeiten sammeln, um sie später statistisch zu analysieren und damit nachfolgende Prozesse besser planen zu können.
Die MuSofT-Lerneinheit 3.3 unterstützt die Studierenden in der Projektplanung auf Basis des Unified Process und begleitet sie in der Projektsoftwareentwicklung. Im Vordergrund der Planung stehen nicht die Vorgabe eines verfügbaren Budgets oder eine Kosten- und Aufwandsschätzung, sondern die Identifizierung sinnvoller Aktivitäten des Entwicklungsprozesses, ihre

sorgfältige zeitliche Einordnung innerhalb eines zeitlichen Rahmens, die Planung von sinnvollen Iterationen, also insbesondere auch die richtige Anwendung des Unified Process als generisches Modell für ihre spezifischen Projektzwecke. Das Lernen erfolgt im Unterschied zu dem Ausbildungskonzept aus Siegen (Lerneinheit 3.4) nicht durch die Durchführung werkzeugunterstützter Simulationen, sondern durch Beobachtung und Analyse während der Durchführung eines Entwicklungsprozesses und durch Rückschlüsse bzgl. der ursprünglichen Planung. Somit wird das Lernen des Planens eines Softwareprojekts mit dem Lernen Software zu entwickeln integriert.
Unsere Lerneinheit wurde entsprechend der bisher dargelegten Überlegungen mehrstufig konzipiert und in die folgenden Lernmodule strukturiert:

Lernmodul *Unified Process*

Dieses Lernmodul ist ein multimediales Lehrbuch, mit dem den Studierenden neben allgemeinen Vorgehensmodellen der Unified Process vorgestellt wird. Im Lernmodul werden multimediale Techniken wie die Verlinkung von Inhalten und der Einsatz von Animationen angewendet.
Das Lehrbuch untergliedert sich in drei Bereiche: In einem Vorlesungsteil wird in Folienform der UP eingeführt. Dieses kann von einem Dozenten zum Lehren des Unified Process innerhalb einer Vorlesung eingesetzt werden. Der zweite Teil beschäftigt sich mit den theoretischen Grundlagen des UP. Im dritten Teil, dem so genannten Praxisbeispiel, wird anhand einer beispielhaften Aufgabenstellung für ein Softwareprojekt dessen Planung und Durchführung mit dem UP ausführlich dokumentiert. Studierende können somit im Selbststudium selbstgesteuert den Ablauf eines Musterprojekts betrachten. Die Darstellung des Projektablaufs soll Aktivitäten, Artefakte und Rollen berücksichtigen.
In Abb. 1 ist beispielhaft eine Seite des Lehrbuchs zu sehen. Es wurde ein Navigationsrahmen entwickelt, in dem die Inhalte des UP-Lehrbuchs eingebettet sind. Er ermöglicht es den Studierenden, einfach und übersichtlich durch die Inhalte des Lehrbuchs zu navigieren. Dazu stehen ihnen ein (aufklappbares) Inhaltsverzeichnis, hierarchische und sequentielle Vor- und Zurück-Funktionen und eine History zur Verfügung. Die drei inhaltlichen Teile des Lehrbuchs, insbesondere die theoretischen Grundlagen und das Praxisbeispiel, sind durch Links inhaltlich eng miteinander in Beziehung gesetzt, um den Bezug der theoretischen Inhalte mit ihrer Anwendung - und umgekehrt - deutlich zu machen. Desweiteren können so genannte geführte Touren definiert werden, in denen ausgewählte Inhalte aus dem Lehrbuch in einer strikt sequentiellen Folge präsentiert werden.
Auf diese Art und Weise haben die Studierenden mehrere Möglichkeiten, sich den Inhalt anzueignen und zu vertiefen. Sie werden sowohl dabei unterstützt, selbstständig den Inhalt des Lehrbuchs in einer von ihnen festzulegenden Reihenfolge frei zu erkunden, als auch dies auf thematisch vorausgewählten Navigationspfaden zu tun.

Lernmodul *Projekte maßschneidern* (engl. Tailoring)

Um die Studierenden bei der Definition eines Prozessmodells nach dem UP zu unterstützen, wird mit diesem Lernmodul ein grafischer Editor bereitgestellt, mit dem ein Prozess für ein konkretes Softwareprojekt instanziiert werden kann.

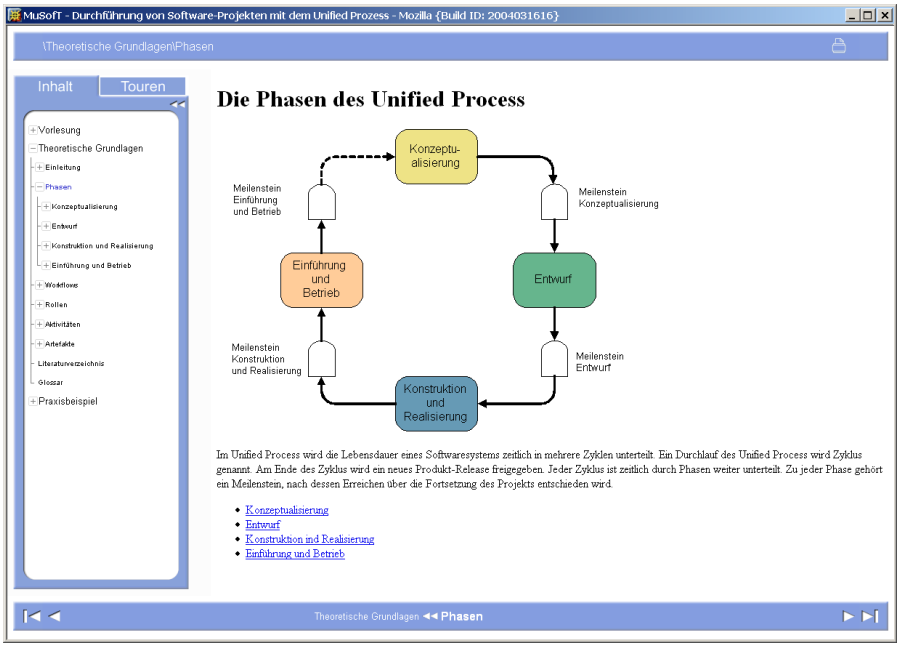


Abbildung 1: Beispielseite des Lernmoduls *Unified Process*

Dieser Prozessmodellierer nimmt dabei eine Anpassung des UP an die besonderen Bedürfnisse eines Softwarepraktikums vor: er erlaubt es, einen Prozess zu instanziieren, der sich auf die fünf für die Softwarentwicklung wesentlichen Kernarbeitsprozesse (engl. Workflows) konzentriert, von der Erfassung der Anforderungen bis zum Testen. Arbeitsprozesse, die für den Einsatz des UP in einer Lehrveranstaltung weniger von Bedeutung sind, wie etwa die Geschäftsprozessmodellierung und die unterstützenden Prozesse finden keine Berücksichtigung. Desweiteren kann der Modellierer mit denen im Projekt auftretenden Rollen konfiguriert werden. Beispiele für konkrete Projektabläufe aus Musterprojekten werden mit ihren Besonderheiten zur Hilfestellung vorgestellt.

Bei der Instanziierung wird der Prozess für das durchzuführende Projekt gemäß den Vorgaben des UP strukturiert und die einzelnen Aktivitäten und zu erstellenden Artefakte in Beziehung gesetzt und zeitlich geplant (s. Abb. 2). Der Prozessmodellierer stellt dabei eine syntaktische Unterstützung zur Verfügung und bietet durch eine übersichtliche Darstellung der einzelnen den UP innewohnenden Hierarchieebenen eine gute Orientierungshilfe. Desweitere erlaubt der Modellierer es, die einzelnen Elemente des zu erstellenden Prozessmodells zu annotieren. Diese Annotationen werden zusammen mit dem Prozessmodell persistent gemacht.

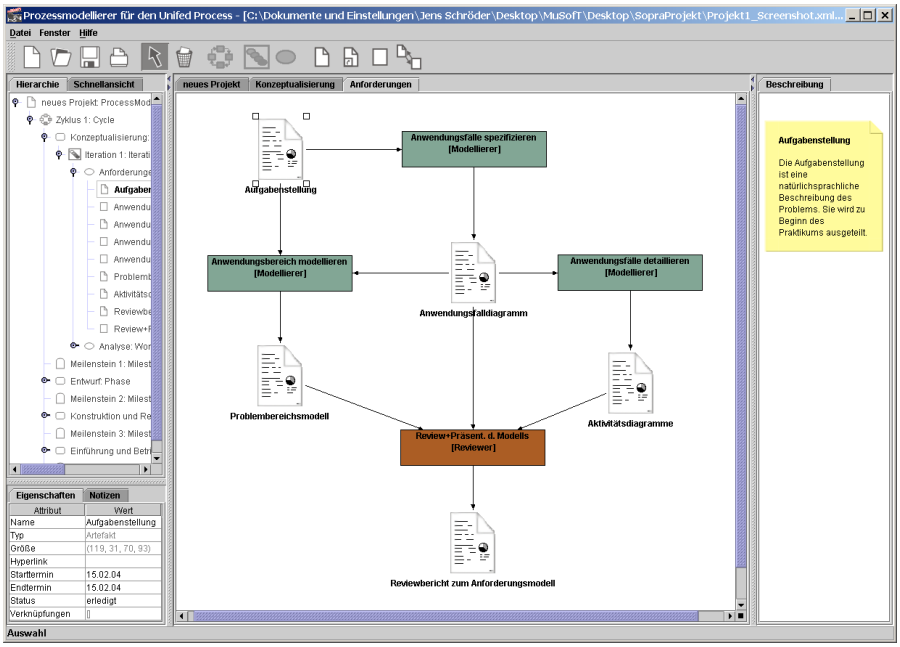


Abbildung 2: Lernmodul *Projekte maßschneidern*: der Prozessmodellierer

Lernmodul *Projekttutor*

Da die Studierenden den modellierten Prozess im Rahmen des Praktikums auch anwenden sollen, wird mit dem Projekttutor ein Werkzeug bereitgestellt, dass sie dabei unterstützt. Der Prozesstutor arbeitet auf der Basis des mit dem Prozessmodellierer instanziierten Prozessmodells und verfolgt den aktuellen Projektfortschritt. Dieses Prozessmodell ist ein von der Arbeitsgruppe geplanter Entwicklungsprozess, kann aber auch ein vorgegebener Standardprozess sein. Dabei legt der Tutor dem Vorgehen bei der Softwareerstellung eine artefaktzentrierte Sichtweise zu Grunde, wie sie im Rahmen von Lehrveranstaltungen üblich ist, da von den Studierenden Artefakte erstellt, abgegeben und von den Betreuern korrigiert werden müssen.

Zur Unterstützung bei der Projektdurchführung bietet der Tutor verschiedene Sichten auf den Prozess an (s. Abb. 3) und stellt dabei unterschiedliche Hilfestellungen zur Verfügung. Neben einer in der Softwaretechnik üblichen Darstellung des zeitlichen Ablaufs des Prozesses in Form eines Gantt-Charts wird ein Aktivitätsbaum angeboten, der eine hierarchische Sicht auf den Prozess liefert, die eine Unterstützung bei der Auswahl der als nächstes auszuführenden Tätigkeiten umfasst. Dabei werden die Abhängigkeiten, die zwischen den auszuführenden Aktivitäten mit ihren zu erstellenden Artefakten existieren, und der aktuelle Projektfortschritt erfasst und visualisiert, und davon abhängig für die einzelnen Prozesselemente Todo-Listen erstellt. Desweiteren steht eine Rollensicht zur Verfügung, bei der den einzelnen Rollen ihre auszuführenden Tätigkeiten zugeordnet sind.

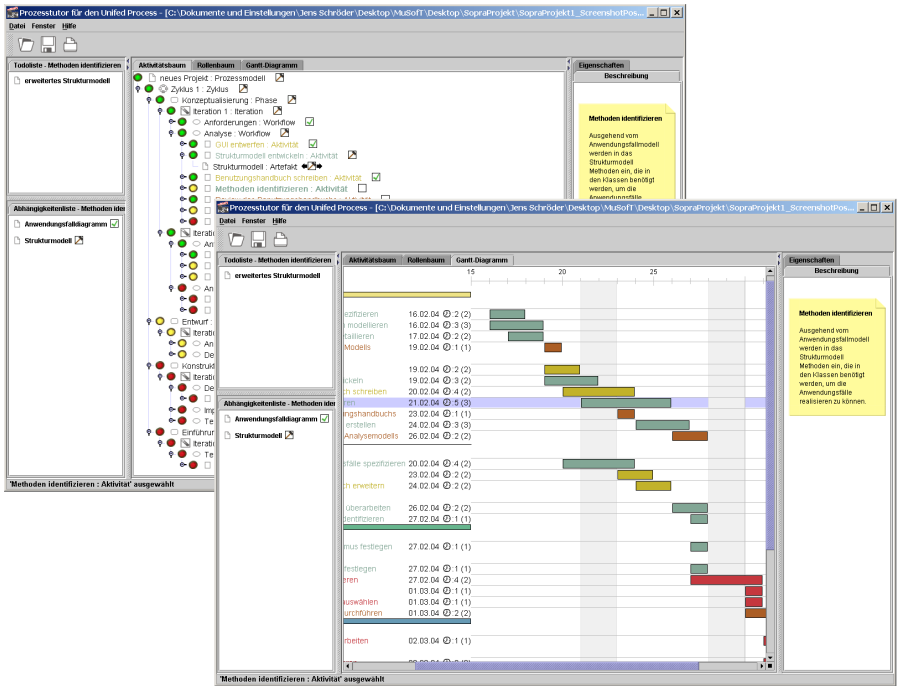


Abbildung 3: Lernmodul *Projekttutor*

Um den Tutor auf dem aktuellsten Stand des Projekts zu halten, wird anhand von Artefakten der Projektfortschritt verfolgt. Dabei werden die Studierenden aktiv mit einbezogen: Sie tragen im Tutor ein, in welchem Bearbeitungsstatus (*unbearbeitet*, *in Bearbeitung* und *erledigt*) ein Artefakt ist. Die davon abhängigen Informationen und Visualisierungen werden immer entsprechend aktualisiert. Da dieses manuelle Eintragen von den Studierenden in der hierarchischen Sicht auf das Prozessmodell vorgenommen wird, haben sie immer den unmittelbaren Rückbezug zum Prozess und können ihre ausgeführte Tätigkeit im Projektablauf richtig einordnen. Ziel ist es dabei, die Studierenden zur Reflexion über ihr Handeln anzuregen und dadurch ihr Wissen über den UP zu vertiefen.

3 Technische Realisierung

3.1 Lernmodul *Unified Process*

Die Entwicklung des Lernmoduls *Unified Process* wurde in den groben Phasen Vorplanung, Konzeptionierung, Realisierung und Test vollzogen. In die Vorplanung gehörten inhaltliche und didaktische Vorüberlegungen, die Festlegung eines Vorgehensmodells und die Projektplanung. Die Konzeptionierungsphase beinhaltete im wesentlichen die Erstellung eines Grob-

konzepts, eines Feinkonzepts und eines Drehbuchs, in denen die spätere Anwendung unter den Aspekten Didaktik, Inhalt, Medieneinsatz/Mediendesign und Navigationsstruktur spezifiziert wird. Eine Beispielseite aus dem Drehbuch wird in Abb. 4 dargestellt. Das Drehbuch bildete die Vorlage für die Realisierungsphase, in der die softwaretechnische Entwicklung der Lerneinheit und die Produktion von Medienobjekten stattgefunden hat. Beim Testen wurde zwischen technischem Test und Evaluation als inhaltlich-didaktischem Test der Lerneinheit unterschieden. Die inhaltlich-didaktische Evaluation wurde einerseits durch die Entwickler der Lernmoduls als Inhaltsexperten (wissenschaftliche Mitarbeiter des Lehrstuhls für Software-Technologie) und andererseits durch die Studierenden durchgeführt.
Die Entwicklung erfolgte iterativ. Zuerst wurde eine Probesequenz (Szenenfolge) beispielhaft entwickelt und getestet, um möglichst früh eine Vorstellung vom Verhalten und Aussehen des Endprodukts zu bekommen und gegebenenfalls Änderungen vorzunehmen. Die Phasen der Drehbucherstellung, Lernprogrammentwicklung, Medienproduktion und der Tests (technischer Test, Evaluation durch Inhaltsexperten) wurden mehrfach iterativ durchlaufen. Eine Iteration wurde für eine Sequenz von Szenen durchgeführt. Anschließend an die Entwicklung des gesamten Lehrbuchs wurde dafür und für die weiteren Lernmodule eine Evaluation durch die Studierenden in mehreren Stufen durchgeführt. Diese wird in Abschnitt 4.1 ausführlich beschrieben.
Bei der Planung des Medieneinsatzes wurde mit einer Mediendesignerin zusammengearbeitet, um weitere Ideen für passende und aussagekräftige Medien für verschiedene Inhalte zu finden. Desweiteren wurde durch die Teilnahme an dem MuSofT-Gestaltungsworkshop im Januar 2003, in dem in allgemeine Gestaltungsrichtlinien eingeführt wurde, die als Grundlage für die Gestaltung von Lerneinheiten und GUIs dienen können, Anregungen für Verbesserungen des Bildschirmdesigns des Lernmoduls gefunden.
Die Produktion von Medien, wie bspw. Grafiken und einfache Animationen erfolgte in Eigenregie durch das Entwicklungsteam des Teilprojekts am Lehrstuhl für Software-Technologie der Universität Dortmund. Es sind aber auch Medien entwickelt worden, die schwierig, aufwändig oder nur von Medienexperten zu erstellen sind. Dies sind beispielsweise einige Animationen mit Sprechertexten und ein Video, das die Aufgabenstellung für ein Softwareprojekt des Softwarepraktikums als Dialog zwischen einem Kunden und einer Softwareentwicklerin darstellt. Für die Erstellung dieser Medien wurde mit dem Medienzentrum an der Universität Dortmund zusammengearbeitet, die auf dem Gebiet der Medienproduktion kompetent sind.
Als Beispielprojekt für die Veranschaulichung des Unified Process wurde eine Aufgabenstellung aus dem Logistikbereich gewählt. Konkret wird der Ablauf eines Projekts dargestellt, in dessen Rahmen Software für die Kommissionierung im Pharmagroßhandel entwickelt wird.
Das Lernmodul ist als HTML-Anwendung unter Einbindung von Flash-Animationen realisiert worden. Dafür wurden die Inhalte in XHTML spezifiziert und mit Hilfe eines Konverters unter Verwendung eines einheitlichen Stylesheets und eines generischen in Javascript realisierten Navigationsrahmens die HTML-Seiten der Anwendung generiert. In Abb. 1 ist beispielhaft eine Seite des Lernmoduls dargestellt.

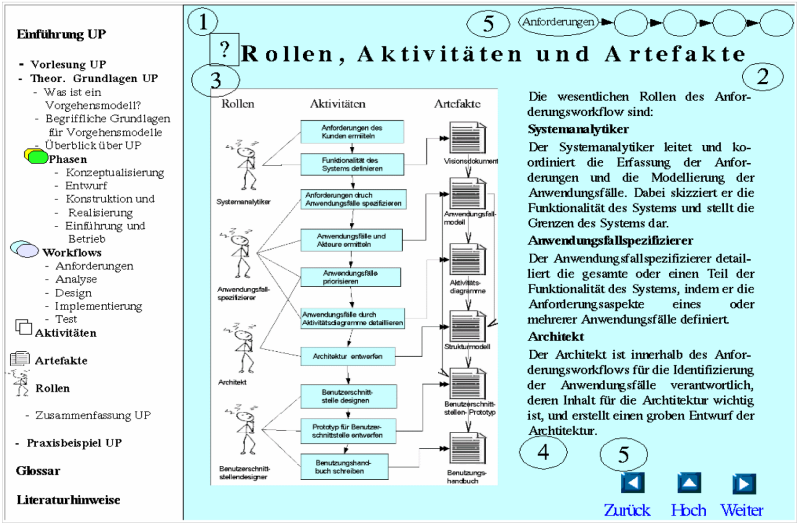
3.2 Lernmodule *Projekte maßschneidern* und *Projekttutor*

Da diese beiden Lernmodule funktionale Werkzeuge sind und nicht wie das Lernmodul *Unified Process* Wissen multimedial präsentieren, erfolgt ihre Entwicklung in Anlehnung an das

20.Szene: Anforderungsworkflow - Rollen, Aktivitten und Artefakte

Beschreibung:

In dieser Szene werden die am Anforderungsworkflow beteiligten Rollen mit den von ihnen auszuführenden Aktivitäten und den zu erstellenden Artefakten dargestellt.



Nr.	Typ	Beschreibung	Name
	Hintergrund	*Farbe*: Blassgrün	
1	Grafik	Wiedererkennungsmerkmal für die Rubrik „Rollen, Aktivitäten und Artefakte“ *Symbol : ...*	
2	Text	Überschrift der Szene. *Inhalt:* Rollen, Aktivitäten und Artefakte.	
3	Grafik	Die Grafik illustriert den Zusammenhang zwischen den Rollen, Aktivitäten und Artefakten.	RollenAktivitätenArtefakte.gif
4	Text	Der Text führt alle am Workflow beteiligten Rollen auf und erläutert die von ihnen auszuführenden Aktivitäten sowie die zu erstellenden Artefakte. *Inhalt:* siehe 4.2.5 Komponente: Rollen, Aktivitäten und Artefakte	4.2.5 Komponente: Überblick

Abbildung 4: Eine Beispielseite aus dem Drehbuch

Vorgehen in der traditionellen Softwareentwicklung, insbesondere sind hier keine Drehbücher als Vorlage für die Implementierung notwendig. Der Entwurf der Werkzeuge geschieht aufgrund bekannter Methoden der Softwaretechnik.

Kern des Lernmoduls *Projekte maßschneidern* ist ein Editor zur Modellierung des eigenen

Entwicklungsprozesses auf Basis des Unified Process. Mit dem Werkzeug können Modellierungselemente wie Phasen, Iterationen, Workflows, Aktivitäten, Artefakte und Abhängigkeiten angelegt werden (Abb. 2).
Vorhandene Prozessmodellierungswerkzeuge kamen für den Einsatz in diesem Lernmodul nicht in Frage, da sie keine gute Unterstützung in der Handhabung des Unified Process bieten können, insbesondere die Charakteristika des Unified Process wie Phasen/Workflows/Iterationen nicht unterstützen. Zudem haben kommerzielle Werkzeuge für den Unified Process den Nachteil, das sie für die Unterstützung professioneller Entwickler konzipiert sind, die viel zu umfangreich ist, und daher eine spezifische Aufbereitung für die Lehre erforderlich ist. Es wurde ein Werkzeug entwickelt, das nicht nur die passenden Modellierungselemente anbietet, sondern auch die Eigenheiten des Unified Process berücksichtigt und hilft, ihn richtig anzuwenden. Beispielsweise besteht ein Entwicklungsprozess auf Basis des Unified Process immer aus den vier Phasen *Konzeptualisierung (inception)*, *Entwurf (elaboration)*, *Konstruktion und Realisierung (construction)*, *Einführung und Betrieb (transition)*. Daher ist zur Unterstützung des Lernenden beim Anlegen eines neuen Projekts das Anbieten dieser Phasen in der richtigen Reihenfolge sinnvoll, ohne dass er diese explizit anlegen muss. Ebenso sind weitere Möglichkeiten gegeben (bspw. Workflows in richtiger Reihenfolge), um den Lernenden mehr Unterstützung zu geben, den Unified Process richtig anzuwenden.
Kern des Lernmoduls *Projekttutor* ist ein Editor zur Unterstützung der Studierenden bei der Durchführung eines Softwareprojekts im Softwarepraktikum auf Basis des eigenen modellierten Entwicklungsprozesses. Anhand eines Ampelsystems kann man erkennen, welche Aktivitäten aufgrund der Abhängigkeit von anderen zu erledigenden Aktivitäten aktuell durchgeführt werden können (Abb. 3).
Die Realisierung des Editors zur Prozessmodellierung und des Projekttutors basiert auf einem Java-Framework für so genannte leichtgewichtige Modellierungswerkzeuge [Ple03, APS04], das im Rahmen des MuSofT-Projekts zusammen mit der Lerneinheit 2.1 am Lehrstuhl für Software-Tehnologie an der Universität Dortmund entwickelt wurde. Es stellt einen generischen Applikationsrahmen mit einer übersichtlichen, für die Bedürfnisse von Lehrveranstaltungen zugeschnittenen Benutzungsschnittstelle und weitere didaktische Funktionalität bereit, auf dessen Grundlage spezielle Modellierungswerkzeuge mit relativ geringem Aufwand implementiert werden können. Es war so möglich, eine einheitliche Familie von eigens auf die Bedürfnisse der Lehre zugesgeschnittenen Modellierungswerkzeugen zu entwickeln, was bei den Studierenden den Einarbeitungsaufwand minimiert und die Akzeptanz der Werkzeuge erhöht hat.

4 Erfahrungen und Evaluierung

Um einerseits die inhaltliche und didaktische Zielsetzung eines Softwarepraktikums und die Möglichkeiten des Medieneinsatzes in einer unterstützenden multimedialen Lerneinheit zu erforschen und um andererseits die Lerneinheit 3.3 nach ihrer Fertigstellung in Softwareprojekten zu evaluieren, wurde im Rahmen dieses Vorhabens mit den Veranstaltern des Softwarepraktikums des Fachbereichs Informatik der Universität Dortmund zusammengearbeitet, die bisher ein grobes Prozessmodell [Sch01] einsetzen, das an das ISP-Modell (*Irrational Separated Model*) von Hitz und Kappel [HK99] angelehnt ist. Der Prozess selbst ist dabei aber nicht

das Lernobjekt. Durch den Einsatz der Lerneinheit 3.3 des MuSofT-Projekts soll in Softwarepraktika auch der Entwicklungsprozess Lerngegenstand werden und beispielhaft anhand des Unified Process gelehrt werden können. Es wurde die Erfahrung gemacht, dass es schwierig ist, in einem Praktikum neben der Lehre der Entwicklung einer Software auch den Entwicklungsprozess zu lehren, weil die Zeit für die reine Entwicklung gekürzt werden muss.
Ebenso muss der Unified Process so skaliert werden, dass er in relativ kleinen Projekten im Softwarepraktikum durchführbar ist. Hierzu wurde im Lehrbuch darauf geachtet, dass im Praxisbeispiel ein Prozessmodell vorgegeben wurde, das einem kleineren Projekt im Softwarepraktikum entspricht, und dass mit Hinweisen auf Erweiterungs-/Änderungsmöglichkeiten für spezifische Anwendungsprojekte Hilfestellung geboten wurde.

4.1 Evaluierung

Der Unified Process wurde an der Universität Dortmund im Grundstudium im Rahmen der vorhandenen Veranstaltungsformen zur Lehre der Softwaretechnik (Vorlesung mit Übung, Softwarepraktikum) eingeführt. Dies war in dieser frühen Phase des Studiums mit Risiken verbunden, da der UP in der Regel nicht zum Lehrstoff des Grundstudiums gehört. In diesem Zusammenhang ist ein ständiger Verbesserungsprozess für den Einsatz der Lerneinheit wichtig, der auf einer wissenschaftlich fundierten Evaluation zur Überprüfung unseres Konzepts basiert. Die Evaluation wurde daher in Zusammenarbeit mit dem Hochschuldidaktischen Zentrum an der Universität Dortmund durchgeführt [KMGSD04]. Für die Evaluation der Lerneinheit wurde ein dreistufiges Vorgehen gewählt [KSS04]:

- Vorevaluation (Pretest)
- Evaluation in der Vorlesung Softwaretechnik im Grundstudium
- Evaluation im Softwarepraktikum im Grundstudium

Ziel des Pretests war, die Lernmodule schon im Vorfeld einer Lehrveranstaltung zu evaluieren, um gravierende Fehler insbesondere bei den funktionalen Werkzeugen (Prozessmodellierer und Tutor) auszuschließen und genügend Zeit für aufwändige Korrekturen zu haben. Dieser Pretest wurde mit Studierenden in 3 Gruppen mit 2-3 Personen an zwei Evaluationstagen durchgeführt. Die Evaluation wurde anhand einer konkreten Projektaufgabenstellung zur Modellierung eines Entwicklungsprozesses vorgenommen. Eine Projektdurchführung wurde an den Rechnern simuliert, indem den Gruppenteilnehmern Rollen zugeordnet wurden und jeder Teilnehmer die von ihm zu erstellenden Artefakte mit Hilfe des Tutors als erledigt gekennzeichnet hatte, so dass sie mittels des Tutors für die anderen Gruppenteilnehmern sichtbar wurden. Auch wenn die Durchführung eines Projekts nur simuliert wurde, sind durch die Erkenntnisse bei der Nutzung der Werkzeuge einige Verbesserungsvorschläge entstanden.
Die Evaluation beim Pretest erfolgte durch Beobachtung der Studierenden bei der Arbeit und einer anschließenden Befragung der Gruppen. Die Studierenden konnten während der Nutzung der Lernmodule jederzeit Fragen stellen. Das Feedback der Studierenden war überwiegend positiv.
Die wichtigsten Ergebnisse bezüglich des didaktischen Konzepts haben wir aus der zweiten und dritten Stufe der Evaluation erwartet, da hier die Lernmodule in *echten* Lehrveranstaltungen eingesetzt wurden.

Die zweite Stufe der Evaluation wurde durch den Einsatz des Lehrbuchs und des Prozessmodellierers im WS 2003/2004 in der Übung zur Vorlesung Softwaretechnik im Grundstudium durchgeführt. Hier wurde der Unified Process in der Vorlesung eingeführt und als Übung die Modellierung eines Prozesses anhand einer konkreten Projektaufgabenstellung angefordert. Die Evaluation erfolgte durch Verteilung und Auswertung eines umfassenden Fragebogens. Es lagen 287 ausgefüllte Fragebögen vor.
Für die dritte Stufe der Evaluation wurde im Softwarepraktikum das Prozessmodell für ein Projekt vorgegeben, das mit Unterstützung des Tutors durchgeführt wurde. Ein zweites Projekt wurde von den Studierenden als UP-Prozessmodell mit Unterstützung des Lehrbuchs und des Prozessmodellierers geplant und mit Unterstützung des Tutors durchgeführt. Die Evaluation erfolgte durch Interviews mit ausgewählten Studierenden. Am Software-Praktikum nahmen 72 Studierende teil, die in neun Projektteams aufgeteilt waren.

4.2 Ergebnisse der Evaluierung

Obwohl beim Pretest die Durchführung eines Projektes nur simuliert wurde, konnten durch die Erkenntnisse bei der Nutzung der Werkzeuge einige funktionelle Verbesserungsvorschläge identifiziert werden. Generell wurden die Werkzeuge von den Studierenden überwiegend positiv eingeschätzt.
Die im Anschluss an die Softwaretechnikvorlesung erhobenen Evaluationsergebnisse bestätigen, dass sich die Konzepte des UP in Form einer Vorlesung nur schwer vermitteln lassen. Während immerhin 39 Prozent der Studierenden angeben, den Nutzen des UP verstanden zu haben, ist 45 Prozent die Anwendung des UP nicht klar geworden. Studierende im Grundstudium, die niemals zuvor ein Softwareprojekt durchgeführt haben, sind anscheinend überfordert, die Inhalte zu verstehen. Insbesondere der generische Charakter des UP macht es den Studierenden schwer, eine Anschauung zu entwickeln.
Auch die zugehörige Übung, in der die Studierenden einen Prozess modellieren sollten, wird nicht gut bewertet. Viele Studierende hatten die Aufgabenstellung nicht richtig verstanden, was sich darin ausdrückt, dass 44 Prozent der Befragten finden, dass Vorlesung und Übung zum UP nicht gut aufeinander abgestimmt sind. Weniger die Erklärungen in der Übungsgruppe, als vielmehr das eigenständige Arbeiten mit dem UP haben den Studierenden die Funktionsweise des UP klar gemacht.
Die Studierenden haben sich neben Vorlesung und Übung selbstständig in die Inhalte eingearbeitet. 66 Prozent haben das Praxisbeispiel im Lehrbuch häufig benutzt, immerhin 22 Prozent haben es einmal benutzt. Außerdem haben noch 18,5 Prozent ein (Papier-) Lehrbuch gelesen. Offensichtlich wirkt sich die Zeit, die eingesetzt wurde, um das UP-Lehrbuch zu erarbeiten, auf die Fähigkeit aus, die Übungsaufgabe eigenständig zu modellieren. 38 Prozent hat das Praxisbeispiel geholfen, die Übungsaufgabe zu modellieren, 36 Prozent stimmten dem nicht zu. 56 Prozent derer, die 2 Stunden eingesetzt haben, um das UP-Lehrbuch zu lesen, fanden das Praxisbeispiel hilfreich, 33 Prozent fanden es nicht hilfreich. Je weniger Zeit eingesetzt wurde, desto mehr verkehrte sich dieses Verhältnis um. Von denen, die weniger als eine halbe Stunde eingesetzt haben, fanden nur 30 Prozent das Praxisbeispiel als hilfreich und 37 Prozent fanden es nicht hilfreich.
Auf die Frage, wie hoch sie den Nutzen des Prozessmodellierers für die Lösung der Übungs-

aufgabe einschätzen, antworteten 11 Prozent sehr gut, 33 Prozent gut und 27 Prozent zeigten sich zufrieden mit dem Nutzen des Werkzeugs.

Im darauffolgenden Softwarepraktikum kamen wieder das multimediale UP-Lehrbuch und die beiden UP-Werkzeuge Prozessmodellierer und Projekttutor zum Einsatz. Mit Hilfe des Modellierers konnten die PraktikumsteilnehmerInnen sich Einblicke in den Ablauf des geplanten ersten Projekts verschaffen. Für jeden Workflow war detailliert die Abhängigkeiten zwischen den einzelnen Tätigkeiten und Artefakten beschrieben. Der Projekttutor bot mit seinem Gantt-Chart den PraktikumsteilnehmerInnen einen guten Überblick über die Zeitplanung des Projekts und durch das Abhaken der erstellten Artefakte im Aktivitätsbaum wurde der Projektfortschritt gut verdeutlicht. Das trug zur Motivation der TeilnehmerInnen bei.

Das zweite Projekt wurde von den TeilnehmerInnen selbstständig mit Hilfe des Modellierers geplant. Wie in der Aufgabenstellung vorgegeben wurden zwei Entwicklungszyklen (zunächst eine Mensch-gegen-Mensch-Spiel, danach das Spiel mit simuliertem Computergegner) definiert. Innerhalb eines Zykluses orientierten sich die Studierenden weitgehend an den Vorgaben des ersten Projekts. 68 Prozent der TeilnehmerInnen fanden es sehr hilfreich für das zweite Projekt, dass beim ersten Projekt die Prozessmodellierung vorgegeben war, weitere 24 Prozent fanden es zumindest teilweise hilfreich. 89 Prozent der TeilnehmerInnen fühlten sich im zweiten Projekt nicht mit der eigenständigen Planung überfordert. Unterschiede zum ersten Projekt ergaben sich z.B. in der Integration von Testmethoden in die Implementierung, dem Zeitpunkt der Erstellung des Benutzungshandbuchs und insbesondere in der Länge der vorgesehenen Zeiten für die Aktivitäten. Nur eine der neun Gruppen lehnte es grundsätzlich ab, ihr Projekt gemäß UP zu planen. Die Gruppe wollte angelehnt an eXtreme Programming einen iterativen Prozess planen, sich aber nicht auf eine bestimmte Anzahl von Iterationen festlegen. Ein derartiges Vorgehen ist wenig sinnvoll und ließ sich auch nicht mit dem Modellierer planen. Stattdessen wurde ein UML-Aktivitätsdiagramm zur Veranschaulichung der Planung eingesetzt.

Insgesamt konnten wir beobachten, dass die Studierenden in der Lage waren, einen Entwicklungsprozess zu planen, ihre Planung zu präsentieren, ein Projekt gemäß der Planung durchzuführen und ihre Erfahrungen zu reflektieren. Dies deckt sich mit der eigenen Einschätzung der Studierenden über die im Softwarepraktikum erworbenen Kenntnisse. 68 Prozent der TeilnehmerInnen schätzten ihre Kenntnisse, Softwareprozesse zu modellieren, als gut ein, weitere 6 Prozent als sehr gut. 61 Prozent sind der Meinung sich gute Kenntnisse in der Entwicklung konkreter Projekte angeeignet zu haben, 14 Prozent sogar sehr gute Kenntnisse. Ihre Fähigkeit, ein Softwareprojekt zu organisieren schätzten 56 Prozent als gut und 11 Prozent als sehr gut ein.

5 Zusammenfassung

Die Lerneinheit 3.3 ist ein Lehr-und Lernsystem, das über die Präsentation reinen Faktenwissens hinaus geht. Die Einbettung der Lerneinheit in ein didaktisches Konzept führt zu Anforderungen wie die didaktischen Aufbereitung von Lehrmaterial, der Einsatz mediendidaktischer Elemente und von Softwarewerkzeugen und die Betreuung der Studierenden im Lernprozess:

- Dozenten sollen mit Hilfe der Lerneinheit den Unified Process lehren können, Studierende sollen im Selbststudium den Unified Process verstehen. Dazu dient im wesentlichen das multimediale Lehrbuch.
- Studierende sollen unter Betreuung in einem konkreten Projekt den Entwicklungsprozess auf Basis des Unified Process planen lernen. Lernen bedeutet an dieser Stelle Anwenden des Faktenwissens über den Unified Process, insbesondere auch das Planen von Iterationen in einem konkreten Projekt. Hierzu setzen sie den Prozessmodellierer ein.
- Studierende sollen den Unified Process explizit erfahren. Lernen bedeutet an dieser Stelle das Durchführen eines vorgegebenen oder des selbst geplanten Entwicklungsprozesses innerhalb des Softwarepraktikums. Hierzu setzen sie den Projekttutor ein. Damit erhalten sie die Gelegenheit, über geplante Prozesse zu reflektieren. Dies ist ebenfalls ein Lernprozess mit dem Ziel, Projekte besser planen zu können.

Die gestellten Anforderungen an die Lerneinheit wurden erfüllt und die Art des Einsatzes der Lerneinheit in Lehrveranstaltungen hat sich bewährt. Die Beobachtungen und Befragungsergebnisse bestätigen den mehrstufigen Einsatz in Vorlesung, Übung und Praktikum. Prozessmodellierung als Inhalt in der Softwaretechnikausbildung macht nur Sinn, wenn Prozessmodellierung mit Softwareentwicklungsprojekten verknüpft wird. Auch eine Übung im herkömmlichen Sinne kann nur sehr begrenzt Erfahrungswissen vermitteln, da die Planung eines Projekts ohne seine tatsächliche Durchführung die Schwachstellen der Planung nicht deutlich werden lässt. Erst die eigene Projektdurchführung im Praktikum ermöglicht den zielgerichteten Einsatz des Wissens über Projektmodellierung. Dabei boten die eingesetzten Werkzeuge durch die explizite Darstellung des Entwicklungsprozesses eine gute Unterstützung für die Lernenden und Lehrenden. Anhand der übersichtlichen Darstellung im Modellierer und Tutor konnte gut über den Prozessablauf und den Fortschritt kommuniziert werden. Die Werkzeuge unterstützten die Studierenden aufgrund ihrer angebotenen Funktionalität bei der Anwendung des vermittelten Wissens.
Diese positiven Erfahrungen haben uns motiviert, den UP mit Hilfe unserer Lerneinheit auch im nächsten Veranstaltungskanon (Vorlesung/Übung/Praktikum) vorzusehen. Auch ein Einsatz in Veranstaltungen des Hauptstudiums (z.B. Projektgruppen) ist angedacht.

Literatur

[APS04] ALFERT, KLAUS, JÖRG PLEUMANN und JENS SCHRÖDER: *Software Engineering Education Needs Adequate Modeling Tools*. In: HORTON, THOMAS B. und ANN E.K. SOEBEL (Herausgeber): *17th Conference on Software Engineering Education & Training (CSEE&T 04)*, Seiten 72–77. IEEE Computer Society, März 2004.

[HK99] HITZ, MARTIN und GERTI KAPPEL: *UML@Work - Von der Analyse zur Realisierung*. dpunkt-Verlag, 1999.

[Hum96] HUMPHREY, WATTS S.: *Using a Defined and Measured Personal Software Process*. IEEE Software, Seiten 77–88, Mai 1996.

[JBR98] JACOBSON, IVAR, GRADY BOOCH und JAMES RUMBAUGH: *The Unified Software Development Process*. Addison Wesley, 1998.

[KA02] KOPKA, CORINA und KLAUS ALFERT: *Der Softwareentwicklungsprozess als Lehrgegenstand oder Von einem, der auszieht, das reflektierte Handeln zu lehren*. In: SCHUBERT, SIGRID, BERND REUSCH und NORBERT JESSE (Herausgeber): *Informatik bewegt. 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 30. Sept.–3. Okt. 2002 in Dortmund*, Nummer P-19 in *Lecture Notes in Informatics*, Seiten 401–407, Bonn, 2002. Gesellschaft für Informatik.

[KMGSD04] KAMPHANS, MARION, SIGRID METZ-GÖCKEL, AIRA SCHÖTTELNDREIER und ANNA DRAG: *Der Unified Process im Test. Evaluationsergebnisse zum Einsatz des UP in der Informatik-Lehre*. MuSofT-Bericht Nr. 7, Hochschuldidaktisches Zentrum, Universität Dortmund, 2004. im Druck.

[Kru99] KRUCHTEN, PHILIPPE: *The Rational Unified Process: an Introduction*. Addison-Wesley, 1999.

[KSS04] KOPKA, CORINA, DORIS SCHMEDDING und JENS SCHRÖDER: *Der Unified Process im Grundstudium - Didaktische Konzeption, Einsatz von Lernmodulen und Erfahrungen*. In: *DeLFI 2004 - 2. e-Learning Fachtagung der Gesellschaft für Informatik, 5. - 8. September 2004 in Paderborn*, September 2004.

[Ple03] PLEUMANN, JÖRG: *A Framework for Lightweight Graphical Modeling Applications*. In: *Proceeedings of the 7th Conference on Systemics, Cybernetics and Informatics (SCI)*, 2003.

[Sch01] SCHMEDDING, DORIS: *Ein Prozessmodell für das Software-Praktikum*. In: LICHTER, HORST und MARTIN GLINZ (Herausgeber): *SEUH 7. Software Engineering im Unterricht der Schulen*, Seiten 87–97, Zürich, Februar 2001. dpunkt-Verlag.

Bericht über das Teilprojekt 3.4 “Projektmanagement” im Projekt MuSofT

Udo Kelter

Dieser Bericht faßt die wesentlichen Ergebnisse des Teilprojekts 3.4 “Projektmanagement” im Projekt MuSofT zusammen. Zunächst wird analysiert, für welche Lehrinhalte im Themengebiet Projektmanagement der Einsatz neuer Medien und die Entwicklung entsprechender Lehrmaterialien sinnvoll ist. Die entstandenen Lehrmaterialien behandeln die Themen Konfigurationsmanagement mit CVS sowie Projektplanung mit Netzplänen. Struktur, Einsatz und Evaluierung der Materialien werden beschrieben. Ferner werden Erfahrungen bei der (Weiter-) Entwicklung eines Lehrfilms über ein CVS-Front-End beschrieben.

Inhaltsverzeichnis

1 Einleitung

Gegenstand des gesamten Projekts war die softwaretechnische Ausbildung unter Einsatz neuer Medien und insb. die Entwicklung und Erprobung von multimedialen Lehrmaterialien aus diesem Themenkomplex, und zwar vor allem solcher Materialien, die in größeren Veranstaltungen einsetzbar sind. Wegen der großen thematischen Bandbreite des Gebiets Softwaretechnik konnte dies nur exemplarisch erfolgen; ferner war eine Arbeitsteilung dahingehend erforderlich, daß die einzelnen beteiligten Arbeitsgruppen komplementäre Themenbereiche bearbeiteten. Aus dem Themenspektrum übernahm die Universität Siegen den Themenbereich Projektmanagement.

Das Projekt MuSofT begann am 1.3.2001 und war auf knapp 3 Jahre Laufzeit (Ende: 31.12.2003) angelegt. Die Grobplanung des Projekts sah vor, innerhalb der ersten Hälfte der Laufzeit bei allen Partnern erste Versionen von Lehrmaterialien zu entwickeln und diese in der zweiten Phase aufgrund der Einsatzerfahrungen zu überarbeiten und zu erweitern.

Abweichend von der ursprünglichen Planung wurde in 2003 die Laufzeit des Projekts bei mehreren Projektpartnern, so auch Siegen, kostenneutral verlängert. Ein entsprechender Teil der Mittel in 2003 wurde zurückgegeben und in ein formell neues Projekt eingebracht.

2 Themenauswahl

Das Themengebiet Projektmanagement ist sehr umfangreich; wenn man den Begriff Management weit faßt, sind fast alle Tätigkeiten in den frühen Projektphasen Managementtätigkeiten. Selbst bei einer eingeengten Definition des Themengebiets – die wir i.f. unterstellen – kann wegen des Aufwands nur ein kleiner Teil des Gebiets abgedeckt werden.

Für die Auswahl der Themen und der erstellten Materialien gab es mehrere Arten von Kriterien, die in diesem Abschnitt diskutiert werden. Vorangestellt werden können zwei übergeordnete Kriterien:

- Die Materialien sollen geeignet sein, die Lehre im Themengebiet zu unterstützen und einen Mehrwert gegenüber klassischen Lehrformen und -Materialien (insb. Lehrtexte und Folien) bieten.

- Die Materialien sollen eine realistische Chance haben, auch von anderen Lehrenden akzeptiert und eingesetzt zu werden.

Das erste Kriterium bezieht sich nur auf die eigentlichen Lernprozesse und ist leicht ersichtlich. Das zweite Kriterium adressiert auch technische, organisatorische, curriculare und sonstige Hindernisse, die einen Lehrenden davon abhalten könnten, im Prinzip brauchbare Lehrmaterialien einzusetzen. Beide Kriterien sind nicht scharf zu trennen und müssen konkretisiert werden. Hinzu kommen projektbezogene Kriterien wie die thematische (Nicht-) Überlappung mit anderen MuSofT-Teilprojekten.

2.1 Auswahlkriterien

2.1.1 Einpaßbarkeit in softwaretechnische Lehrveranstaltungen

Für die Einschätzung des Themenbereichs Projektmanagement in der softwaretechnischen Lehre sind folgende Punkte wesentlich:

- Er zerfällt in eine mehrere Teilbereiche, die inhaltlich weitgehend unabhängig voneinander sind und die auch isoliert lehrbar sind[1].
- Im Vergleich zu Themen wie Programmierung, Architekturen oder Systemanalyse hat der Themenbereich Projektmanagement in der grundlegenden Ausbildung in Informatik nur eine geringere Priorität.
- Manche Projektmanagement-Themen können nur vor dem Hintergrund längerer Praxiserfahrungen – die bei Studenten nicht vorliegen – verstanden werden bzw. sie können im universitären Kontext nicht praktiziert oder geübt werden.

Die vorstehend genannten Merkmale des Stoffgebiets haben folgende Konsequenzen für die Projektmanagement-Lehrinhalte in gängigen Lehrbüchern bzw. in Lehrveranstaltungen:

- Der Umfang ist gering, besonders in den Lehrveranstaltungen der frühen Studienphasen.
- Die Themenauswahl schwankt stark. Es gibt praktisch keine Themen, die fast überall z.B. in der Softwaretechnik-Vorlesung gelehrt werden.
- Das gleiche Thema wird manchmal in einer frühen Studienphase behandelt (z.B. in der Grundvorlesung über Programmierung), manchmal erst im Hauptstudium in Spezialvorlesungen.

Hieraus ergaben sich für die Auswahl der Themen, die durch die Materialien des Teilprojekts 3.4 abgedeckt werden sollen, und für die Struktur der Materialien folgende Leitlinien:

- Es soll sich um Themen handeln, die vergleichsweise “häufig” gelehrt werden, häufig in dem Sinne, daß sie in entsprechendem Umfang in Vorlesungen an vielen Standorten gelehrt bzw. in vielen Softwaretechnik-Lehrbüchern vertreten sind.

[1] Beispiele solcher Teilbereiche sind Aufwandsschätzung, Termin- und Ressourcenplanung, Versions- und Konfigurationsmanagement, Qualitätssicherung und Änderungsmanagement. Bei anderen softwaretechnischen Themengebieten, z.B. Modellierungssprachen, sind die Querbezüge und Zusammenhänge zwischen den Teilbereichen viel ausgeprägter, so daß einzelne Teilbereiche nicht ohne weiteres ausgetauscht oder weggelassen werden können.

- Die Materialien sollen in möglichst kleinen Teilmengen isoliert einsetzbar sein. Insbesondere sollten Lehrende, die dem Thema nur sehr wenig Raum in ihren Vorlesungen geben (können), hierzu geeignete Materialien finden, und der Einsatz der Materialien sollte möglichst wenig Einarbeitungs- und Installationsaufwand verursachen.
- Die Materialien sollen in unterschiedlichen Studienphasen einsetzbar sein. Dies bedeutet, daß die Materialien, die weniger Erfahrung erfordern, in möglichst frühen Studienphasen (z.B. im Programmierpraktikum im 3. Semester) einsetzbar sein sollen.

2.1.2 Projektkontext

Aus dem Gesamtprojekt heraus ergibt sich die naheliegende Leitlinie, thematische Überlappungen mit anderen MuSofT-Teilprojekten zu vermeiden. Eine Konsequenz hieraus ist, daß folgende Themen, die im Prinzip in die engere Wahl kämen, im Teilprojekt 3.4 nicht behandelt werden:

- Vorgehensmodelle und Prozeßmodellierung
- ISO-9000-Standards

2.1.3 Fachdidaktische Aspekte

Im Sinne einer Ingenieurausbildung wird dem Erwerb praktischer Lösungskompetenzen eine hohe Priorität eingeräumt, während rein konzeptuelle Kenntnisse, die nicht durch praktische Erfahrungen ergänzt (oder relativiert oder schlimmstenfalls widerlegt) werden, eher kritisch gesehen werden. Praktische Erfahrungen wiederum werden am ehesten in realistischen Anwendungen erworben. Da aus Aufwandsgründen keine umfangreichen “Sandkastenprojekte” durchführbar sind, ist man praktisch gezwungen, ohnehin im Studium durchgeführte größere Entwicklungsprojekte als Basis mitzubenutzen. Derartige Entwicklungsprojekte sind:

- das Programmierpraktikum
- die Projektgruppe
- Studien-, Bachelor- und Diplomarbeiten

Die Lehrerfahrung zeigt auch umgekehrt, daß Themen bzw. Techniken, die konkrete, im universitären Kontext auftretende Probleme behandeln, wesentlich besser motivierbar sind und besser aufgenommen werden also solche, die erst in einer fernen beruflichen Zukunft zum ersten Mal wirklich benötigt werden (oder bis dahin vergessen wurden oder schon technisch überholt sind).

Einige Projektmanagementtechniken sind zwar in der Praxis wichtig und verbreitet und im Prinzip ohne große Probleme vermittelbar, sie finden sich auch in vielen Lehrbüchern, sie sind aber im konkreten universitären Kontext kaum einsetzbar bzw. unter realistischen Randbedingungen übbar. Beispiele hierfür sind einige Methoden zur Aufwandsschätzung (z.B. Funktionspunkte oder COCOMO). Derartige Themen wurden daher in den Materialien von Teilprojekt 3.4 nicht berücksichtigt.

2.1.4 Multimedialer Mehrwert

Die Entwicklung multimedialer Lehrmaterialien ist sehr aufwendig. Schon aus Kostengründen ist es nicht denkbar, in größeren Lehrgebieten die klassischen Lehrformen und -Materialien ausschließlich durch multimediale Lehrmaterialien zu ersetzen. Es stellt sich also die Frage, auf welche Themen und Teilgebiete sich der Einsatz multimedialer Lehrmaterialien (bzw. neuer Medien) konzentrieren sollte.

Diese Frage ist nicht zu trennen von der Zusatzfrage, ob neue Medien in der Lehre überhaupt sinnvoll sind, wenn die klassischen Lehrziele und Prüfungsformen (z.B. Klausuren) und grundlegende Vermittlungsformen (z.B. Frontalunterricht ergänzt durch Übungsaufgaben) beibehalten werden, oder ob nicht zugleich mit der Einführung neuer Medien die grundlegenden Lehrziele an die Möglichkeiten und Grenzen der neuen Medien angepaßt werden müssen. So kann in multimedialen explorativen Lernumgebungen durch lernergesteuertes Studium von Anwendungsbeispielen eine andere Art von “Erfahrung” gesammelt werden als durch klassische Übungsaufgaben und Kleinprojekte.

Sollen eher handwerkliche Fähigkeiten erworben werden, z.B. zur Bedienung von Entwicklungswerkzeugen, muß ein präzises Verständnis der Schnittstellen der Systeme und der dahinterstehenden Konzepte erworben werden. Für solche Lernziele erscheinen die klassischen Vermittlungsformen besser geeignet; sie werden daher auch bei allen Materialien des Teilprojekts 3.4 unterstellt.

Die grundlegende Entscheidung zugunsten klassischer Lernziele und Vermittlungsformen hat als Konsequenz, daß multimediale Lehrmaterialien die klassischen Lehrformen und -Materialien nicht generell ersetzen können, sondern nur punktuell ergänzen, und zwar vor allem bei solchen Themen, wo klassische Materialien (Lehrtexte, Folien) systembedingte Schwächen haben, konkret bei:

- der Visualisierung von Algorithmen und dynamischen Vorgängen
- der Bedienung konkreter interaktiver Werkzeuge

Inwieweit einzelne Lehrstoffe aus dem Bereich Projektmanagement in diesem Sinne von multimedialen Lehrmaterialien profitieren und in der Praxis einsetzbar sein würden, war zu Beginn des Projekts nicht mit Sicherheit prognostizierbar; diese Frage kann als besonderer Forschungsaspekt des Projekts angesehen werden.

2.2 Übersicht über die abgedeckten Themen

Auf Basis der vorstehenden Auswahlkriterien wurden letztlich in folgenden Bereichen Lehrmaterialien erstellt und erprobt:

- Versions- und Konfigurationsmanagement (VKM)
- Projektplanung

Im Bereich VKM wurde in erster Linie ein Lehrfilm realisiert, der in die Bedienung eines interaktiven Front-Ends zu CVS einführt und der vorhandene und frei verfügbare klassische Materialien (Lehrtexte und Folien) ergänzt. Dieser Film erwies sich als sehr erfolgreich, infolgedessen konzentrierten sich im Verlauf des Projekts die Aufwände auf die Weiterentwicklung dieses Films. Details über den Film sind in Abschnitt 3.3 dargestellt.

Ebenfalls zum Bereich VKM gehörig sind einige Animationen, die auch fortgeschrittene Versions- und Konfigurationskonzepte erklären.

Im Bereich Projektplanung wurde ein Werkzeug erstellt, durch das Netzpläne erstellt und der Ablauf der Auswertung simuliert werden kann.

Ferner wurde der Bereich Fehler- und Problemmanagement daraufhin analysiert, wie dort multimediale Materialien sinnvoll einsetzbar sind. Es ergab sich jedoch, daß das Thema zu komplex für einführende, breiter einsetzbare Materialien ist und daß die Erstellung umfangreicherer Materialien nicht sinnvoll ist, weil das Thema nur selten in entsprechender Tiefe in Lehrveranstaltungen berücksichtigt wird und die Entwicklung der Materialien zu aufwendig gewesen wäre.

3 Versions- und Konfigurationsmanagement

3.1 Stoffauswahl

Einfache Konfigurationsmanagementprobleme treten bereits sehr früh im Studium auf, insb. im Programmierpraktikum, später bei praktischen Arbeiten im Rahmen von Projektgruppen, Studienarbeiten und Diplomarbeiten. Das Themengebiet VKM erfüllt alle o.g. Kriterien für die Stoffauswahl.

Beim Versions- und Konfigurationsmanagement handelt es sich um einen sehr weitgespannten Themenkomplex:

- Mit VKM assoziiert werden einerseits sehr anwendungsnahe, in der täglichen Praxis auftretende Tätigkeiten, andererseits komplexe, teilweise spezielle konzeptionelle Fragestellungen. Es sind daher sowohl Begriffe und abstrakte Konzepte zu vermitteln als auch die Bedienung konkreter Werkzeuge. Beide Aspekte sind nicht strikt trennbar, da die Werkzeuge zu vielen Detailproblemen eigene Begriffe prägen.
- Art und sinnvoller Umfang der VKM-Maßnahmen hängen stark von der Struktur und der Größenordnung des versionierten Systems ab. VKM-Konzepte für komplexe Systeme können nur vermittelt werden, wenn auch die Struktur dieser Systeme bekannt ist.
- VKM ist nicht ganz zu trennen von Annahmen über das unterliegende Betriebssystem, insb. dessen Dateisystem und teilweise dessen Kommandointerpreter.
- Bei einem verteilten Zugriff auf ein Repository ist man in diverse Fragen und Probleme der Kommunikation zwischen Rechnern involviert.

Die in den Materialien abgedeckten Inhalte müssen sich notwendigerweise auf die am meisten in der Praxis und im Studium relevanten Aspekte beschränken. Diese sind je nach Studienphase unterschiedlich.

Im *Grundstudium* ist vor allem das Programmierpraktikum betroffen. Hier treten die folgenden praktischen Probleme auf, bei denen der Einsatz von VKM motiviert ist und praktisch geübt werden kann:

- verteilte Bearbeitung von Dokumenten (von Rechnern in universitären Pools und von heimischen PCs aus)

- zeitliche Revisionen von Dokumenten (Konfigurationen treten nur in sehr beschränktem Ausmaß auf)

Neben den Grundbegriffen ist hier vor allem der Umgang mit entsprechenden Werkzeugen bzw. VKM-Systemen zu erlernen. Aus diversen Gründen wurde als Basis für die in Teilprojekt 3.4 erstellten Materialien das System CVS ausgewählt. Hauptergebnis in diesem Bereich ist ein Lehrfilm, der in die Bedienung eines CVS-Front-Ends einführt. Der Film und die Begleitmaterialien werden im Rest dieses Abschnitt ausführlich dargestellt.

Im *Hauptstudium* ist in erster Linie in einer Projektgruppe und bei der Bachelor- bzw. Diplomarbeit mit signifikanten VKM-Problemen zu rechnen.

Bzgl. der verteilten Gruppenarbeit ergeben sich bei Projektgruppen nicht generell neue Anforderungen. Im Prinzip wäre es wünschenswert, die Teilnehmer an einer Projektgruppe in die Lage zu versetzen, selbständig ein entfernt zugreifbares Repository (lokal zugreifbare sind schon durch die einführenden Materialien abgedeckt) aufzubauen. Die Entwicklung von multimedialem Lehrmaterial hierzu erwies sich jedoch als weniger sinnvoll: es müssen teilweise gute Kenntnisse über Betriebssysteme und Rechnernetze vorausgesetzt werden, das Aufsetzen und Administrieren eines Repositorys wird in einer Arbeitsgruppe daher typischerweise einem einzelnen Spezialisten übertragen, ein “breiter” Einsatz der Materialien ist somit wenig unwahrscheinlich und der hohe Aufwand zu seiner Realisierung kaum zu rechtfertigen.

3.2 Lernziele und Zusammensetzung der Materialien

Übergeordnetes Lernziel der Materialien zum Thema VKM ist die Fähigkeit, das VKM-System CVS im Kontext von Aufgabenstellungen, wie sie im Programmierpraktikum anfallen, sinnvoll praktisch einsetzen zu können. Dies bedingt und beinhaltet

- das Erlernen der Grundbegriffe des VKM und deren spezielle Ausgestaltung in CVS (z.B. das Numerierungsschema)
- das Erlernen grundlegender Produkteigenschaften von CVS (z.B. das Verteilungskonzept)
- das Erlernen von Bedienelementen eines graphischen Front-Ends zu CVS
- das Sammeln von Erfahrungen, wie ein Repository gestaltet und in der Gruppenarbeit eingesetzt werden sollte

Zur Erreichung dieser Lernziele werden unterschiedliche Materialien eingesetzt:

- klassische Lehrbuchtexte
- Übungsaufgaben
- Foliensätze
- ein Lehrfilm (Detail s. Abschnitt 3.3)
- eine live-Demonstration (Linux-Shell-Skript)

Die Grundbegriffe des VKM (elementare Begriffe wie Revision, Variante, Arbeitsbereich usw.) und die grundlegenden Produkteigenschaften von CVS werden am besten durch klassischen Vortrag und/oder Lektüre von Lehrbuchtexten vermittelt. Entsprechende Lehrbuchtexte

mit begleitenden Übungsaufgaben und Foliensätzen zu diesen und vielen weiteren Themen wurden schon vor dem Projekt erstellt und standen schon vorab zur Verfügung.

Das Erlernen der Bedienung eines graphischen CVS-Front-Ends wird durch den Lehrfilm abgedeckt. Dieser konzentriert sich auf die wichtigsten Funktionen eines CVS-Systems aus Entwicklersicht. Grundbegriffe des VKM und die Struktur eines CVS-Systems werden nur knapp behandelt und sollten im Prinzip durch Vortrag oder Lektüre vorher erlernt worden sein[2]. Die Rolle des Lehrfilms liegt vor allem darin, Anschauung zu vermitteln, typische Arbeitsabläufe zu zeigen und Berührungsängste abzubauen; Ziel ist nicht, die Funktionalität von CVS oder dem Front-End flächendeckend und systematisch durchzugehen.

Die Funktionen von CVS werden im Film indirekt in den Beispielen erklärt; in den Lehrtexten sind diese Funktionen detaillierter und für das Kommandozeileninterface beschrieben. Die Lehrtexte decken auch einige Themen ab, die nur bei einer vertieften Behandlung interessant sind oder zu deren Behandlung ein Lehrfilm weniger geeignet ist, u.a.

- Sperrkonzepte in Versionsarchiven, Benutzungsregeln für Versionsarchive
- Dokumentdifferenzen, Mischverfahren und -Werkzeuge
- Architektur eines CVS-Systems, Initialisierung eines Repositorys, Kommandozeilenschnittstelle von CVS, Notifizierung

Die Nutzung der Kommandozeilenschnittstelle wird zusätzlich durch ein Linux-Shell-Skript erklärt, das in Form einer live-Demonstration einige typische Arbeitsschritte auf einem frisch angelegten Repository durchführt.

Neben den eigentlichen Lehrmaterialien wurden noch ergänzende Materialien erstellt, u.a. Handreichungen für Betreuer von Lehrveranstaltungen, in denen ein CVS-Server installiert werden muß.

3.3 Der CVS-Lehrfilm

3.3.1 Inhaltsübersicht

Bei dem CVS-Lehrfilm handelt es sich um einen ca. 30 Minuten langen Videofilm, der die grundlegenden Arbeitsabläufe bei der Benutzung von CVS anhand von Beispielen zeigt.

Genauer gesagt wird die Benutzung eines graphischen Front-Ends zu CVS gezeigt. Bild 1 zeigt ein Beispielbild aus dem Film.

Das CVS-System als solches hat “nur” eine Kommandozeilen-Bedienschnittstelle. Letztere wird von erfahrenen Benutzern oft bevorzugt, für Anfänger sind dagegen graphische Bedienschnittstellen besser geeignet. Es gibt eine größere Anzahl graphischer Front-Ends zu CVS, die die Funktionalität von CVS in Menüs und Dialoge “verpacken”. In den Details unterscheiden sich die Front-Ends durchaus. Hauptkriterien für das im Film zu verwendende Front-End waren, daß es auf allen gängigen Plattformen möglichst kostenlos verfügbar ist und zuverlässig funktioniert. Im Rahmen einer Marktstudie ergab sich, daß die Systemfamilie MacCvs / gCvs / WinCvs (Varianten des “gleichen” Systems für MacIntosh, Linux bzw. Windows-Betriebssysteme) die Anforderungen am besten erfüllte.

[2] Die Einsatzerfahrungen haben allerdings gezeigt, daß der Film auch von Zuschauern ohne diese Voraussetzungen gut aufgenommen wird.

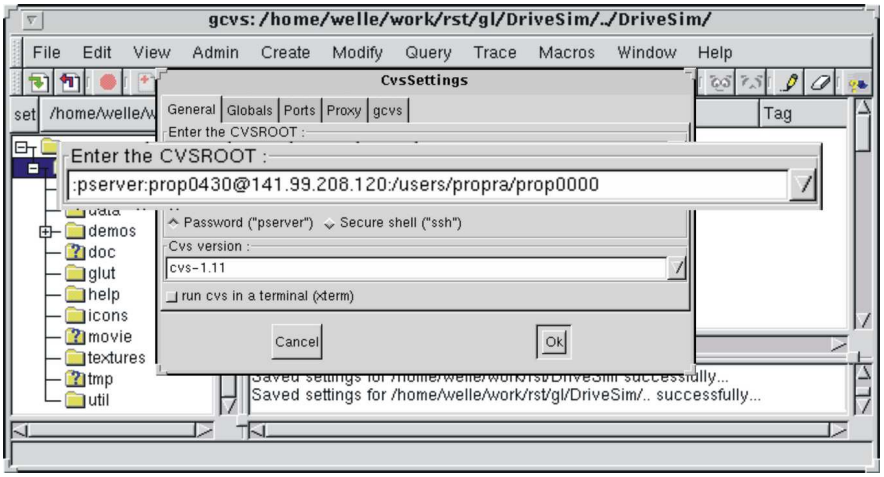


Abbildung 1: Beispielbild aus dem CVS-Film

Der Film zeigt primär ein gCvs-Fenster mit den Mausbewegungen darauf sowie den jeweiligen Ausgaben des Systems, ferner stellenweise Animationen (Bild 2 zeigt ein Beispiel); die Bilder werden durch gesprochenen Text zusätzlich erläutert.

Der Film war ursprünglich unter der Annahme konzipiert worden, daß er einzelnen Gruppen eines Programmierpraktikums (ca. 5 - 10 Personen) in einem kleineren Raum per Beamer gezeigt wird. Der Film ist daher in 10 Abschnitte[3] von ca. 1.5 - 6 Minuten Länge gegliedert, zwischen denen jeweils eine Diskussionspause eingelegt werden kann bzw. sogar sollte. Die Themen der Abschnitte sind:

Szene 0: Beschreibung des Szenarios
Szene 1: Ein bereits bestehender Workspace
Szene 2: Das erste Einloggen
Szene 3: Checkout vom zentralen Repository
Szene 4: Bearbeitung einer Datei
Szene 5: Dateien neu anlegen
Szene 6: Aktualisierung der lokalen Daten
Szene 7: Paralleles Arbeiten in mehreren Workspaces
Szene 8: Tagging
Szene 9: Parallele Entwicklungszweige

[3] Anfangs nur 8, die beiden letzten Abschnitte kamen erst später hinzu.

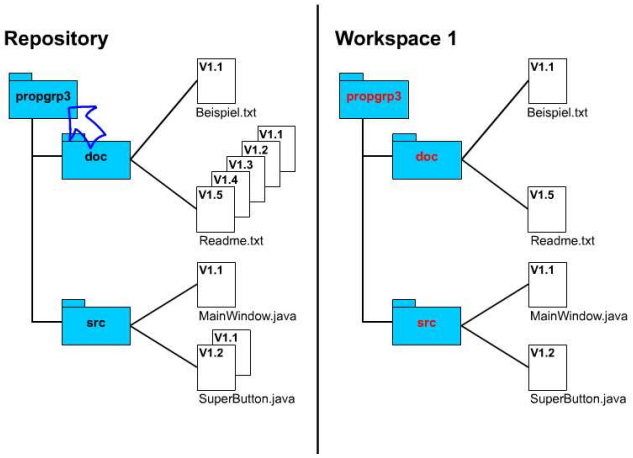


Abbildung 2: Beispiel einer Animation aus dem CVS-Film

3.3.2 Einsätze und Distribution

Eine erste, noch unvertonte Version des Films wurde im Wintersemester 2001/02 testweise im Programmierpraktikum in Siegen bei 8 Gruppen eingesetzt. In den folgenden Semestern wurde er mehrfach erweitert und in diversen Details verbessert:

- zusätzliche Vor- und Nachspanne
- Behebung diverser kleinerer Mängel
- Einfügung von kurzen Flash-Animationen an den Stellen, wo gerade ein CVS-Kommando ausgeführt worden ist. Die Animationen zeigen graphisch den Effekt des Kommandos.
- Erweiterung um die beiden letzten Szenen
- Integration visueller Effekte (z.B. Hervorhebung und Vergrößern von gerade benutzten Eingabefeldern)

Die jeweils aktuellen Versionen wurden in Siegen ab 2001 in jedem Semester im Programmierpraktikum (ca. 12 - 17 Gruppen zu 5 - 7 Personen) vorgeführt, ferner in der Softwaretechnik-Vorlesung.

Die im Rahmen des Projekts erstellten Materialien sowie die zugrundeliegenden Softwaresysteme wurden bei mehreren Zwischenständen zu einer CD zusammengestellt, die an interessierte Projektpartner des MuSofT-Projekts weitergegeben wurde. Das Material wurde ferner an zwei externe Kooperationspartner weitergegeben: an die Fachhochschule Gelsenkirchen / Bocholt (Prof. Convent), wo das Material in in einem sehr ähnlichen Kontext wie in Siegen eingesetzt wurde, nämlich in einem Programmierpraktikum, und an die Universität Oldenburg (Prof. Hasselbring), wo es nur in einer Softwaretechnik-Vorlesung eingesetzt wurde, also in einem deutlich anderen Kontext als in Siegen.

Ende 2002 wurde zusätzlich ein Streamingserver installiert und der Film in das zugehörige Format (Realmedia) konvertiert und über den Server abrufbar gemacht. Über den Streamings-

erver wurde der Film von einer größeren Zahl externer Interessenten ganz oder teilweise abgerufen. Da keine Zugangshindernisse errichtet wurden, ist abgesehen von wenigen Einzelfällen nicht bekannt, wie sich diese Nutzerschaft zusammensetzt.

3.3.3 Evaluation

Der Film wurde bei jedem Einsatz in Siegen durch Einsatz von Fragebögen evaluiert. Ein erster, auf den unmittelbaren Eindruck orientierter Fragebogen wurde unmittelbar nach Vorführung des Films ausgefüllt, ein zweiter, mehr auf den Lernerfolg zielender Fragebogen ca. 2 Wochen später. Auch an der FH Bocholt wurde der Film intensiv evaluiert.

Die Fragebögen wurden im Verlauf des Projekts mit freundlicher Unterstützung der Projektpartner in Lübeck hinsichtlich der Fragetechnik verbessert; der Gegenstand der Fragen änderte sich nicht wesentlich.

Durch die Evaluierungen wurden diverse Möglichkeiten erkannt, den Film zu verbessern. Der Lehrfilm wurde weit überwiegend als nützlich und hilfreich angesehen. In der Tat verlief die Benutzung des zentralen CVS-Repositories in den Praktika problemlos, obwohl es sich um eine für die Studenten völlig neue Technologie handelt.

Überraschenderweise stieg nach Einführung der zusätzlichen Animationen der Anteil der Befragten, die das Tempo des Films als etwas zu langsam empfanden, signifikant an. Dies geht sehr wahrscheinlich darauf zurück, daß die Animationen für einen Teil der Zuschauer nicht nötig gewesen sind und zu einer mentalen Unterforderung geführt haben.

3.4 Erfahrungen bei der Entwicklung des CVS-Films

Der Film wurde gegenüber seiner Ursprungsversion mehrfach verändert und erweitert. Hierbei wurden einige Erfahrungen gemacht, die für ähnliche Vorhaben nützlich sein können.

Synchronisation von Bild und Ton. Die Synchronisation von Bild und Ton ist auch bei dieser Art von Filmmaterial sehr kritisch. Dies mag zunächst überraschend klingen, denn, da “nur” Bildschirme gefilmt werden, braucht man keinen lippensynchronen Ton. Es zeigt sich allerdings, daß die Bewegungen der Maus und eventueller Texteingaben auf dem gezeigten Bildschirm sehr genau zum gesprochenen Text passen müssen und daß bereits ein geringfügiges zeitliches Auseinanderdriften als sehr störend empfunden wird.

In diesem Zusammenhang hat sich das Vorgehen, das bei der Realisierung der ersten Version des Films gewählt wurde, als ungünstig erwiesen. Das Vorgehen war in folgende Schritte gegliedert:

1. inhaltliche Planung, Entwicklung der Beispiele
2. Schreiben eines Drehbuchs incl. Szenenfolge und gesprochenem Text
3. Bildaufnahme, wobei die Person, die den Rechner bedient, zugleich den Text spricht, um das richtige Bewegungstempo zu haben.
4. Tonaufnahme, wobei das vorher aufgenommene Video abgespielt wird und zugleich der Text von Blatt gelesen wird.

5. Synchronisation von Bild und Ton

6. Kodierung bzw. Komprimierung des Materials

Die Synchronisation von Bild und Ton (Schritt 5) erwies sich als äußerst zeitraubend. Wenn ein gesprochener Satz und der Bewegungsablauf auf den Bildschirm nicht ausreichend synchron verlaufen, läßt sich dies nachträglich nur schwer beheben. Der Ton läßt sich praktisch nicht stauchen oder strecken, bei den Bildaufnahmen kommen allenfalls die Pausen für Streckungen infrage. Daher muß sehr viel ausprobiert werden. Der Aufwand für die Synchronisation lag letztlich bei 1 - 2 Stunden Arbeitszeit pro Minute Film. Da die mit diesem Arbeiten betraute Arbeitskraft nicht ganztags und zu beliebigen Zeiten zur Verfügung stand, kam es außerdem zu längeren Wartezeiten, die relativ störend wirkten und unökonomisch sind.

Bei späteren Aufnahmen wurden die Schritte 3 und 4 vertauscht; dies reduzierte den Aufwand, weil man das Bild leichter an den Ton anpassen kann als umgekehrt: bei der Bedienung von Maus und Tastatur kann man zugleich den schon vorhandenen Ton hören und die Bewegungen an das gesprochene Wort anpassen.

Länge von Pausen und Sprech- bzw. Anzeigetempo. Als ein ziemlich lästiges Problem stellte sich die richtige Wahl der Länge von Pausen (sowie von Vor- und Nachspannteilen) und allgemeiner die Wahl des Sprech- und Anzeigetempos heraus. Zu kurze Sprechpausen zwischen den Sätzen (bzw. Szenen) waren eine der häufigsten Ursachen für nachträgliche Änderungen an den Filmen[4].

Derartige Schwachpunkte stellen sich leider i.d.R. erst dann heraus, wenn ein Film komplett synchronisiert ist. Es ist bei Kontrollen nicht ganz einfach, derartige Fehler zu notieren – wenn man den Film anhält, um Notizen zu machen, hat man den Probelauf nachhaltig gestört! – und eine genaue Korrekturvorschrift anzugeben (die Angabe, daß die Pause nach dem Satz, der 3:15 Minuten nach Beginn der Szene endet, zu kurz war, ist keine direkt umsetzbare Änderungsanweisung).

In der Entwicklergruppe fehlte die Kenntnis einer “Theorie”, nach der z.B. die Länge von Sprechpausen schon im Vorfeld definiert und die systematische Einhaltung entsprechender Regeln kontrolliert werden kann. Eine solche Theorie ist umso wichtiger, wenn, wie hier, immer wieder im Abstand von mehreren Wochen kleinere Änderungen an dem Video vorgenommen werden und das Timing je nach der Tagesform der Beteiligten variiert.

Anpassung der Informationsdichte. Welches Tempo bzw. welche Informationsdichte als angenehm empfunden wird, ist in einem gewissen Rahmen nur subjektiv zu beurteilen. Im Gegensatz zu einem Sprecher, der unterbrochen werden kann oder der bemerkt, daß das Publikum unaufmerksam wird, weil der Inhalt zu langsam vorankommt, und dann das Tempo erhöht, läuft ein Film immer im gleichen Tempo weiter.

Hinzu kommt, daß bei einer längeren Laufzeit wie der hier vorliegenden (rund 30 Minuten, wenn man alle Szenen ohne Unterbrechung abspielt) durchaus damit zu rechnen ist, daß bei den Zuhörern zwischendurch die Konzentration nachläßt. Ein Vortragender könnte z.B. durch

[4] Die Pausen werden übrigens bei der Synchronisation festgelegt, da im Prinzip jeder einzelne aufgenommene Satz separat angeordnet wird.

Tempoverlangsamung oder Wiederholung und / oder Zusammenfassung von Stoff ad hoc eine "Durchhängephase" einbauen.

Sofern man die Szenen einzeln ansieht bzw. nach jeder Szene in der Gruppe kurz über die vorangegangene Szene spricht, tritt das Problem praktisch nicht auf. Man kann hieraus schlußfolgern, daß das Timing nur für jeweils eines der Nutzungsszenarien (Ansehen einzelner Szenen vs. ununterbrochenes Ansehen aller Szenen) optimal gestaltet werden kann.

Rapid Prototyping des Drehbuchs. Das Drehbuch beschreibt den Ablauf einer Szene; es beinhaltet insb. den gesprochenen Text und beschreibt die Vorgänge auf dem Bildschirm. Letztere wurden bei den ersten Versionen des Films eher summarisch beschrieben.

Selbst bei einer detaillierten Angabe der Vorgänge fällt es schwer - sofern man nicht sehr viel Erfahrung in der Produktion derartiger Filme besitzt - sich schon beim Schreiben des Drehbuchs die optische Wirkung - die u.a. von den Ausgaben und Reaktionen des gezeigten Systems abhängt - und das Timing klarzumachen. Die Problem ist ähnlich wie beim Schreiben eines Papiers oder Vorbereiten eines Vortrags, wo auch in der Regel die erste Version des Textes bzw. der Folien Fehler oder Schwächen (langatmige Passagen oder Gedankensprünge) haben. Bei Texten kann man die Schwächen durch wiederholtes Lesen im Zusammenhang i.d.R. erkennen, bei Vorträgen durch einen Probevortrag. Analog dazu braucht man bei Filmen eine Probeaufnahme bzw. -Vorführung, andernfalls entdeckt man Schwächen im Drehbuch erst dann, wenn der Film komplett produziert ist und eine nachträgliche Änderung einen enormen Aufwand verursacht.

Probeaufnahmen sind vom Konzept her direkt vergleichbar mit dem Rapid Prototyping bei der Software-Entwicklung. Dieser Ansatz wurde (leider) erst bei den beiden letzten Szenen systematisch angewandt und hat sich sehr bewährt. Er bestand darin, in einer Gruppe bestehend aus der Sprecherin, einem Bediener des Rechners und einem Aufnahmeleiter die neuen Szenen abschnittsweise "live" aufzunehmen, also hier auch unter Verwendung eines konkreten CVS-Systems, anschließend kurze oder auch längere Teile der Aufnahmen zu kontrollieren und ggf. das Drehbuch abzuändern oder ggf. Varianten derselben Passage auszuprobieren[5].

Für das Drehbuch zu ca. 4.5 Minuten Film verbrauchte die Gruppe für das Rapid Prototyping Arbeitszeit in der Größenordnung von 25 Stunden. Dieser Aufwand wirkt auf den ersten Blick hoch, er ist aber klein in Relation zu den Gesamtkosten der Filmentwicklung und erst recht im Vergleich zum Aufwand, der für die Korrektur eines Fehler entsteht, der erst in der Endversion des Films entdeckt wird.

Bei der Begutachtung der Testversionen des Films wurde in der Tat eine Vielzahl von kleineren oder größeren Schwächen gefunden und behoben. Ein bemerkenswerter Unterschied zur früher trat auch beim Detaillierungsgrad des Drehbuchs auf: Die Vorgänge auf dem Bildschirm wurden deutlich genauer als früher beschrieben, z.B. in Form genauer Anweisungen, bis zu welchen Wort im gesprochenen Text der Mauszeiger auf einem bestimmten Eintrag in einem Menü angekommen sein muß, wie lange er dort stehenbleiben soll usw. Durch die höhere Präzision der Angaben im Drehbuch waren auch die späteren Hauptaufnahmen einfacher reproduzierbar.

[5] Zur Verwaltung von Versionen und Varianten von - am besten in LaTeX geschriebenen - Drehbüchern eignet sich übrigens das System CVS sehr gut.

Wiederholung interaktiver Ausgaben. Sowohl bei den vorstehend beschriebenen Testaufnahmen wie auch später bei den Hauptaufnahmen kommt es immer wieder durch Versehen bei der Bedienung, falsche Eingaben oder sonstigen Störungen dazu, daß Aufnahmen wiederholt werden müssen. Dies ist aber bei einem System wie CVS / gCvs meist nicht ohne weiteres möglich; Ursachen hierfür sind:

- Bei dem fehlgeschlagenen Aufnahmeversuch ist trotzdem der Zustand des Repositorys verändert worden, z.B. kann eine Versionsnummer erhöht worden sein. Eine Wiederholung der Bedienung würde die Nummer erneut erhöhen und zu Anzeigen und Zuständen führen, die nicht mehr zum Drehbuch passen.
- Der fehlgeschlagene Aufnahmeversuch erzeugt Ausgaben im Protokollfenster von gCvs; diese bleiben auch beim Folgeversuch sichtbar.

In beiden Fällen ist der Zustand des Repositorys bzw. Front-Ends praktisch unbrauchbar für weitere Aufnahmen geworden und kann auch nicht durch eine Undo-Funktionen zurückgesetzt werden - eine solche gibt es systembedingt nicht. Es bleibt hier bzgl. des Repositorys nichts anderes übrig, als es komplett zu löschen und neu bis zur entsprechenden Stelle aufzubauen, was manuell einen hohen Aufwand verursacht. Um diesen zu vermeiden, wurden schon vor den Aufnahmen Shell-Skripten vorbereitet, die das Repository frisch installieren und bis zu bestimmten Wiederaufsetzpunkten des vorzuführenden Beispiels rekonstruieren.

Komprimierung. Einen erheblicher Aufwand verursachten die Komprimierung und die Wahl eines geeigneten Komprimierungsverfahrens (Codec). Die gängigen Verfahren sind auf Bildmaterial abgestimmt, das typisch für Fernsehfilme ist. Aufnahmen von Bildschirmen unterscheiden sich hiervon ganz erheblich. So dürfen durch die Komprimierung z.B. keine Kanten verwischt werden, und die hohen Kontraste zwischen Buchstaben (genauer gesagt Linien, die zufällig Buchstaben darstellen) und dem Hintergrund sollten erhalten bleiben.

Die ersten Versionen des Films wurden im AVI-Format mit dem Codec DIVX 4.12 erzeugt. Die Versuche führten letztlich zu wenig zufriedenstellenden Ergebnissen.

Im Verlauf des Projekts wurde im Zusammenhang mit der Installation eines Streamingserver das dazugehörige Realmedia-Format untersucht. Vorüberlegungen hinsichtlich der Wahl der Streamingtechnologie ergaben, daß die Realmedia-Plattform für die Situation in der Fachgruppe Praktische Informatik am besten geeignet ist.

Das Realmedia-Format hat die Besonderheit, mit einer variablen, dynamisch angepaßten Framerate zu arbeiten. Wenn viel Bewegung im Bild vorhanden ist, wird die Framerate bis auf einen vorgegebenen Maximalwert erhöht, um die Bewegung möglichst fließend darzustellen. Ändert sich wenig im Bild, so wird die Framerate bis auf 1 fps zurückgesetzt. Dieses Verfahren erwies sich für den CVS-Film als sehr günstig, denn die Bewegungen in dem Film sind nur relativ langsame Dateneingaben in Eingabefeldern, Mausbewegungen und einige Überblendeffekte. Trotz hoher Auflösung und sehr guter Bildqualität, bei der auch scharfe Kanten mit starkem Kontrast ohne Unschärfe und Grauschleier darstellt werden, sind die entstandenen Dateien kleiner als bei DIVX oder vergleichbar groß.

Ein weiterer großer Vorteil des Verfahrens liegt darin, daß man sich bei der Komprimierung praktisch nicht um die Framerate kümmern muß, während bei anderen Verfahren zeitaufwendige Experimente mit verschiedenen Frameraten erforderlich waren.

Leider traten bei den letzten Versionen des Films, in denen verstärkt Animationen und graphische Effekte enthalten sind, bei der Erstellung der Realmedia-Versionen wiederholt erhebliche Probleme auf, die sich in Form von störenden Darstellungsfehlern an einzelnen Stellen des Films äußerten und die sehr zeitraubende Kontrollmaßnahmen und Behebungsversuche erforderlich machten. Die Ursache der Fehler konnte nicht endgültig ergründet werden.

3.5 Copyright-Probleme

Die in dem CVS-Film benutzte Software (WinCVS) unterliegt der GNU General Public License (GPL), so daß es zunächst problemlos erschien, das System für den Film zu nutzen. Dies erwies sich überraschenderweise als unzutreffend.

Das Problem liegt darin, daß die GPL primär das Ausführen, Kopieren, Verteilen und Verändern einer Software behandelt und nur diese Nutzungen allgemein erlaubt. Wesentlich unklarer wird entschieden, was mit den Ausgaben eines Programms gemacht werden darf. Die Rechte an den Ausgaben gehören nur dann dem Produzenten, wenn die Ausgaben ein “eigenes”, also nicht allein durch das Programm erzeugtes Werk darstellen. Auch durch Rückfragen bei Rechtsexperten konnte nicht geklärt werden, wo die exakten Grenzen dieses Begriffs liegen.

Sicherheitshalber wurde daher versucht, mit den Rechteinhabern in Kontakt zu treten und ein explizites Einverständnis zur Nutzung des Systems für den Film einzuholen. Bei Produkten, die unter die GPL fallen, existiert ein eindeutiger Rechteinhaber bzw. Ansprechpartner in Form einer natürlichen oder juristischen Person meist nicht. Im Falle von gCvs gelang es immerhin, über die Distributionsplattform in Kontakt mit einem der betreuenden Entwickler zu treten, der die Erlaubnis bestätigte und zusätzlich auf der Distributionsplattform einen Link auf den Videoserver einrichtete, seinem Einverständnis also hierdurch zusätzlichen Ausdruck verlieh.

4 Projektplanung und -Verfolgung

4.1 Stoffauswahl und Lernziele

Aus dem Themengebiet Projektplanung und -Verfolgung werden Methoden zur Aufwandsschätzung und die Netzplantechnik häufig in Lehrbüchern behandelt, da sie mit überschaubarem Zeitaufwand vermittelbar und in der beruflichen Praxis relevant sind. Sie erfüllen also zumindest die in Abschnitt 2.1.1 erwähnten Auswahlkriterien. Problematisch ist indessen die geringe oder fehlende Anwendungsmöglichkeit im Studium:

- Die gängigen Aufwandsschätzungsmethoden sind praktisch nicht anwendbar, denn den Studenten fehlen fast immer die notwendigen Vergleichsdaten aus früheren Projekten.
- Netzpläne können im Hauptstudium, insb. bei Projektgruppen, Studienarbeiten und Diplomarbeiten, ohne weiteres eingesetzt werden; im Grundstudium sind die praktischen Arbeiten z.B. im Rahmen des Programmierpraktikums nicht umfangreich genug, einen Einsatz sinnvoll erscheinen zu lassen.

In Teilprojekt 3.4 wurden daher im Themengebiet Projektplanung nur für die Netzplantechnik Materialien erstellt. Als Basis waren bereits vorab konventionelle Lehrmaterialien (Volltexte und Vortragsfolien) für die grundlegenden Themen wie

- Methoden der Aufwandsschätzung,
- Netzplantechnik, elementare und geschachtelte Netzpläne, Netzplanauswertung, Kapazitätsplanung und -Ausgleich sowie
- Aufwandserfassung

vorhanden. Ein Schwachpunkt der konventionellen Lehrmaterialien liegt bei der Vorführung von Beispielen, wie Netzpläne entwickelt und ausgewertet werden. Daher konzentrierten sich die Arbeiten in diesem Bereich auf die Entwicklung eines Editor-Simulators für Netzpläne.

4.2 Der Editor-Simulator für Netzpläne

4.2.1 Systembeschreibung

Realisiert wurde in mehreren Revisionen ein in Java geschriebenes Programm, mit dem man Netzpläne editieren und die Vorwärts-Rückwärts-Rechnung kommentiert ablaufen lassen kann. Merkmale des Programms sind:

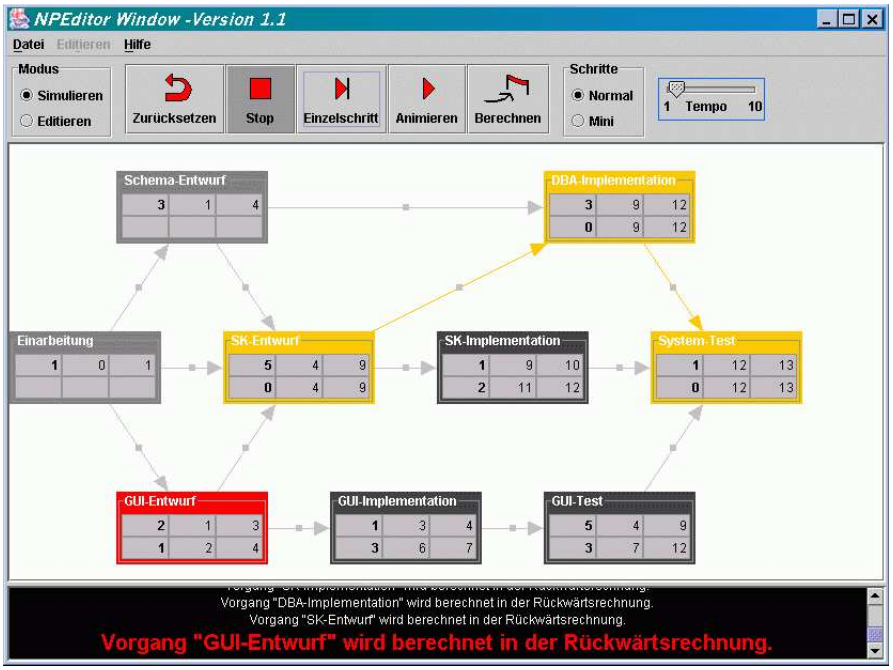


Abbildung 3: Beispielnetzplan im Editor-Simulator für Netzpläne

- Die Netzpläne, deren Auswertung simuliert werden soll, können sowohl hinsichtlich der Graphstruktur als auch der anderen Inhalte in der Unterrichtssituation jederzeit beliebig verändert werden, d.h. es handelt sich hier nicht um eine starre oder nur marginal modifizierbare Animation, sondern um einen (einfachen) Editor für Netzpläne und einen allgemeinen Simulator für die Vorwärts-Rückwärts-Rechnung.
- Die Bedienung wurde bewußt einfach gehalten, indem auf Funktionen, die bei den typischen kleineren Beispielen in Vorträgen nicht benötigt werden (z.B. Gruppierung und Kopieren), verzichtet wurde. Das System ist optimiert für den Einsatz in einem Vortrag (oder auch in einer Übungsgruppe bei der Bearbeitung von Übungsaufgaben) und nicht als Werkzeug für das Management großer Projekte gedacht. Eine ausführliche Darstellung der Regeln, nach denen das System gestaltet worden ist, findet sich in [1].
- Um Farbverfälschungen, die bei Beamern häufig auftreten, behandeln zu können, sind Dialoge vorhanden, über die der Nutzer alle verwendeten Farben einstellen kann.
- Bei der Simulation kann zwischen kleinen und großen Schritten gewählt werden. Alle Schritte werden in einem Protokoll-Unterfenster textuell erläutert.

4.2.2 Einsätze, Distribution und Evaluation

Das System wurde in Siegen jeweils in der aktuellen Version in der Vorlesung Softwaretechnik I angewandt. Aufgrund der Einsatzerfahrungen kann davon ausgegangen werden, daß ein etwa 10 - 15 Minuten dauernder Abschnitt der Vorlesung sinnvoll mit dem System gestaltet werden kann. Hierbei ist folgender Ablauf sinnvoll: Zunächst wird an einem Beispiel mit ca. 5 - 7 Knoten die Modellierung eines Projekts gezeigt und die Vorwärts-Rückwärts-Rechnung anfangs in kleinen Schritten, später in großen Schritten simuliert. Anschließend wird nach Einführung des Begriffs kritischer Pfad derselbe im Netzplan gezeigt und durch Variation der Dauer einzelner Vorgänge verändert. Abschließend wird der Begriff freie Pufferzeit (ggf. nach Erweiterung des bisherigen Beispiels) erläutert.

Das System wurde ferner über die lokalen WWW-Seiten in Siegen abrufbar gemacht und auf diese Weise den Teilnehmern der Vorlesung und anderen Projektpartnern zur Verfügung gestellt.

Eine erste Version des Systems stand im Herbst 2001 zur Verfügung und wurde erstmals im Wintersemester 2001/02 eingesetzt. Die eigenen Einsatzerfahrungen in diesem und den folgenden Semestern führten zu einer Überarbeitung des Systems in vielen Details. Der Einsatz in den Übungsgruppen wurde nicht formal evaluiert, da die interessanten Erkenntnisse einfacher mündlich bzw. informell an die Entwickler weitergegeben werden konnten und da die Bereitschaft der Nutzer, schriftliche Darstellungen von Schwächen oder möglichen Verbesserungen anzufertigen, als sehr gering eingeschätzt wurde.

Literatur

[1] Kelter, U.: Gestaltungsrichtlinien für Editor-Simulatoren für graphartige Dokumente; S. 393-400 in: Schubert, S.; Reusch, B.; Jesse, N. (ed.s): Proc. GI Jahrestagung, 30.9.-3.10.2002, Dortmund; Lecture Notes in Informatics; Gesellschaft für Informatik; 2002/10

KT1-Abschlussbericht

Johannes Magenheim

Zentraler Inhalt des Abschlussberichts des Koordinationsteam 1 *Gestaltung von Lernmodulen* sind die didaktischen Leitlinien, die vom Koordinationsteam als didaktisch-methodischer Orientierungsrahmen für die Lernmodule in den einzelnen Teilprojekten entwickelt wurden. Ferner werden die vom Koordinationsteam entwickelten Kriterien zur Beurteilung der Qualität von Materialien vorgestellt, die als Ergebnis des Projekts auf dem MuSofT-Portal eingestellt wurden.

Inhaltsverzeichnis

1 Einleitung

Das Koordinationsteam 1 *Gestaltung von Lernmodulen* hatte innerhalb des MuSofT-Projektes vornehmlich die Aufgabe, Leitlinien bereitzustellen, die den Projektpartnern Unterstützung und Hilfestellung bei der Entwicklung von an didaktisch-methodischen Kriterien orientierten Lerneinheiten, Lernmodulen und Gruppenobjekten geben.

Schon zu Beginn der Projektlaufzeit hat das Koordinationsteam didaktisch-methodische Leitlinien für diese Zwecke entwickelt. Diese wurden dann in der Folgezeit, vor allem während der gemeinsamen Treffen der Projektpartner im Plenum hinsichtlich der Umsetzung in den Teilprojekten diskutiert. Die Projektarbeit im Jahr 2003 war bei allen Partnern vornehmlich durch die Vervollständigung einzelner Module und von einzelnen Lernobjekten geprägt, in denen die Vorschläge des KTs integriert werden sollten. Im Rahmen von Projekttreffen wurden auch im letzten Jahr der Projektlaufzeit der jeweilige Arbeitsstand der Gruppen zwischen den Partnern ausgetauscht und im Rahmen von Plenumsdiskussionen u.a. mit anwesenden Mitgliedern des KT1 hinsichtlich realisierter didaktischer Konzepte diskutiert. Einzelne Teilprojekte wurden einer internen Evaluation unterzogen, an deren Gestaltung Mitglieder des

KTs zum Teil in beratender Funktion, zum Teil direkt beteiligt waren. Neben dieser allgemeinen Koordinationsfunktion hat das KT 1 einen Fragebogen zur Bewertung der MuSofT Materialien für externe Nutzer entwickelt, der auf die Webseite des MuSofT-Projektes eingestellt wurde.

Dieser Bericht wird daher sowohl die allgemeinen Aufgaben des Koordinationsteams als auch den Prozess der Entwicklung von Evaluationsinstrumenten zum Gegenstand haben und eine abschließende Gesamteinschätzung der Tätigkeit des KT 1 während der Projektlaufzeit von MuSofT geben.

2 Aufgabe des Koordinationsteams

Aufgabe des Koordinationsteams *Gestaltung von Lernmodulen* war es, den Teilprojekten Hilfestellung beim strukturell einheitlichen und gleichförmigen Aufbau und der didaktisch-methodischen Gestaltung der Lernobjekte auf allen Hierarchiestufen (Einheiten, Module und Gruppenobjekte) zu geben. Dazu wurden didaktisch-methodische Leitlinien für die Erstellung von Lernobjekten entwickelt.

Diese Hilfen erfolgten zwar vornehmlich in der Startphase des Projektes, doch wurden auch während der Phase der Materialproduktion in den einzelnen Teilprojekten auf Plenums- und Workshoptreffen didaktisch-methodische Fragestellungen in Orientierung an dem vom KT 1 entwickelten Schema für Lernobjekte diskutiert. Dabei wurden die zuvor entwickelten Kriterien zur Gestaltung von Lernobjekten in den einzelnen Teilprojekten angewandt und inhaltlich mit Bezug auf die je spezifischen Fragestellungen konkretisiert.

Darüber hinaus haben Mitglieder des KT1 ein Evaluations- und Bewertungsschema für bereits fertig gestellte Lernobjekte entwickelt, die für die Projektevaluation in den Teilprojekten aber auch für die nachhaltige Weiterentwicklung der Objekte/Module/Einheiten genutzt wurde.

Schließlich hat das Koordinationsteam einen Bewertungsfragebogen erstellt, mit dessen Hilfe es externen Nutzern der im Projekt MuSofT entwickelten Materialien ermöglicht wird, diese im Hinblick auf ihre Transferierbarkeit und Praxistauglichkeit in anderen Einsatzkontexten zu bewerten. Die in dem Fragebogen operationalisierten Bewertungsitems wurden in Anlehnung an die den Lernobjekten zu Grunde liegenden didaktischen Leitlinien entwickelt.

3 Aktivitäten und Ergebnisse

3.1 Didaktische Leitlinien

Lernobjekte für das MuSofT Portal werden entsprechend den von dem KT entwickelten didaktischen Leitlinien nach folgenden Kriterien klassifiziert: Leitbild, Lernziele, Lernszenario und methodisch-didaktisches Arrangement.

Das Leitbild bezüglich des Lernenden beinhaltet den individuellen, den institutionellen und den technischen Kontext des Lernens. Im individuellen Kontext werden kognitive und methodische Vorkenntnisse des Lernenden bzw. der Lerngruppe sowie angestrebte berufliche Qualifikationen und Einsatzfelder beschrieben. Der institutionelle Kontext wird durch die

Ausbildungsinstitution, den Studiengang, das Studienfach und schließlich durch den Studienabschnitt charakterisiert. Der technische Kontext kann mittels individueller und institutioneller technischer Voraussetzungen und Rahmenbedingungen definiert werden. Der Einsatz, die Funktion und der Nutzen der Lernobjekte ist nicht zuletzt von den jeweiligen Einsatzkontexten abhängig.

Alle Lernobjekte sollten auf Lernziele, abhängig vom jeweiligen Leitbild, ausgerichtet sein. Mit Lernzielen sind hier nicht nur die auf die Informatik bezogenen und fächerübergreifenden kognitiven Lernziele gemeint, sondern auch fachmethodische (informatikbezogene, fachübergreifende), sozial- kommunikative und normativ-bewertende Ziele.

Die Lernszenarien beinhalten Veranstaltungstypen (Präsenz-Vorlesung, Übung, Praktikum, Seminar, webbasiertes Studienmaterial, Projekte...), die Rolle der Lehrenden (Präsentieren, Organisieren von Lernprozessen, Beraten, Prüfen, Entwurfsentscheidungen), die Rolle der Lernenden (Rezipient, Selbststudium, Erkundender, Selbstorganisation von Lern- und Arbeitsprozessen), den Typ der Lernobjekte (Hypertext, sequentieller Text, Bilder, Grafiken, Animation, interaktive Elemente, Aufgaben, Tests, webbasierte Dokumente) und auch die Funktion dieser Objekte (Inhaltspräsentation, inhaltliche Differenzierung, Vertiefung, Erweiterung, Repetition, Übung, Prüfung..).

Das Arrangement von Lehrmaterialien bzw. Lernobjekten schließlich bezieht sich nicht nur bestimmte didaktische Konzeptionen, sondern auch auf die Methoden, mit denen die Lerninhalte vermittelt werden sollen (sequentiell lehrgangsmäßig - hypermedial erkundend; problemorientiert - prozessorientiert; fallbasiert - exemplarisch; instruktional - konstruktivistisch; konstruktiv induktiv - dekonstruktiv analytisch; projektartig; virtual company - real situation; virtuelle Erkundungsumgebung; produktorientiert).

Die didaktischen Leitlinien fanden Eingang in die Struktur und die Inhalte der Metadaten mit denen die Lernobjekte im MuSoft-Portal beschrieben werden. Näheres zu diesem Prozess ist in den Berichten der Koordinationsteams 2 und 4 nachzulesen. KT 2 befasst sich im Rahmen der inhaltlichen und stilistischen Abstimmung von Lerneinheiten u. a. mit der Nutzung von Metadaten nach dem LOM-Standard. Die von KT 1 entwickelten Leitlinien fungieren hier vor allem im ontologischen Bereich als differenzierende Kriterien. KT 4 ist für die Bereitstellung des MuSofT Internetportals zuständig, über welches die erstellten Lerneinheiten und die dazugehörigen Werkzeuge auf der Grundlage der von KT 2 vorgegebenen Festlegungen zu Metadaten, Werkzeugen, etc. angeboten werden. An dieser Stelle wird auch die enge Verzahnung der Tätigkeiten einzelner Koordinationsteams deutlich.

Das an den Leitlinien orientierte didaktisch-methodische Konzept und die Inhalte des MuSofT-Projektes wurden von Johannes Magenheim auf zwei BMBF-Workshops des Jahres 2002 vorgetragen: Auf dem kevih-Workshop (Konzepte und Elemente virtueller Hochschule) in Tübingen vom November 2002 und auf dem Workshop Didaktik und Evaluation von e-Learning in Erlangen.

3.2 Evaluationskonzepte für Teilprojekte

Unter der Federführung des Teams der Fachhochschule Lübeck (S. Seehusen, D. Irmscher-Lecon) wurde ein Evaluationskonzept entwickelt, das auf dem fünften MuSofT-Projekttreffen am Beispiel der Lehreinheit *Entwurfsmuster* der Fachhochschule Lübeck den Projektpartnern vorgestellt und exemplarisch konkretisiert wurde. Im Rahmen eines Vortrages wurde der Eva-

luationsbegriff begrifflich präzisiert und es wurden Fragestellungen diskutiert, die Gegenstand einer möglichen Evaluation im Rahmen von MuSofT sein könnten. Weiterhin wurden unterschiedliche empirische Vorgehensweisen, Instrumente zur Datenerhebung und Vorgehensweisen zur Auswertung der erhobenen Daten erläutert. Schließlich berichtete Frau Irmscher-Lecon als Vertreterin der Fachhochschule Lübeck über die geplante praktische Umsetzung des vorgestellten Evaluationskonzepts am Beispiel der Evaluation der Lerneinheit *Entwurfsmuster* im Sommersemester 2003.

In der sich anschließenden Diskussion im Plenum wurden Evaluationsvorhaben der Teilprojekte vor dem Hintergrund des vorgestellten Evaluationskonzepts diskutiert. Ferner wurden Verabredungen für konkrete Hilfestellungen der Lübecker Gruppe bei der Evaluation von Lehreinheiten anderer Teilprojekte verabredet.

Mitglieder des KT1 haben schließlich ein modifiziertes prozessbezogenes Evaluationskonzept für die Analyse von Lern- und Transferprozessen im Informatik Lernlabor am Beispiel der an der Universität Paderborn entwickelten MuSofT-Lerneinheit *Hochregallager* (LE 1.3) erstellt und durchgeführt.

3.3 Bewertung der Materialien

Vom KT 1 ist im Berichtszeitraum ein standardisierter Fragebogen für Lehreinheiten entwickelt worden, der im MuSofT-Portal zur Verfügung gestellt wird. Dieser Fragebogen soll ein Feed-back über die externe Anwendung der MuSofT-Materialien ermöglichen und so Hinweise über die Nachhaltigkeit und Transferierbarkeit der Projektergebnisse liefern.

Die im Fragebogen operationalisierten Items orientieren sich teilweise ebenso an den didaktischen Leitlinien, wie das bei den Kriterien zur Evaluation der MuSofT Teilprojekte bereits der Fall war. Darüber hinaus wurden im Fragebogen weitere Kriterien verwendet, die Nutzer/innen und Nutzungssituation an den externen Institutionen näher charakterisieren sollten. Hierzu gehörten neben Angaben zur Person und Institution der Nutzer/in vor allem Angaben zum Einsatz der Materialien. Sie bezogen sich beispielsweise auf die Organisationsform des Materialeinsatzes (Vorlesung, Übung, Praktikum, Seminar) und auf die dabei praktizierten Arbeitsformen (Frontalunterricht, Einzelarbeit, Arbeit in Kleingruppen, Arbeit in Großgruppen). Ferner wurden Angaben zu Art und Umfang der eingesetzten Materialien erbeten. Insbesondere erscheint es von Bedeutung, ob das Material als kompletter Kurs, lediglich in Form einzelner ausgewählter Kapitel oder gar nur eine Folienauswahl verwendet wurde bzw. einzelne Beispiele oder Materialabschnitte ausgewählt wurden. Ebenso wurden Hinweise auf verwendete Medienobjekte (Video-Material, Bild-Material, Ton-Material, einzelne Animationen) erbeten.

Die materialbezogenen Angaben wurden hinsichtlich der ausgewählten Inhaltsbereiche (Algorithmen und Datenstrukturen, Informationssysteme, Anforderungsdefinition, Entwurfsmuster, Software-Architektur, Unified Process, Konfigurationsmanagement, Qualitätssicherung) ergänzt und es wurde erfragt, wie stark die ausgewählten Materialien verändert bzw. angepasst werden mussten, um sie in die Lehre der externen Institution integrieren zu können.

Ein weiterer Abschnitt der Erhebung beschäftigte sich mit der Bewertung der eingesetzten Materialien durch die Lehrperson. Es wurde nach Gründen für den Einsatz der MuSofT-Materialien in der eigenen Lehre gefragt (Unterstützung des selbstorganisierten Lernens der Studierenden, Unterstützung von Gruppenarbeit, Unterstützung der eigenen Lehre durch ande-

re/zusätzliche Lehrmaterialien, Erproben neuer Ideen, Erleichterung der eigenen Lehre durch Verwendung von bereits entwickelten Materialien, Materialien ergänzten gut die eigene Lehre). Die externen Nutzer/innen wurden weiterhin um Informationen darüber gebeten, zu welchem Zweck sie die Materialien eingesetzt haben (Stoffvermittlung, Vertiefung von Inhalten der eigenen Lehre, Erläuterung von Inhalten der eigenen Lehre, als Demonstrationsbeispiele).

Schließlich wurden die Nutzer/innen um eine Bewertung der eingesetzten Lehrmaterialien im Hinblick auf deren technische Qualität, die inhaltliche und didaktische Aufbereitung der Materialien sowie die Qualität der materialbezogenen Dokumentation gebeten.

Eine Auswertung dieser Befragung liegt bisher noch nicht vor.

4 Gesamtresümee

Das KT 1 konnte mit den von ihm kooperativ mit den Teilprojekten entwickelten didaktischen Leitlinien den methodisch-didaktischen Aufbau der Lerneinheiten und der Lernobjekte mitgestalten und konnte auf diese Weise einen Beitrag zur einheitlichen Gestaltung der MuSofT-Materialien leisten. Die zur Beschreibung der Lernobjekte in MuSofT verwendeten Metadaten spiegeln auf der inhaltlichen Ebene diese Leitlinien wieder. Außerdem fanden die didaktisch-methodischen Leitlinien bei der Erstellung von Evaluationskonzepten und der Entwicklung des Bewertungsfragebogens zum externen Materialeinsatz durch das KT Berücksichtigung.

Literatur

[Mag03] MAGENHEIM, JOHANNES: *Wissensmanagement, Dekonstruktion und Learning Communities in der Softwaretechnik - Didaktische Konzepte im BMBF-Projekt MuSofT*. In: RINN, U. und J. WEDEKIND (Herausgeber): *Didaktik der neuen Medien*, Seiten 255–269, Münster, 2003. Waxmann.

KT2 - Abstimmung von Lehrmoduln

Andy Schürr

andy.schuerr@es.tu-darmstadt.de
FG Echtzeitsysteme, FB 18
TU Darmstadt

Das MuSofT-Koordinationsteam KT2 befasste sich mit der inhaltlichen und stilistischen Abstimmung aller entwickelten Lernobjekte, um so die Kombinierbarkeit von Lernmaterialien zu gewährleisten, die von verschiedenen Projektpartnern für ganz unterschiedliche Zielgruppen entwickelt wurden. Bei der im Vordergrund stehenden inhaltlichen Abstimmung von Lernobjekten spielte deren einheitliche Beschreibung durch sogenannte Metadaten eine herausragende Rolle. Hier ein Metadatenkonzept zu finden, das zu internationalen Standards kompatibel und dennoch in der Praxis des Projektalltags einsetzbar ist, erwies sich als anspruchsvolle Aufgabe. Ihr waren die meisten Aktivitäten von KT2 im MuSofT-Projekt gewidmet, die im vorliegenden Tätigkeitsbericht überblicksmäßig dargestellt werden.

Inhaltsverzeichnis

1 Einleitung

Um die in MuSofT an unterschiedlichen Standorten entwickelten Lernobjekte zur Realisierung einer Vorlesung einsetzen zu können, sind die an den verschiedenen Projektstandorten entstehenden Lerneinheiten *aufeinander abzustimmen*. Diese Abstimmung erfordert übergreifende Aktivitäten, die maßgeblich zur Qualitätsförderung in MuSofT beitragen. Deshalb wurden zu Beginn des Projekts zunächst vier Koordinationsteams eingerichtet, welche über die einzelnen Standorte hinweg tätig sind.

Wie wir im folgenden noch sehen werden, ließen sich die Aufgaben der einzelnen KTs nicht ganz unabhängig voneinander bearbeiten. Insbesondere das Koordinationsteam KT2 besaß mit seiner zentralen Aufgabe der *Abstimmung von Lernobjekten* Berührungspunkte zu allen anderen KTs. Auf diese Berührungspunkte können wir jedoch erst nach einer detailierteren Darstellung der Aufgaben von KT2 genauer eingehen.

Der Bericht über die Aktivitäten des Koordinationsteams KT2 ist deshalb wie folgt aufgebaut: Zunächst werden im folgenden Abschnitt die Aufgaben und Ziele von KT2 genauer beleuchtet sowie die sich daraus ergebenden Berührungspunkte zu den anderen KTs. Darauf aufbauend beschreibt der Hauptabschnitt dieses Berichts zunächst kurz unsere Aktivitäten zur Festlegung einheitlicher Vorgaben für Lernobjekte. Schwerpunkt dieses Abschnittes bildet allerdings die Entwicklung eines Metadatenkonzeptes für die einheitliche Beschreibung von Lernobjekten.

Die beiden letzten Abschnitte des Berichtes befassen sich schließlich mit den weiteren Aktivitäten von KT2 innerhalb der verbliebenen Projektlaufzeit und skizzieren, wie auf Basis der dabei gemachten Erfahrungen weiterführende Aktivitäten in künftigen Projekten aussehen könnten.

2 Aufgabe des Koordinationsteams

Die im vorhergehenden Abschnitt geforderte inhaltliche und stilistische Abstimmung von Lernobjekten durch KT2 ist kein Selbstzweck. Vielmehr sollen hierdurch die wesentlichen Grundlagen dafür bereitgestellt werden, dass (1) sich Lernobjekte für neue Zielgruppen durch *Kombination und Adaption bereits vorhandener Lernobjekte* erzeugen lassen und (2) die Anforderungen an ein gemeinsames MuSofT-Portal genauer festgelegt werden können. Um diesem Anspruch gerecht zu werden, wurden folgende drei Primärziele von KT2 formuliert:

1. die Nutzung von Lernobjekten in anderen Lernobjekten verschiedener Teilprojekte und die gleichzeitige Vermeidung von Redundanzen, die durch Doppelentwicklungen in verschiedenen Teilprojekten entstehen können,

2. die Identifikation fehlender Lernobjekte zur Vermittlung von Basiswissen, das von den erstellten Lernobjekten vorausgesetzt wird, und die Vergabe entsprechender Aufträge an Teilprojekte,

3. die Erstellung neuer Lernobjekte durch die Wiederverwendung vorhandener Ojekte bzw. die Anpassung bereits erstellter Lernobjekte an neue Zielgruppen.

Aus diesen Primärzielen lässt sich nunmehr ableiten, dass die inhaltliche und stilistische Abstimmung von Lernobjekten bzw. ihrer Bestandteile notwendig ist und welche Punkte dabei zu berücksichtigen sind. Auf *inhaltlicher* Ebene spielen die Dokumentation verschiedener Arten von Beziehungen zwischen einzelnen Lernobjekten, die Auswahl durchgängiger Fallstudien, die Festlegung der verwendeten Modellierungs- und Programmiersprachen sowie die einheitliche Beschreibung aller erstellten Lernobjekte durch sogenannte "Metadaten" eine herausragende Rolle. Auf der *stilistischen* Ebene geht es hingegen darum, Modellierungs- und Programmierkonventionen festzulegen sowie die Menge der alternativ eingesetzten Modellierungs- und Programmierwerkzeuge einzuschränken. Der stilistischen Ebene lässt sich schließlich auch der Punkt "einheitliches Layout" von Lehrmaterialien zuordnen.

Auf Basis dieser Festlegungen von KT2 und ähnlicher Überlegungen der anderen KTs wurden die folgenden Bereiche identifiziert, in denen sich die Aufgaben von KT2 mit denen der anderen Koordinationsteams überschnitten:

- KT1 war mit seiner Aufgabe der Festlegung didaktischer Richtlinien ebenfalls (wenn nicht sogar in erster Linie) für den einheitlichen Aufbau und die einheitliche Gestaltung der erstellten Lernobjekte zuständig. Des weiteren waren die Ergebnisse von KT1 für die Beschreibung der didaktischen Eigenschaften (Attribute) von Lernobjekten von entscheidender Bedeutung.

- KT3 spielte bei der Festlegung von Lehrinhalten sowie bei der Auswahl kleinerer Beispiel und größerer Fallstudien eine wichtige Rolle; hier wurde deren Eignung für ganz unterschiedliche Zielgruppen untersucht und sichergestellt. Ebenso beeinflussten die Ergebnisse von KT3 die Art und Weise, wie Lernobjekte durch Metadaten beschrieben werden sollten.

- KT4 schließlich hatte die Aufgabe übernommen, auf der Grundlage der von KT2 vorgegebenen Festlegungen zu Metadaten, Werkzeugen, etc. das MuSofT-Portal zu realisieren, also eine entsprechende Integrationsplattform für alle MuSofT-Ergebnisse bereitzustellen.

3 Bisherige Aktivitäten und Ergebnisse

Im vorhergehenden Abschnitt haben wir gesehen, dass die Beschreibung der Metadaten von Lernobjekten ein, wenn nicht der zentrale Punkt ist, in dem sich die Aufgaben der verschiedenen Koordinationsteams überschnitten und selbst wieder einer Koordination bedurften. Deshalb hatten wir in KT2 diesem Aspekt die höchste Priorität eingeräumt. Auf der Basis der Ergebnisse von KT1 und KT3 wurde ein Vorschlag für den Aufbau der Metadaten zur Beschreibung von Lernobjekten in MuSofT erarbeitet, der die Grundlage für die Arbeiten von KT4 zur Realisierung des MuSofT-Portals bildete.

Im Folgenden werden wir aus diesem Grund nur kurz auf die anderen Aktivitäten von KT2 zur Auswahl von Programmiersprachen, Werkzeugen, Fallbeispiele etc. eingehen und den Hauptaugenmerk auf das Thema *Metadaten für Lernobjekte* legen.

3.1 Notationen, Werkzeuge und Fallstudien

Trotz der Vielfalt der behandelten Softwaretechnikthemen, der mannigfaltigen Unterschiede der Curricula der beteiligten Universitäten und der großen Menge konkurrierender Methoden und Techniken in der Softwaretechnik ist es uns gelungen, für alle in MuSofT entwickelten Lehrmaterialien die folgenden Vereinbarungen zu treffen:

- Als Programmiersprache wurde - wo immer sinnvoll - Java eingesetzt und damit der objektorientierten Softwareentwicklung der Vorzug gegeben. Eine begründete Ausnahme von dieser Regel bildete der Datenbankbereich; dort haben sich objektorientierte Technologien gegenüber relationalen Datenbanken mit SQL als "Programmiersprache" bislang nicht durchgesetzt.

- Als Modellierungssprache wurde die UML verwendet, als Vorgehensmodell dem dazugehörigen "Unified Process" der Vorzug gegeben. Auch von dieser Regel wurde nur in einigen wenigen Fällen abgewichen; im Datenbankbereich wurden beispielsweise auch "Entity-Relationship"-Diagramme verwendet, während in Lehreinheiten, die das Thema Vorgehensmodelle selbst ansprechen, natürlich auch andere Standards wie das V-Modell zur Sprache kommen.

- Auf der Ebene der CASE-Tools fiel unsere Wahl auf das Together Control Center als das Standardwerkzeug, da es eine enge Integration von UML-Modellierungshilfsmitteln mit Java-Entwicklungswerkzeugen anbietet und zudem über einige Anpassungs- und Erweiterungsmöglichkeiten verfügt.

- Auf der Ebene der Metawerkzeuge zur Erstellung spezialisierter CASE-Tools für bestimmte Unterrichtseinheiten wurde nach einer Diskussion verschiedener Alternativen (Dome, MetaEdit, ArgoUML-Kern, jViews, yFiles, etc.) folgendes Resümee gezogen: die Ansprüche der einzelnen Projektpartner bei der Erstellung spezieller Projektplaneditoren, Architektureditoren, interaktiver Algorithmenanimationen usw. waren zu verschieden, um diese durch ein Meta-CASE-Tool abdecken zu können. Die Festlegung auf ein Werkzeug war deshalb an dieser Stelle nicht sinnvoll; umso mehr, als die "Nutzer" unserer Lehrmaterialien ein solches Meta-Werkzeug niemals direkt zu Gesicht bekommen.

- Ebenfalls als schwierig hat sich das Thema "einheitliche Fallstudien" erwiesen. Auch hier waren die Erfordernisse verschiedener Lerneinheiten so divergierend, dass die Festlegung *einer* zentralen Fallstudie als durchgängiges Beispiel für alle entwickelten Lehrmaterialien sich als nicht sinnvoll erwies. Immerhin ist es uns jedoch gelungen, dem Thema Logistik mit den Unterthemen Lagerverwaltung, Kommissionierung und Speditionswesens eine zentrale Rolle einzuräumen.

3.2 Vorüberlegungen zur Festlegung eines Metadatenformates

Die Beschreibung von Lernobjekten durch umfangreiche Sätze von Metadaten ist eine Funktion, die heutzutage von nahezu allen Lernplattformen, -portalen, etc. angeboten wird. Um die

Interoperabilität all dieser Werkzeuge in Zukunft zu gewährleisten, gibt es eine ganze Reihe von Standardisierungsbemühungen für Metadatenformate von Lernobjekten. Aus diesem Grunde war es nicht notwendig, in MuSofT ein eigenes Metadatenformat zu entwickeln. Vielmehr hat es sich als sinnvoll erwiesen, auf den IEEE LOM-Standard (*Learning Objects Metadata*) [IEE02] zurückzugreifen, der ein konzeptionelles Datenschema vorgibt, welches die Struktur von Metadaten für Lernobjekte beschreibt. Die Verwendung dieses Metadatenstandards erlaubt nicht nur eine einheitliche Beschreibung der Lernobjekte innerhalb von MuSofT, sondern ermöglicht auch eine Interpretation der verwendeten Metadaten über die Projektgrenzen hinaus. Unser Hauptgrund für die Auswahl des LOM-Standards war zum einen seine weite Verbreitung. Zum anderen wird der LOM-Standard nahezu unverändert in verschiedene Standardisierungsbemühungen, wie z.B. dem IMS Global Learning Consortium [IMS02], SCORM [Adv02] oder ARIADNE [DFC+01] integriert.

Obwohl LOM aus den oben genannten Gründen der für unsere Zwecke am besten geeignete Standard für die Beschreibung von Lernobjekten ist, war sein Einsatz im MuSoft-Projekt mit umfangreichen Vorarbeiten verbunden. So erschien es uns vor allem unrealistisch, *alle* von LOM vorgeschlagenen über hundert Metaattribute für die Dokumentation unserer Lernobjekte einzusetzen und zudem die tatsächliche Verwendung der LOM-Attribute ohne weitere Richtlinien den einzelnen Teilprojekten zu überlassen. Ein solcher ungeregelter Einsatz von LOM würde möglicherweise zu weit voneinander abweichenden Metabeschreibungen einzelner Lernobjekte führen und damit die oben aufgeführten Zielsetzungen konterkarieren.

3.3 Die Anpassung des LOM-Standards - Objekthierarchien

Der LOM-Standard schreibt eine genau vierstufige Hierarchie von *Lernobjekten (LOs, Learning Objects)* vor, ohne allerdings im Detail festzulegen, wie eine solche Hierarchie für die Gliederung von Lernmaterial eingesetzt werden soll. Nach einer längeren Diskussion - insbesondere darüber, ob in der Praxis eine Beschränkung auf vier Hierarchiestufen akzeptabel ist - haben wir in KT2 beschlossen, die LOM-Hierarchie wie folgt auf unsere Bedürfnisse zugeschnitten auszulegen:

1. Auf der obersten Ebene gibt es *Lerneinheiten (LEs, Learning Entities)*: eine LE ist eine sich abgeschlossene Sammlung von Lernmaterialien zu einem Themenkomplex. Typischerweise handelt es sich dabei um einen Bereich, der im Rahmen *einer* Lehrveranstaltung behandelt wird.

2. Lerneinheiten setzen sich aus *Lernmoduln (LMs, Learning Modules)* zusammen: ein LM ist eine Sammlung von Lernmaterialien, die in verschiedenen Lerneinheiten (wieder-)verwendet werden kann. Sie behandelt also ein in sich abgeschlossenes Teilgebiet, das Bestandteil verschiedener Lehrveranstaltungen sein könnte.

3. Für die weitere Unterteilung von Lernmoduln gibt es die *Gruppenobjekte (GOs, Group Objects)*: ein GO ist die kleinste Ansammlung von Lernmaterialien, die nicht als atomare Einheit betrachtet wird. In aller Regel wird ein Gruppenobjekt Bestandteil genau eines Lernmoduls sein, da es weder wohldefinierte Schnittstellen zu anderen Lernobjekten besitzt, noch einen in sich abgeschlossenen Themenkomplex behandelt.

4. Auf der untersten Ebene unserer Enthaltenseinshierarchie gibt es sogenannte *Medienobjekte (MEs, Media Objects)*: ein ME ist die kleinste Untereinheit einer Lerneinheit, deren interne Struktur auf der Ebene der Metadaten nicht weiter betrachtet wird. Es kann sich dabei um einen vollständigen Foliensatz, eine html-Seite, eine Animation, ... handeln. Üblicherweise wird ein Medienobjekt einer Datei entsprechen, es kann aber auch ein Verweis auf einen Ausschnitt einer Datei (z.B. Folie 10 bis 20 einer Powerpoint-Präsentation) sein. Oft werden solche Medienobjekte Bestandteil mehrerer Gruppenobjekte sein.

Die vier Stufen von LOM bilden damit also eine azyklische, aber i.a. nicht baumartige Hierarchie, deren Objekte im Folgenden durch Metaattribute und -beziehungen genauer beschrieben werden. Vergleicht man die hier getroffenen Festlegungen mit den unverbindlichen Vorschlägen von LOM zum Einsatz der vier Hierarchiestufen, so ist folgendes festzustellen: auf der niedrigsten Hierarchiestufe 1 (ME) sieht LOM nur einzelne Fragmente von Dokumenten wie etwa html-Seiten vor, während wir für größere Einheiten plädieren, die bereits ganzen Dateien entsprechen. Eine zu detailierte Modellierung von Lernobjekten erschien uns zu aufwändig und für die Ablage und Wiederverwendung von LOs auch wenig zielführend. Auf der obersten Hierarchiestufe 4 (LE) sieht LOM hingegen wesentlich größere Einheiten vor, nämlich ganze Curricula, also Mengen von Lerneinheiten in unserem Sinne. Auch diesem Vorschlag sind wir nicht gefolgt, da uns damit a) nur noch zwei Stufen für die Beschreibung von Medienobjekten, Lernobjekten, Lernmoduln und Lerneinheiten zur Verfügung gestanden hätten und b) uns die Ablage ganzer Curricula als in sich abgeschlossene Lernobjekte im MuSofT-Portal wenig sinnvoll erschien.

3.4 Die Anpassung des LOM-Standards - Metaattribute

Der LOM-Standard umfasst in seiner aktuellen Version über hundert Metaattribute, die in neun Kategorien unterteilt sind. Von diesen neun Kategorien haben wir nach langen Diskussionen fünf mit bislang 22 Metaattributen für die Verwendung in MuSofT ausgewählt. Dabei haben wir darauf geachtet, für jedes ausgewählte Attribut dessen Bedeutung so präzise wie möglich festzulegen und bei der Festlegung seiner Bedeutung konform zu den Erläuterungen des Standards zu bleiben. Auf die Hinzunahme neuer Metaattribute oder die Uminterpretation existierender Metaattribute haben wir bislang also gänzlich verzichtet, wenn auch gerade im Bereich der sogenannten “didaktischen Eigenschaften” der Status Quo von LOM für unsere Bedürfnisse bei weitem nicht ausreichend ist.

Im einzelnen verwendet die MuSofT-Adaption von LOM folgende Kategorien von Metaattributen:

1. *General* mit 8 Attributen für allgemeine Informationen wie eindeutige LO-Bezeichner, Titel, Schlüsselworte, Anmerkungen, etc.

2. *Technical* mit 3 Attributen für technische Eigenschaften wie Formate und “Adresse” von Dateien

3. *Educational* mit 6 Attributen für didaktische Eigenschaften wie Arbeitsformen, Einsatzkontext, Anspruch etc.

4. *Relation* mit 3 Attributen für die Beschreibung von Beziehungen zwischen Lernobjekten (unterstützt wird Versionierung, Hierarchiebildung, sowie drei verschiedene Arten von Querbeziehungen zwischen LOs)

5. *Classification* mit 2 Attributen für die Anbindung von LOs an verschiedene Klassifikationshierarchien (wie etwa das ACM-Klassifikationssystem)

Weitere Informationen zum Einsatz von Metadaten für die Beschreibung und Ermittlung von Lernobjekten im MuSofT-Portal sowie zur Ausgestaltung der Benutzeroberfläche des MuSofT-Portals findet man im Bericht des Koordinationsteams KT 4. Für eine vollständige Auflistung aller ausgewählten Metaattribute mit der Beschreibung der zulässigen Werte sei ebenfalls auf das MuSofT-Portal und die zugehörigen WWW-Seiten verwiesen.

3.5 Offene Punkte bei der Adaption des LOM-Standards

Als besonders schwierig bei der Anpassung des LOM-Standards für MuSofT erwies sich zum einen die Festlegung einer festen Hierarchie von Lernobjekten sowie zum anderen die Auswahl und Beschreibung der Metaattribute der Kategorie *Educational*. So ist es fraglich, ob eine vierstufige Hierarchie mit genau festgelegten Rollen der Lernobjekte auf den einzelnen Ebenen immer ausreichend ist oder ob man nicht lieber die Erstellung beliebig tiefer Hierarchien unterstützen sollte. Mit mindestens sechsstufigen Hierarchien könnte man beispielsweise die von uns benötigte Unterscheidung von Lernmoduln, Gruppenobjekten und Medienobjekten mit der von LOM empfohlenen Verwendung von allumfassenden “Curricula”-Objekten auf der einen Seite und sehr kleinen Dokumentfragmenten auf der anderen Seite zusammenführen; zudem würde eine beliebig tiefe Hierarchisierung beispielsweise auf der Ebene unserer Lernobjekte es uns erlauben, die Unterteilung Lehrmateralien (insbesondere von Büchern) in Kapitel, Abschnitte, Unterabschnitte, etc. mitzumodellieren.

Unbehagen löste auch die Tatsache aus, dass alle Metaattribute für die Beschreibung von Lernobjekten auf allen Hierarchiestufen zugelassen sind. Technische Attribute, wie etwa das Format eines Lernobjektes, scheinen auf den oberen Ebenen wenig Sinn zu machen, während hingegen didaktische Attribute, die z.B. den eingesetzten Studiengang eines Lernobjektes beschreiben, auf den unteren Ebenen fraglich erscheinen. Hier sind weitergehende Festlegungen dringend notwendig, die den Einsatz einzelner Metaattribute entweder für bestimmte Hierarchieebenen ganz verbieten oder aber ein entsprechendes Vererbungskonzept anbieten. Die genaue Ausgestaltung eines solchen Vererbungskonzeptes bedarf jedoch noch umfangreicher Überlegungen und Diskussionen. Beispielsweise könnte man sich bei einem Attribut wie den “Sprachen”, in denen die bereitgestellten Lernobjekte formuliert sind, folgende Vererbungsszenarien entweder alternativ oder sogar miteinander kombiniert vorstellen:

- “Sprache” ist ein *ererbtes* mengenwertiges Attribut, das auf oberster Ebene festgelegt werden kann und in die unteren Ebenen entweder immer unverändert propagiert wird oder dort ggf. eingeschränkt bzw. vielleicht sogar erweitert werden kann.

- “Sprache” ist ein *synthetisches* mengenwertiges Attribut; die Sprachen eines primitiven Lernobjektes werden explizit angegeben, die Sprachen zusammengesetzter Lernobjekte ergeben sich immer implizit aus der Vereinigung der Sprachen ihrer Bestandteile.

Abgesehen von diesen Überlegungen zur Einschränkung der Verwendung von Metaattributen und zum Einsatzung von Vererbungskonzepten für die Reduktion des Erstellungsaufwandes von Metadaten hat sich gezeigt, dass insbesondere die didaktischen Attribute eine unklare Semantik besitzen und in der vorliegenden Form kaum nutzbringend sind. An dieser Stelle wurden deshalb vom Koordinationsteam KT1 in Zusammenarbeit mit KT4 einfache Vorschläge für eine sinnvollere Definition didaktischer Attribute erarbeitet und in den Rahmen von LOM eingebettet, die allerdings aus didaktischer Sicht noch keine befriedigende Lösung darstellen.

Trotz der oben skizzierten Probleme haben wir mit der Anpassung des LOM-Standards einen gangbaren Weg gefunden, der den Aufwand für die Erstellung von Metadaten nicht in unrealistische Höhen treibt und trotzdem die für die (Wieder-)Verwendung von Lernobjekten benötigten Informationen bereitstellt. Letztendlich sind jedoch die Erfahrungen mit dem flächendeckenden Einsatz des MuSofT-Portal durch externe Anwender in den nächsten Monaten und Jahren nach Projektende abzuwarten.

4 Erfahrungen mit der LOM-Anpassung im MuSofT-Portal

Mit der Definition eines LOM-basierten Metadatenformates für Lernobjekte war eine der Hauptaufgaben von KT2 - wenn nicht die Hauptaufgabe von KT2 - abgeschlossen. Nunmehr war es die Aufgabe aller MuSofT-Teilprojekte unter Verwendung des von KT4 realisierten Portals, ihre erstellten Lernobjekte dort mitsamt der entsprechenden Metadaten einzustellen. Hierfür wurde von KT2 ein stufenweises Konzept vorgeschlagen, das zunächst die Erfassung von Lerneinheiten und Lernmoduln mit einem minimalen Satz von Attributen erforderte und darauf aufbauend Ergänzungen und Überarbeitungen vorsah.

Inzwischen habe alle Projektpartner ihre Ergebnisse in das MuSofT-Portal eingestellt und dokumentiert, in einigen Fällen sogar in mehreren Varianten für unterschiedliche Zielgruppen. Für die dabei gemachten Erfahrungen mit dem Portal und der Dokumentation von Lernobjekten sei auf den Projektbericht von KT4 in diesem Band verwiesen.

Weitere Überlegungen zur Entwicklung des oben angesprochenen Attributvererbungskonzeptes oder eines “vernünftigen” Modularisierungskonzeptes, das die Definition von Lernmoduln mit wohldefinierten Import- und Exportschnittstellen als Startpunkt und Ende von Querverweisen erlauben würde, konnten innerhalb der Projektlaufzeit leider nicht mehr in Angriff genommen werden.

5 Zusammenfassung

Im ersten Jahr der Laufzeit von MuSofT war es die Hauptaufgabe von KT2 gewesen, anhand größerer Fragebogenaktionen einen Überblick über die geplanten Entwicklungen von Lernmaterialien, die dabei zum Einsatz kommenden Sprachen, Methoden, Werkzeuge, Fallbeispiele etc. zu gewinnen. Auf dieser Basis wurden im zweiten Projektjahr eine Reihe von Festlegungen getroffen, die die Grundlage für die Kombinierbarkeit aller entwickelten Lernmaterialien bilden. Hierzu gehörte der einheitliche Einsatz von Java als Programmiersprache, UML als

Modellierungssprache, die Verwendung von Together Control Center als CASE-Tool sowie die schwerpunktmäßige Verwendung von Fallstudien aus dem Logistikbereich.

Dreh- und Angelpunkt für alle Koordinationsaktivitäten der einzelnen Teilprojekte sowie auch der teilprojektübergreifenden vier Koordinationsteams untereinander war aber im Berichtszeitraum die Entwicklung eines LOM-basierten Metadatenkonzeptes. Wie auf den vorgehenden Seiten ausgeführt wurde, beruht dieses Metadatenkonzept nicht nur auf den eigenen Vorarbeiten von KT2, sondern auch auf den Aktivitäten der anderen Teams KT1 und KT3. Zudem bildete es die Basis für die Entwicklung des MuSofT-Portals durch KT4.

Literatur

[Adv02] ADVANCED DISTRIBUTED LEARNING: *SCORM – Sharable Content Object Reference Model*. `http://www.adlnet.org`, 2002.

[DFC+01] DUVAL, ERIL, EDDY FORTE, KRIS CARDINAELS, BART VERHOEVEN, RAFAEL VAN DURM, KOEN HENDRIKX, MARIA WENTLAND FORTE, NORBERT EBEL, MACIEJ MACOWICZ, KEN WARKENTYPE und FLORENCE HAENNI: *The ARIADNE Knowledge Pool System*. Communications of the ACM, 44(5):72–78, Mai 2001.

[IEE02] IEEE LEARNING TECHNOLOGY STANDARDS COMMITTEE, IEEE, 3 Park Avenue New York, NY 10016-5997, USA: *Final Draft of the IEEE Standard for Lerning Objects and Metadata*, Juni 2002. Online erhältlich unter `http://ltsc.ieee.org/wg12/`.

[IMS02] IMS GLOBAL LEARNING CONSORTIUM, INC.: *Product Directory*. `http://www.imsproject.org/direct/getproducts.cfm`, 2002.

KT 4 – Integrationsplattform

Klaus Alfert, Ernst-Erich Doberkat, Gregor Engels,
Jan Hendrik Hausmann, Corina Kopka, Marc Lohmann,
Jörg Pleumann, Annika Wagner

Dieser Bericht fasst die Arbeit des Koordinationsteams 4 (KT4) im Rahmen des Projektes MuSofT (Multimedia in der Softwaretechnik) zusammen. Aufgabe von KT4 war die Realisierung einer technischen Plattform zur Sicherung der Nachhaltigkeit der Projektergebnisse. Diese Aufgabe hat KT4 dahingehend gelöst, ein webbasiertes Portal zu entwickeln, mit dem die Lerneinheiten, die in MuSofT entwickelt werden, zentral gesammelt und der Öffentlichkeit zugänglich gemacht werden können.

Inhaltsverzeichnis

1 Einleitung

Das Projekt Multimedia in der Softwaretechnik (MuSofT) ist ein verteiltes Hochschulprojekt zur Erstellung multimedialer Lehrmaterialien und wird im Rahmen des Programms Neue Medien in der Bildung (NMB) gefördert. An MuSofT sind 8 Hochschullehrer aus 7 verschiedenen deutschen Hochschulen beteiligt. Eine solche verteilte Entwicklung macht eine Koordination

zwischen den Projektpartnern unumgänglich, um ein konsistentes und erfolgreiches Projektergebnis zu erreichen. Zu diesem Zweck wurden Koordinationsteams gebildet, die übergreifende Aufgaben im Projekt planen und realisieren sollen. In diesem Bericht werden Aufgaben und Arbeiten des Koordinationsteams 4 in den Jahren 2001 bis 2003 zusammengefasst.

2 Aufgabe des Koordinationsteams

Aufgabe des Koordinationsteams 4 (im Folgenden kurz KT4) war laut Projektbeschreibung [DDE02] die Erstellung einer gemeinsamen multimedialen Integrationsplattform. Die erste Aufgabe des KTs war es nun, diesen Begriff schärfer zu fassen. Aufgrund zweier schriftlicher Befragungen der Projektpartner und begleitenden Diskussionen kristallisierte sich heraus, dass die zentrale Aufgabe dieser Plattform die *Sicherung der Nachhaltigkeit* der erzielten Projektergebnisse sein sollte. Konkret bedeutet dies, dass die Lehr-/Lerneinheiten, die im Projekt MuSofT produziert werden, sowohl während der Projektlaufzeit insbesondere aber auch darüber hinaus einen Einsatz in der Hochschullehre erfahren sollen.

Um dies zu erreichen, hat das KT4 die Einrichtung und den Betrieb eines zentralen webbasierten Portales vorgesehen, über das alle Ergebnisse von MuSoft öffentlich zugänglich sind. Zur Verwirklichung des Ziels der Nachhaltigkeit sind dabei eine Reihe von Faktoren zu berücksichtigen: Es muss ein *Nutzungsszenario* entwickelt werden, das darstellt, wer, wie und mit welchen Anforderungen die Projektergebnisse nutzen will und mit welchen Funktionen das Portal dies unterstützt (siehe Abschnitt 3.1). Eine zentrale Fragestellung ist dabei die Definition *technischer Schnittstellen* zu Systemen mit denen die Lerneinheiten erstellt, verwaltet bzw. eingesetzt werden (siehe Abschnitte 3.4 bzw. 3.5). Eine weitere Herausforderung ist es, die Lerneinheiten so zu verwalten, dass ein potentieller Nutzer schnell und bequem *Zugriff* auf die seinen Bedürfnissen entsprechenden Einheiten bekommt. Dies berührt den Bereich der Metadaten für Lerneinheiten, die von KT2 definiert wurden und für die wir eine technische Umsetzung entwickelt haben (siehe Abschnitt 3.2). Hierbei waren vor allem sinnvolle Vorgaben für die Beschreibung der einzelnen *Themengebiete* zu entwickeln, um dem Benutzer eine gezielte Suche nach bzw. eine Navigation über Materialien im Portal zu ermöglichen; Abschnitt 3.3 stellt die Arbeiten zu diesem Thema dar. Bei der Betrachtung der langfristigen Nutzung des Portales muss ein *Betriebskonzept* entwickelt werden, dass den Bestand sichert, gleichzeitig aber auch durch Aktualisierungen und Angebotserweiterungen die Attraktivität und damit die Nutzung sichert. Konzepte hierzu finden sich in 3.6. Diese Konzepte werden realisiert in der Implementierung des Portals, das zunächst projektintern eingesetzt wurde und seit dem Winter 2003 auch der Öffentlichkeit zugänglich ist. Details zur technischen Umsetzung dieses Portals finden sich in Abschnitt 3.7.

3 Aktivitäten und Ergebnisse

3.1 Nutzungsszenario

Das Nutzungsszenario des Portals legt die grundlegenden Anforderungen an das Portal fest [Ple02]. Die Aufgabe des Portals ist es, wie oben bereits dargestellt, die MuSofT-Lehrmaterialien webbasiert zu distributieren. Nutzer des Portals sollen Lehrende der Softwa-

retechnik in Universiäten und Fachhochschulen sein, die für ihre Lehrveranstaltungen Materialien suchen. Neben diesen konsumierenden Nutzern gibt es Autoren, die Materialien in das Portal einstellen. Zu letzteren gehören insbesondere die MuSofT-Projektpartner, eine aktive Beteiligung eines weiteren Personenkreises ist hier jedoch ausdrücklich erwünscht. Zusätzlich fallen Verwaltungsaufgaben innerhalb des Portals an. Wir haben uns daher dafür entschieden, drei Nutzerklassen einzuführen (Konsumenten, Autoren und Administrator). Die Funktionalität des Portals kann nun gestuft durch diese drei Nutzerklassen beschrieben werden.

Konsumenten können Materialien im Portal recherchieren, dabei stehen ihnen verschiedene Recherchemöglichkeiten zur Verfügung: Geordnet nach Themengebieten, nach Autoren oder nach LOM-Hierarchiestufen (siehe Abschnitt 3.2) sowie die direkte Suche. Lerneinheiten können dann als Kombination aus inhaltlichen Mediendateien und beschreibenden Metadaten aus dem Portal exportiert und lokal beim Konsumenten genutzt werden. Das standardisierte Exportformat ermöglicht zudem einen Import in eventuell bereits an einer Hochschule vorhandene Lehr-/Lernplattformen (siehe Abschnitt 3.4). Fehlerkorrekturen, Verbesserungsvorschläge etc. können über eine Feedbackfunktion an die Autoren einer Lerneinheiten direkt verschickt werden.

Autoren haben Schreibrechte innerhalb des Portals und müssen sich dafür im Portal authentifizieren. Sie können zusätzlich zu den Funktionaliäten der Konsumenten Materialien in das Portal einstellen und dort modifizieren. Es wird dabei gewährleistet, dass jeder Autor nur seine eigenen Materialien modifizieren darf.

Der Administrator hat zusätzliche Verwaltungstätigkeiten zu erledigen. Hierzu zählen insbesondere die Einrichtung und Verwaltung von Nutzern sowie die Pflege der Lernobjekte inklusive des Löschens/Berichtigens von Daten, die fehlerhaft im Portal gespeichert sind.

3.2 Adaption des LOM-Standards

Im Rahmen von KT4 ist nach einer XML basierten Implementation für die von KT2 vorgegebenen Metadaten zur Beschreibung von Lernobjekten gesucht worden. Die Vorgaben von KT2 basieren auf dem LOM Standard. Der IEEE LOM Standard definiert eine Menge von Metadatenelementen zur Beschreibung von Lernobjekten. Damit soll eine konsistente Verwendung von Metadaten als Basis für unterschiedliche Implementierungen festgelegt werden. Eine Beschreibung wie bzw. in welchem Datenformat diese Metadaten zwischen verschiedenen Systemen ausgetauscht werden können wurde vom IEEE jedoch nicht festgelegt. Die IMS (IMS Global Learning Consortium, Inc.) hat hierzu ein XML-Format mit Hilfe von DTDs bzw. XML-Schemas festgelegt.

Ein Grund für die Verwendung der IMS-Spezifikation ist, dass diese bereits von unterschiedlichen Lern-/Lehrumgebungen unterstützt wird. Weiterhin ist die IMS-Spezifikation auch innerhalb des SCORM-Standards, der verschiedene Standards bzw. Spezifikationen aus dem Bereich eLearning zu integrieren versucht, mit geringfügigen Erweiterungen aufgenommen worden, wodurch SCORM-konforme Metadaten erzeugt werden können.

Das vom IMS vorgegebene XML Schema kann nahezu unverändert übernommen werden [Loh02b]. Grundsätzlich stellen die Vorgaben von KT2 nur eine Teilmenge der IMS-Vorgaben dar, d.h. Musoft-Instanzen des IMS Schemas enthalten nicht alle vom IMS vorgegebenen Elemente. Zwei Einschränkungen sind jedoch zu beachten: Erstens kann das von KT2 vorgegebene Attribut „Lernziele“ nicht ohne Änderungen des IMS XML Schemas verwendet werden,

da dieses Attribut nicht dem LOM-Standard entstand. Zweitens sind die von KT2 für einige Attribute festgelegten Vorgabewerte nicht Teil der IMS-Spezifikation.

3.3 Taxonomie

Um gezielt nach bestimmten Lehreinheiten zu einem Fachgebiet suchen zu können, ist es notwendig, den Inhalt der Einheit gemäß eines vorgegebenen Kataloges (einer *Taxonomie*) zu klassifizieren. So können Einheiten mit ähnlichen/zusammengehörenden Themen erkannt werden, der Suchraum wird auf sinnvolle Begriffe eingeschränkt und eine Navigation über thematische Zusammenhänge wird ermöglicht.

Als Grundlage für einen solchen Themenkatalog stellt die ACM seit 1964 eine Klassifikation zur Verfügung, die das Fachgebiet Informatik über mehrere Ebenen in einzelne Themengebiete aufteilt. Die aktuelle Version stammt aus dem Jahr 1998 und ist abrufbar unter `www.acm.org/class`. Da die ACM-Klassifikation Standardcharakter besitzt und international Anwendung findet, bietet eine Orientierung daran die größtmöglichen Chancen, eine vom Benutzer akzeptierte Einordnung in Themen festzulegen.

Auf Basis dieser Klassifikation haben wir die für MuSofT relevanten Gebiete und ihre Unterteilungen festgelegt:

- D. Software
- H. Information Systems
- K. Computing Milieux

Details dazu sowie eine erst Zuordnung von MusofT-Lerneinheiten zu den einzelnen Gebieten finden sich in [Hau02a]. Der für MuSofT relevante Teil der ACM-Klassifikation ist bereits im Portal eingefügt, eine Erweiterung auf andere Felder der ACM-Taxonomie bzw. eventuelle MusofT-interne Themenbezeichnungen ist aufgrund der dynamischen Struktur des Portals jederzeit möglich.

3.4 Export von Lehrmaterial

Das Musoft-Portal soll zur Verwaltung und Verbreitung von Lehreinheiten verwendet werden, die im Rahmen des Projektes Musoft erstellt werden. Um den nachhaltigen Einsatz der entwickelten Lehreinheiten aus Sicht des Portals zu ermöglichen, muss unter anderem sichergestellt werden, dass die darin abgelegten Lehreinheiten in bestehende Lehr-/Lernplattformen integriert werden können. Zu diesem Zweck muss das Musoft-Portal ein entsprechendes (standardisiertes) Exportformat anbieten. Gleichzeitig muss dieses Format den LOM-Standard, bzw. die XML-basierte Implementierung des LOM-Standards, unterstützen, damit auch die existierenden Metadaten zur Beschreibung der Lernobjekte in anderen Lehr-/Lernplattformen verwendet werden können.

Hier kommen grundsätzlich zwei standardisierte Datenaustauschformate in Frage [Loh02a]: Zum einen das IMS Content Packaging und zum anderen die entsprechenden Teile des Sharable Content Object Reference Model (SCORM) von ADL. Die IMS Content Packaging Spezifikation beschreibt Datenstrukturen, um den Austausch von Inhalten zwischen Lehr-/Lernplattformen und auch entsprechenden Autorensystemen zu standardisieren. So können

Systeme Pakete mit Lehr-/Lerneinheiten erstellen, die von anderen unabhängig entwickelten Systemen aufgrund der standardisierten Struktur der Pakete eingelesen werden können. IMS Content Packaging wird bereits von unterschiedlichen Plattformen (z.B. Blackboard, Centra, CourseKeeper) unterstützt. Jedoch ist hier kritisch anzumerken, dass die meisten Plattformen diesen Standard (noch) nicht vollständig bzw. nicht korrekt unterstützen [Boy02, Wil02].

Der SCORM-Standard beinhaltet einen Content Packaging Teil, der im wesentlichen identisch mit der IMS Spezifikation ist, und bietet darüber hinausgehend weitere Möglichkeiten, die Reihenfolge der Darstellung von Inhalten in Abhängigkeit von den Erfahrungen des Studierenden zu beeinflussen. Konkrete Aussagen, in wie weit die unterschiedlichen Plattformen den SCORM-Standard unterstützen, lassen sich noch nicht machen, da entsprechende Untersuchungen nicht verfügbar sind. Jedoch ist aufgrund der hohen Komplexität der Spezifikation davon auszugehen, dass auch dieser Standard nicht korrekt und vollständig implementiert ist.

Aufgrund der Verbreitung und der Beteiligung der wichtigsten Hersteller von eLearning-Plattformen an der Entwicklung der IMS-Spezifikationen, schien der IMS Content Packaging Standard am geeignetsten zu sein, um mittelfristig die Verwendung von Lernmaterialien in kommerziell erhältlichen und weit verbreiteten Lernplattformen zu ermöglichen. Das Musoft-Portal sieht daher einen Export von Lernmaterialien entsprechend der IMS Spezifikationen wie folgt vor: Beim Herunterladen eines oder mehrer Lernobjekte, werden diese zu einer IMS-konformen ZIP-Datei zusammengefaßt. Die Struktur komplexer Lernobjekte bleibt innerhalb der ZIP-Datei erhalten. Ebenso finden sich in der Datei alle Metadaten der exportierten Lernobjekte wieder. Schließlich wird noch Sorge dafür getragen, daß sämtliche Lizenzen, denen die Nutzung des exportierten Materials unterworfen ist (für das projekteigene Material in den meisten Fällen die MuSofT-Lizenz), in die ZIP-Datei aufgenommen werden.

Eine weitere zu beachtende technische Hürde beim Export von Material kann das Speicherformat multimedialer Medienobjekte sein. Den IMS Standard berührt dies nicht. Wir haben daher in Abstimmung mit den Projektpartnern eine verbindliche Liste verbreiteter Formate für Animationen, Filme, Dokumente etc. erstellt, die auf eine weitestgehende Verbreitung bei den Lehrenden hin optimiert ist [PA02].

3.5 Import von Lehrmaterial

Um Lerneinheiten in das Portal einzustellen gibt es grundsätzlich zwei Szenarien:

1. Der Benutzer hat eine Menge von Medienobjekten, die eine Lehreinheit bilden. Zur Veröffentlichung dieser Lehreinheit erlaubt ihm das Portal den Upload über einen beliebigen Web-Browser und die anschließende Annotation der Medienobjekte. Mit Hilfe des Portals können die einzelnen Medienobjekte dann zu einer Lehreinheit zusammengesetzt werden.

2. Der Benutzer setzt bereits eine Lernplattform ein, die es ihm erlaubt, Lernobjekte mit Metadaten zu annotieren, oder die Metadaten liegen aus anderen Gründen schon in digitaler Form vor. Wenn dies der Fall ist, dann ist ein manuelles Einspielen von Lehrmaterial und dessen Annotation mit Metadaten über die Web-Schnittstelle unnötig aufwendig.

Um den zweiten Fall angemessen zu unterstützen, wurde innerhalb des abschließenden Projektjahres der MuSofT Upload Wizard entwickelt, eine Client-Software, die das Einspie-

len von bereits mit Metadaten annotierten Lernobjekten in das MuSofT-Portal erleichtert. Der Wizard existiert in zwei Geschmacksrichtungen: Eine einfach zu bedienende Variante mit graphischer Oberfläche erlaubt das interaktive Setzen der benötigten Einstellungen und das Beobachten des Übertragungsvorgangs. Eine textuelle Variante ermöglicht den automatisierten Upload von Lehrmaterial über die Kommandozeile oder Skriptsprachen. Beide Varianten basieren auf dem gleichen Back-End und verwenden identische, XML-basierte Eingabedateien, deren Format in [Ple03] beschrieben ist.

3.6 Betriebskonzept

Um mit dem MuSoft-Portal nachhaltig die im Projekt erstellten Lehreinheiten verbreiten zu können ist ein Betriebskonzept erforderlich, das den erfolgreichen Betrieb auch nach Ende der Projektlaufzeit (12/2003) sicherstellt. Dies betrifft zum einen eine Sicherung des Bestandes. Hier müssen z.B. Pläne zum Betrieb und der Betreuung des Webservers festgelegt werden. Detaillierte Rahmenbedingungen, die innerhalb eines derartigen Betriebskonzeptes beachtet werden müssen, wurden innerhalb von KT4 diskutiert. Ein vorläufiges Betriebskonzept findet sich in [Hau02b]. Dieses Betriebskonzeptes kann während der Nutzungsdauer des Portals regelmäßig überprüft und gegebenenfalls an geänderte Anforderungen oder Feedback der Nutzer angepaßt werden.

Daneben gilt es aber auch, das Portal attraktiv zu machen und zu erhalten. Es gibt Anregungen zum Marketing für den Server, die in der Diskussion zwischen den Projektpartnern konkretisiert werden müssen. Auch das von KT4 entwickelte neue Layout des Portals ist ein Schritt zur Erreichung dieser Attraktivität. Die Aktualisierung, Pflege und Ausweitung der angebotenen Lehreinheiten haben eine zentrale Bedeutung für den nachhaltigen Erfolg von MuSoft. KT4 hat hierzu Konzepte entwickelt und richtet die Bedienung des Portals darauf aus, diese Aufgaben möglichst einfach und bequem durchführen zu können. Ein entsprechender User Guide steht im Portal selbst zur Verfügung.

3.7 Implementierung

Das MuSofT-Portal ist als webbasiertes System realisiert (siehe Abbildung 1) und unter der Adresse `http://www.musoft.org` erreichbar. Es unterstützt die Distribution der im Rahmen von MuSofT erstellten Lehreinheiten und den dazugehörigen Werkzeugen. In diesem Abschnitt beschreiben wir die technische Realisierung des Portals und den Einsatz der Metadaten.

3.7.1 Technische Realisierung

Das MuSofT-Portal kann man als eine spezielle Variante eines Content-Management-Systems (CMS) auffassen, das als Inhalte die multimedialen Lernobjekte zusammen mit Metadaten für die Recherche verwaltet. Kommerziell verfügbare CMS haben allerdings unsere Anforderungen nicht erfüllt. Als besonders problematisch haben sich dabei vier Punkte herausgestellt:

1. Klassischerweise unterscheiden CMS zwischen einer Entwicklungssicht mit und einer Präsentationssicht ohne (ausgefeilte) Recherchemöglichkeiten. Dies widerspricht aber

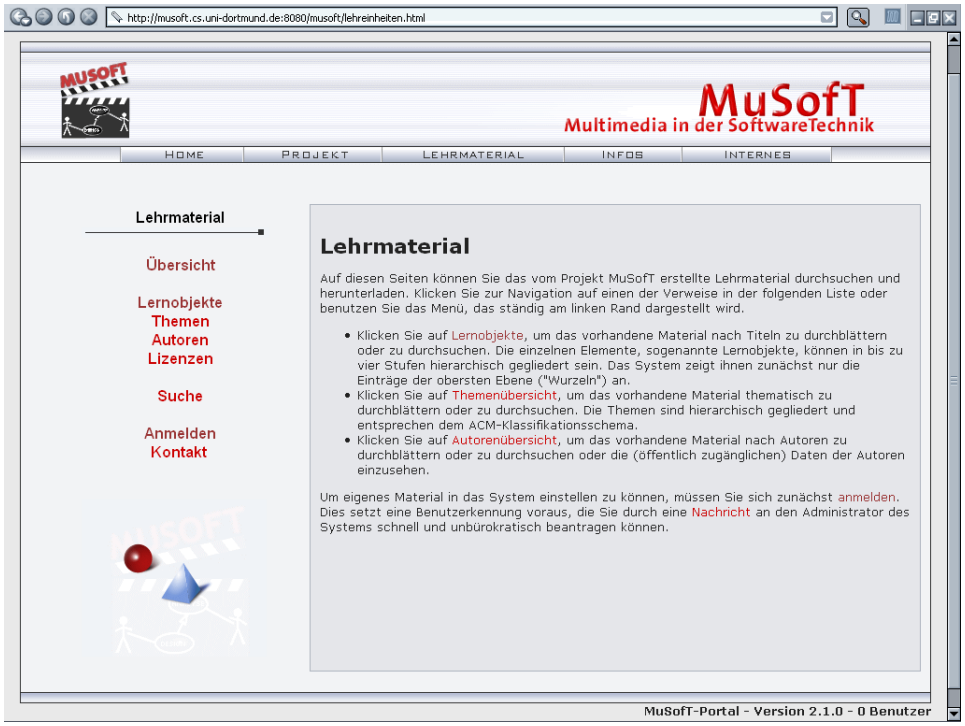


Abbildung 1: Startseite für Recherchen im MuSofT-Portal

der Arbeitsweise im MuSofT-Portal, da es dort nur eine (Präsentations-) Sicht gibt, in der insbesondere recherchiert werden soll.

2. Wir benötigen nicht nur eine fest vom System vorgegebene Menge von Metaattributen -wie dies bei CMS oft der Fall ist-, sondern zusätzlich strukturierte Metaattribute (z.B. für die Klassifikationshierarchie).

3. Da wir erwarten, dass der Satz an Metaattributen und ihrer Wertemengen sich aufgrund unserer künftigen Erfahrungen mit dem Portal ändern wird, brauchen wir eine leichte Änderbarkeit der Metaattribute und ihrer Ausprägungen im laufendem Betrieb.

4. Schlussendlich sind für eine nutzerfreundliche Bedienung spezielle Suchanfragen entlang der Hierarchie der Lernobjekte und der Hierarchie des Klassifikationsschemas nötig.

Wir haben uns daher für eine Eigenentwicklung auf der Basis einer existierenden objektorientierten Datenbanklösung entschieden, die es uns ermöglicht, ein flexibles webbasiertes Portal zu erstellen.

Serverseitig basiert das MuSofT-Portal auf dem *Infolayer-System* [HP02], das als Servlet realisiert wurde. Das Infolayer-System ist eine objektorientierte Datenbank, deren Schema mit

Abbildung 2: Metadaten eines Lernobjektes im MuSofT-Portal

durch OCL annotierten UML-Klassendiagrammen spezifiert wird. Als Abfragesprache wird ebenfalls OCL verwendet. Die Daten werden als XML-Dateien abgespeichert. Eine Standardweboberfläche zur Navigation durch das Schema und durch die vorhandenen Objektinstanzen wird vom Infolayer-System automatisch zur Verfügung gestellt und kann von jedem (neueren) HTML-Browser aus bedient werden. Diese Oberfläche kann sukzessive durch Schablonen an eigene Anforderungen angepasst werden, so dass Entwicklungsarbeiten sehr schnell zu bereits produktiven Prototypen führen. Diese Eigenschaften erlauben eine einfache Anpassung der Datenbank und des Portals, wenn sich die Metadaten aufgrund von Erfahrungen beim Einsatz des MuSofT-Portals ändern.

3.7.2 Verwendung von Metadaten

Das MuSofT-Portal dient zur Archivierung von Lernobjekten, die innerhalb von MuSofT erstellt werden. Die wichtigsten Aktivitäten, die mit dem Portal ausgeführt werden können, sind das Einfügen, Aktualisieren und Suchen von Lernobjekten. Beim Einfügen eines neuen Lernobjekts oder beim Modifizieren eines bestehenden Lernobjekts sind die bereits erwähnten Vorgaben für Metadaten zu berücksichtigen, damit innerhalb des MuSofT-Portals eine einheitliche Beschreibung der Lernobjekte erfolgt. Die Oberfläche des Systems unterstützt den

Benutzer dabei, in dem sie Eingabefelder für die erforderlichen Attribute anbietet und für Attribute mit fester Wertemenge eine Auswahlbox verwendet, so dass dort keine ungültigen Werte eingegeben werden können. Abbildung 1 zeigt einen Ausschnitt der Benutzeroberfläche des MuSofT-Portals zur Beschreibung eines Lernobjekts.

Neben dem reinen Hochladen von Lernobjekten unterstützt das MuSofT-Portal die vom LOM-Standard vorgeschriebene Hierarchisierung von Lernobjekten. So kann z.B. eine neue Lehreinheit aus verschiedenen Lernmodulen, die zuvor in dem MuSofT-Portal erstellt wurden, zusammengesetzt werden.

Da Metadaten für Lernobjekte immer einen sehr subjektiven Charakter haben, kann es schwierig sein, passende Lernobjekte zu finden, wenn man sich ausschließlich auf Freitexteingaben für Stichworte und inhaltliche Beschreibungen verlässt. Wir verwenden daher für die inhaltliche Beschreibung zusätzlich eine festgelegte Taxonomie (siehe Abschnitt 3.3. Bei Bedarf kann dieses Klassifikationsschema sowohl um zusätzliche neue Themenbereiche als auch um verfeinerte Klassifikationen erweitert werden. Ebenso können bei Bedarf weitere unabhängige Klassifikationssysteme zur Verfügung gestellt werden. Lernobjekte können mit einem oder mehreren Einträgen aus dem Klassifikationsschema versehen werden, so wie dies bei der Klassifikation von Zeitschriftenartikeln üblich ist. Wir erwarten, dass die inhaltliche Recherche wesentlich über das Klassifikationsschema stattfinden wird.

4 Zusammenfassung

Die Aufgabe der Erstellung einer gemeinsamen multimedialen Plattform für MuSofT hat KT4 dahingehend gelöst, ein webbasiertes Portal zu entwickeln, mit dem die Lerneinheiten, die in MuSofT entwickelt werden, zentral gesammelt und der Öffentlichkeit zugänglich gemacht werden können. Ausgestattet mit komfortablen Recherchemöglichkeiten sowie Im- und Export auf der Basis von etablierten Standards, bietet das MuSofT-Portal einen effektiven Weg zur nachhaltigen Distribution der MuSofT-Lerneinheiten.

Literatur

[Boy02] BOYLE, E.: *CETIS EC-SIG Content Exchange Report*, Januar 2002. Erhältlich auf `http://www.cetis.ac.uk/groups/20010809144711/FR20020301142909`.

[DDE02] DISSMANN, STEFAN, ERNST-ERICH DOBERKAT und GREGOR ENGELS: *Projektantrag MuSofT – Multimedia in der SoftwareTechnik.* Unveröffentlichter Projektantrag für das Programm *Neue Medien in der Bildung* des BMBF, Februar 2002.

[Hau02a] HAUSMANN, JAN HENDRIK: *MuSofT - Klassifikation der Lehr/Lerneinheiten - SE-Taxonomie*. Erhältlich auf `http://www.musoft.org`, erscheint als MuSofT-Bericht, 2002.

[Hau02b] HAUSMANN, JAN HENDRIK: *Nachhaltigkeit der Ergebnisse des MuSofT-Projektes*. Erhältlich auf `http://www.musoft.org`, erscheint als MuSofT-Bericht, 2002.

[HP02] HAUSTEIN, STEFAN und JÖRG PLEUMANN: *Is Participation in the Semantic Web Too Difficult?* In: HORROCKS, I. und J. HENDLER (Herausgeber): *The Semantic Web - First International Semantic Web Conference*, Band 2342 der Reihe *LNCS*, Heidelberg, 2002. Springer.

[Loh02a] LOHMANN, MARC: *MuSofT - Exportformat für Lehreinheiten*. Erhältlich auf `http://www.musoft.org`, erscheint als MuSofT-Bericht, 2002.

[Loh02b] LOHMANN, MARC: *Vergleich Metadaten KT2 - IMS Learning Resource Meta-Data XML Binding*. Erhältlich auf `http://www.musoft.org`, erscheint als MuSofT-Bericht, 2002.

[PA02] PLEUMANN, JÖRG und KLAUS ALFERT: *Unterstütze Formate für MuSofT*. Erhältlich auf `http://www.musoft.org`, erscheint als MuSofT-Bericht, September 2002.

[Ple02] PLEUMANN, JÖRG: *Anwendungsfälle für das MuSofT-Portal*. Erhältlich auf `http://www.musoft.org`, erscheint als MuSofT-Bericht, Juli 2002.

[Ple03] PLEUMANN, JÖRG: *Der MuSofT Upload Wizard*. Erhältlich auf `http://www.musoft.org`, erscheint als MuSofT-Bericht, Oktober 2003.

[Wil02] WILSON, S.: *Content Packaging interoperability tests reveal room for improvement*, April 2002. Erhältlich auf `http://www.cetis.ac.uk/content/20020307103412`.

Der Abschlussbericht für das Koordinationsteam 5: Medienproduktion

Klaus Alfert, Corina Kopka

Inhaltsverzeichnis

1 Einleitung

Ein wesentlicher Teil der finanziellen Ausstattung von MuSofT war gedacht für die Erstellung von Medien, insbesondere für Videoproduktionen wie sie etwa für das Teilprojekt 1.1 (Videogestützte Anforderungsanalyse) vorgesehen sind. Um diese Summe nicht nur in einem einzigen Teilprojekt einzusetzen, sondern möglichst vielen anderen Teilprojekten ebenfalls die Möglichkeit zu geben, entsprechende Medien zu produzieren, ist das Koordinationsteam 5 (Medienproduktion, kurz: KT5) mit dem dritten Plenum Anfang März 2002 ins Leben gerufen worden. Die Aufgabe des KT5 war es, Medienproduktionen, die über Teilprojektgrenzen hinweg möglich ist, zu koordinieren.

Im Folgenden stellen wir die Aufgaben, Aktivitäten und Ergebnisse des Koordinationsteams vor.

2 Aufgabe des Koordinationsteams

Aufgabe des KT5 war es, die Planung, Konzeption und Durchführung von Medienproduktionen zu koordinieren, die teilprojektübergreifend eingesetzt werden können. Wesentlich war dabei, den Bedarf der einzelnen Teilprojekte herauszuarbeiten und Kristallisationspunkte zu finden, an denen die Teilprojekte gut kooperieren können. Auf der Basis dieser Überlegungen konnten die Teilprojekte einzelnen oder gemeinsam ihren Mittelbedarf für die Medienproduktion bei der Projektleitung anmelden und abrechnen. Die Mittel für diese Medienproduktionen stehen hälftig in 2002 und 2003 zur Verfügung.

3 Aktivitäten

Die Aktivitäten des KT5 lassen sich in die Bereiche Bedarfsermittlung, Planung und Medienproduktionen teilen.

3.1 Bedarfsermittlung

Zur Ermittlung des Bedarfes an Medienproduktionen haben zwei Treffen des KT5 stattgefunden, am 20. April 2002 in Dortmund und am 7. Juni 2002 in Paderborn.

Das erste Treffen beschäftigte sich im wesentlichen mit Fragen zu Videoproduktionen an sich. Als unterschiedliche Ausrichtungen von Videos wurden drei Kategorien herausgearbeitet:

Melodram: zeigt Fehlverhalten und die daraus resultierenden Konsequenzen,

Instruktion: zeigt beispielhaftes Vorgehen,

Aufgabenstellung: zeigt eine Situation, die die Basis für eine (Übungs-) Aufgabe darstellt.

Die Kategorie *Melodram* ist nur schlecht einsetzbar, weil der Aufwand an schauspielerischer Konzeption und Leistung als entschieden zu hoch angesehen wird. *Instruktionsvideos* eignen sich gut für Werkzeugschulungen und scheiden damit zwar als teilprojektübergreifende Medienproduktionen aus, sind aber für einzelnen Teilvorhaben sinnvoll. Die dritte Kategorie *Aufgabenstellung* ist daher die favorisierte Variante der Medienproduktion, die teilprojektübergreifend am sinnvollsten eingesetzt werden kann.

Als interessantes Anwendungsgebiet für die Medienproduktion hat sich die Logistik herausgestellt, da hier viele für MuSofT interessante Probleme diskutiert werden können. Zusätzlich sind bereits einige Vorarbeiten in den Teilvorhaben verfügbar (z.B. Hochregallagersteuerung in TP 1.3, Algorithmen für Speditionen in TP 2.3).

Beim zweiten Treffen in Paderborn wurde daher der Themenkomplex Logistik mit dem Schwerpunkt Hochregallager intensiv diskutiert. Produktvideos von Mannesmann-Demag für Hochregallager haben einen ersten Eindruck gegeben, wie ein derartiges technisches System präsentiert werden kann. Dabei wurde deutlich, dass diese Produktvideos zwar für einzelne Aspekte etwa zur Modellierung des Systems direkt sinnvoll einsetzbar sind, aber für viele

weitere Fragestellungen sich zu sehr auf die maschinenbautechnischen Fragestellungen konzentrieren und softwaretechnikrelevante Aspekte nicht thematisieren. Daher wurde beschlossen, dass eine eigene Videoproduktion angestrebt werden soll.

3.2 Planung und Medienproduktion

Die teilprojektübergreifende Medienproduktion zum Thema Logistik spaltete sich von Beginn an in verschiedene Teilvorhaben auf. Für die Teilprojekte 1.1, 1.3 und 3.3 wurden Abläufe aus dem Lagerwesen und der Bestellabwicklung betrachtet. Für weitere Teilprojekte (z.B. 1.2, 2.1, 3.3) war der gewünschte Inhalt noch unklar, der sich aber auch im Umfeld des Lagerwesens und der Bestellabwicklung bewegen sollte. Für diese beiden eng miteinanderverwandten Bereiche sollte daher ein Video produziert werden. Für das Teilprojekt 2.3 wurden dagegen explizit Problemstellungen aus dem Speditionswesen betrachtet, die damit unabhängig von den Bereichen Lagerwesen und Bestellabwicklung sind. Daher wurde hier auch keine Videoproduktion realisiert.

Im Vorfeld des zweiten KT-Treffens in Paderborn wurden bereits erste Kontakte zur Logistiksparte von Bertelsmann, der arvato services GmbH in Gütersloh, geknüpft und ein Besuch des Hochregallagers in Gütersloh hat stattgefunden. Bei der weitergehenden Planung wurde der dringende Bedarf an Unterstützung und Einweisung in das Drehbuchschreiben sowie weiterer Vorarbeiten zur Vorbereitung eines Videodrehs durch ein professionelles Team festegestellt. Daher wurde ab Mitte November die Industrie-Designerin Bettina Neu von Siemens Business Services als Beraterin hinzugezogen. Zur Vorbereitung der Drehbucherstellung hat dann ein weiterer Besuch unter der Leitung von Herrn Bätge (Leitung Wareneingang) bei arvato in Gütersloh stattgefunden, welcher mit Hilfe von Fotos und Videos dokumentiert wurde, die für Skizzen innerhalb eines Drehbuches und zur Festlegung von Drehorten benutzt werden konnte. Desweiteren hatte Herr Bätge zugesagt, dass MuSofT auf das firmeneigene Bild- und Videomaterial von arvato selbst zugreifen darf, so dass auf diese Weise der Bedarf an neuen Videoproduktionen gemindert werden kann.

Zu Beginn des Jahres 2003 fanden am 18. und 19. Januar zwei Workshops zu den Themen *Drehbuch schreiben* und *Gestaltung* statt. Beide Workshops dienten dazu, die mediale Konzeption der einzelnen Teilprojekte zu überarbeiten bzw. in eine Form zu bringen, die es erlaubt, mit externen Medienproduzenten effektiv zusammen zu arbeiten. Beide Workshops wurden von Bettina Neu geleitet. Speziell für den Drehbuchworkshop wurde das Rohmaterial des Besuchs bei arvato gesichtet, damit als Nebeneffekt zugleich der Bedarf an einer MuSofT-eigenen Videoproduktion genauer eingeschätzt werden konnte.

Ein wesentliches Ergebnis der beiden Workshops war die Erkenntnis, dass sich der Bereich Logistik als Videodarstellung für die Teilprojekte 1.1, 1.2, 1.3 und 3.3 anbietet, während für die anderen Projekte sich dagegen andere Beispielszenarien als adäquater herausgestellt haben.

Nach Kontaktaufnahme mit der Medienproduktionsfirma mediaprojekt aus Verl, die bereits Videos für arvato gedreht hatte, und einer ersten Sichtung von existierenden Material, wurden dann bei einem Treffen im Mai in Paderborn für die Teilprojekte 1.1, 1.2 und 1.3 der genaue Bedarf der Logistikvideos fixiert. Auf der Basis dieses Bedarfs wurden dann im Folgenden Drehbücher geschrieben, die sowohl mit mediaprojekt als auch mit arvato abgestimmt wurden. Ab Herbst 2003 begann die eigentliche Medienproduktion durch mediaprojekt, die

neben der Wiederverwendung bereits existierenden Materials auch Neuverfilmungen in den Räumlichkeiten von arvato eingeschlossen.

Für das Teilprojekt 3.3 wurde im Sommer 2003 ein Video geplant, das einen typischen Dialog zwischen einer Softwareentwicklerin und einem Auftraggeber über Anforderungen der zu erstellenden Software darstellt. Dafür wurde ein Drehbuch geschrieben und zwei Schauspieler engagiert, die an dem Drehbuch mitgearbeitet haben und mit denen dieser Dialog gedreht wurde. Um den gedrehten Film aufzulockern, wurden einige Szenen aus einem Video des Pharmagroßhandels Pharmlog eingeblendet, die das im Dialog Gesagte bildlich untermauern. Für diese Produktion wurde mit dem Medienzentrum der Universität Dortmund zusammengearbeitet, die für den Videodreh und die nachträgliche Bearbeitung (Schnitt, Einblendungen, Vorspann, Nachspann) zuständig waren.

4 Ergebnisse

Neben der Vorbereitung und Produktion der Videodrehs zur Logistik im Umfeld von arvato sind vier weitere Medienproduktionen abgeschlossen worden: das Video über Anforderungsanalyse im Pharmagroßhandel, das CVS-Video, die Algorithmenanimationen und letzlich Einführungsvideos für die Teilprojekte 3.3 und 3.4.

4.1 Logistik-Videos

Insgesamt 9 verschiedene Videos zur Logistik wurden in Kooperation mit arvato realisiert. Sie dienen unterschiedlichsten Zwecken:

- Einführung in den Gesamtkomplex Lagerwesen und Kommissionierung
- Beschreibung von Tätigkeiten im Umfeld des Hochregallagers
- Darstellung der Datenverwaltungssicht
- Spielszenen zur Illustration von Zielkonflikten bei Optimierungsversuchen

Die verschiedenen Videos sind für die Bedürfnisse der Teilprojekte 1.1, 1.2 und 1.3 zugeschnitten, lassen sich aber auch in davon unabhängigen Lehrsituationen einsetzen, in denen Logistik als Anwendungsbereich der Softwaretechnik beispiel herangezogen wird.

4.2 Pharmagroßhandel-Video

Für das Teilprojekt 3.3 ist ein typisches Gespräch in der Anforderungsanalyse zwischen einer Softwareentwicklerin und einem Kunden im Pharmagroßhandel, für den eine Software zur Bestell- und Lagerverwaltung entwickelt werden soll, als Video erstellt worden. Entstanden ist ein Video von etwa sieben Minuten Länge, der die Schwierigkeiten beim Erheben der Anforderungen darstellt (nicht erklärte Begriffe, unvollständige Anforderungen, usw.). Das Video wurde vom Medienzentrum der Universität Dortmund in Zusammenarbeit mit zwei Schauspielern gedreht.

4.3 CVS Video

Im Rahmen des Teilprojektes 3.4 ist das Instruktionsvideo zur Benutzung des Versions- und Konfigurationsmanagementwerkzeuges CVS entstanden. Es zeigt die Benutzung eines CVS-Clients und illustriert durch Animationen die Vorgänge im CVS-Repository, die durch den Client initiiert werden.

4.4 Algorithmenanimationen

Eine weitere Produktion animiert Algorithmen und Datenstrukturen für das Teilprojekt 2.3 mit dem Anwendungsgebiet Speditionswesen. Neben der Vorführung vorprogrammierter Animationen und Algorithmen wird eine Programmierschnittstelle angeboten, so dass auch Studierenden die Animationen nutzen können, um ihre eigene Algorithmen visualisieren zu können.

4.5 Einführungsvideos

Für die Teilprojekte 3.1 und 3.2 sind kurze Einführungsvideos für dazugehörige Lehrveranstaltung entstanden, die sowohl in die Benutzung von Werkzeugen als auch in die Aufgabenstellung der praktischen Übungen einführen.

5 Zusammenfassung

In diesem Bericht wurden die wesentlichen Aktvitäten und Ergebnisse des Koordinationsteams 5 vorgestellt, dessen Aufgabe die Koordination von teilprojektübergreifenden Medienproduktionen war.

Der MuSofT-Abschlussbericht des Koordinationsteams 6: Nachhaltigkeit

Klaus Alfert

Inhaltsverzeichnis

1 Einleitung

Das Koordinationsteam 6 (Nachhaltigkeit) wurde auf dem dritten Plenum Anfang März 2002 ins Leben gerufen. Die Aufgabe des KT ist es, die Grundlagen für die nachhaltige Wirkung des MuSofT-Projektes zur Verfügung zu stellen.

Die Nachhaltigkeit von MuSofT wird durch drei verschiedene Faktoren wesentlich beeinflusst. Zum einen müssen produzierten Materialien eine Qualität besitzen, die eine Nutzbarkeit über das Projektende und über die Projektmitglieder hinaus überhaupt erst ermöglichen. Dies liegt primär in der Verantwortung der jeweiligen Teilprojektleiter. Zum zweiten setzt die Nutzung von Materialien, die fremde Autoren erstellt haben – unabhängig davon ob sie zu MuSofT gehören oder nicht – voraus, dass die Nutzungsrechte von den Rechteinhabern eingeholt wurden und somit einer Nutzung der fremden Materialien nichts entgegensteht. Der dritte Faktor betrifft die Wartung und Weiterentwicklung der Materialien nach dem Ende von MuSofT mit Ende des Jahres 2003. Hier ist notwendig sicherzustellen, dass die Finanzierung und Personalsituation den Aufgaben entsprechend gesichert ist. Dies kann z.B. durch Nachfolgeprojekte gewährleistet werden.

Erst wenn diese Faktoren zufriedenstellend beachtet worden sind, ist ein nachhaltiger Einsatz der MuSofT-Materialien sowohl in der Hochschullehre als auch in der kommerziellen Weiterbildung sinnvoll und möglich. Die vordringliche Aufgabe und der Auftrag des KT6 beziehen sich auf die Regelung der rechtlichen Rahmenbedingungen für MuSofT, da diese als das wesentliche teilprojektübergreifende Problem angesehen werden.

2 Aufgabe des Koordinationsteams

Die Aufgabe des Koordinationsteams war die Regelung der rechtlichen Rahmenbedingungen für MuSofT, um so sicher zu stellen, dass inhaltlichen Ergebnisse von MuSofT nachhaltig genutzt werden können. Dies schließt insbesondere die Frage nach Lizenzierung der entwickelten Materialien ein.

3 Aktivitäten

Vorbereitend und zur Gründung des KT6 führend war der Besuch von Klaus Alfert auf dem Workshop „Recht einfach. Rechtemanagement für Multimediaprojekte“, der vom Kompetenznetzwerk Universitätsverbund MultiMedia Nordrhein-Westfalen (UVM) im Auftrages des Projektträger veranstaltet [UVM01] wurde. Die für diese Veranstaltung erstellte Broschüre zum Rechtemanagement [Ved01] wurde für alle Projektpartner beschafft und zum Jahreswechsel 2001/2002 verteilt. Auf dem dritten Plenum hat Christiane Dusch vom UVM einen Vortrag zum Rechtemanagement gehalten [Dus02]. Aus den nachfolgenden Diskussionen hat sich die Notwendigkeit ergeben, das Rechtemanagement in MuSofT explizit zu betrachten

Bei der Betrachtung der Rechteproblematik für MuSofT stellt sich heraus, dass der Kooperationsvertrag zwischen den Projektpartnern einige wesentliche Fragen nicht regelt bzw. dem Geist des Projektantrages widersprechen, z.B.:

- Die Nutzung der Materialien nach Ende des Projektes ist nicht geregelt, insbesondere die kommerzielle Nutzung.
- Die Überlassung der Materialien auf Quellcode-Ebene inkl. dem Änderungsrecht (also als *Open Content*, dies entspricht im Wesentlichen dem Begriff *Open Source* aus dem Projektantrag), wird im Kooperationsvertrag im Gegensatz zum Projektantrag nicht zugelassen.

Daher ist es notwendig, den alten Kooperationsvertrag, der der Mustervertrag für Projektkooperationen des BMBF ist, in angemessener Weise für die Belange von MuSofT zu überarbeiten. Auf der Basis eines überarbeiteten Vertrages kann dann die weitere Gestaltung der rechtlichen Situation in MuSofT erfolgen.

Da die Vertragsausarbeitung juristische Kenntnisse voraussetzt, ist dies keine Tätigkeit, die ein Projektpartner selbst übernehmen kann. Daher wurde ein Auftrag an die Juristen des UVM erteilt, einen Vorschlag für einen Kooperationsvertrag zu entwickeln. Ein zweiter Auftrag an den UVM sieht vor, dass die Justitiare der Partnerhochschulen sensibilisiert und geschult werden sollen, die Rechteproblematik in Multimediaprojekten zu erkennen und zu bearbeiten. Die Einbindung der Justitiare ist notwendig, da die Hochschulen auch immer Rechteinhaber werden, wenn Multimediaproduktionen an Hochschulen durchgeführt werden. Für die weitere Verwertung ist daher immer eine Kooperation mit den Hochschulen notwendig und erfordert daher entsprechende Kenntnisse in den Verwaltungen.

Auf dem fünften Plenum wurde zusammen mit Andreas Wolfrum vom UVM die Anforderungen an eine MuSofT-spezifische Lizenz diskutiert. Diese Anforderungen wurden dann in Zusammenarbeit des UVM und des Instituts für Rechtsfragen der Open Source Software zu einer Lizenz ausgearbeitet und auf dem sechsten Plenum präsentiert. Auf dem sechsten

Plenum wurde ebenfalls ein Entwurf für die neue Kooperationsvereinbarung diskutiert, die anschließend von den Projektpartnern zusammen mit ihren Hochschulen unterzeichnet werden wird.

4 Ergebnisse und Zusammenfassung

Das wesentliche Ergebnis des KT6 ist die MuSofT-Lizenz, die in Anhang A aufgeführt ist. Die MuSofT-Lizenz ist die erste Open-Content-Lizenz (soweit uns bekannt ist), die für den europäischen Rechtsraum entwickelt worden ist. Sie ist die wesentliche Grundlage, um die Nachhaltigkeit der Projektergebnisse zu sichern.

Literatur

[Dus02] DUSCH, CHRISTIANE: *Rechtemanagement in Multimediaprojekten.* Vortrag auf dem 3. Plenum des Projektes MuSofT, März 2002.

[UVM01] KOMPETENZNETZWERK UNIVERSITÄTSVERBUND MULTIMEDIA NRW (UVM): *Tagungband Recht einfach – Rechtemanagement in Multimediaprojekten für Beteiligte am Bundesprogramm „Neue Medien in er Bildung“*, November 2001.

[Ved01] VEDDERN, MICHAEL: *Update – Ratgeber. Multimediarecht für die Hochschulpraxis*. Ministerium für Wissenschaft und Forschung des Landes Nordrhein-Westfalen, November 2001.

A Lizenz für die nichtkommerzielle Nutzung von Inhalten an Schulen und Hochschulen (MuSofT Lizenz)

Version 1.0, Oktober 2003

Präambel

Ziel der Lizenzierung eines Werkes unter der MuSofT Lizenz ist es, die Verwendung von Inhalten zum Zwecke der nichtkommerziellen Nutzung bei der Forschung und Lehre an Schulen und Hochschulen zu ermöglichen. Die Lizenz richtet sich vornehmlich an diejenigen, die ihre urheberrechtlich geschützten Leistungen zum Zwecke der nichtkommerziellen Nutzung bei der Forschung und Lehre an Schulen und Hochschulen zur Verfügung stellen wollen, ohne dass für einzelne Nutzungen oder Änderungen gesondert Rechte eingeholt werden müssen.

Sie richtet sich aber auch an diejenigen, die ein Werk vervielfältigen, verbreiten oder verändern möchten, welches nach den Bedingungen dieser Lizenz genutzt werden darf.

Durch die MuSofT Lizenz werden dem Lizenznehmer die Nutzungsrechte für alle bekannten Nutzungsarten eingeräumt und auch die Bearbeitung des Werkes in jeder beliebigen Form gestattet. Die ideellen Interessen der Urheber am Werk werden von der Lizenz dabei beachtet, denn es ist eines der Ziele der Lizenz, die kreativen Leistungen der Urheber und anderen Leistungsschutzberechtigten in angemessener Weise anzuerkennen und ihre geistigen Belange zu schützen. Der Urheber soll mit seinem Werk in Verbindung gebracht werden, indem sein Name genannt wird oder - für den Fall, dass das Werk bearbeitet wurde - in der History des Werkes ein Hinweis auf ihn erfolgt.

Ein wesentlicher Zweck dieser Lizenz besteht darin, die weitere Bearbeitung von Werken zu ermöglichen. Texte, Datenbanken, Multimediawerke und sonstige Inhalte entstehen oft durch die Zusammenarbeit einer Vielzahl von Personen, etwa weil das entstehende Werk zu komplex ist, um von einer Person hergestellt zu werden oder weil Aktualisierungsbedarf besteht, den der Ursprungsautor nicht leisten kann oder möchte. Die MuSofT Lizenz bietet ein Modell zur Entwicklung und Verbreitung von Werken durch eine beliebige Zahl von Personen, die nicht organisatorisch verbunden sein müssen. Sie kann aber auch bei jeder anders motivierten Freigabe von Werken verwendet werden.

Um eine freie Bearbeitung durch andere zu gewährleisten, ist es erforderlich, dass neben der rechtlichen Erlaubnis auch die technischen Voraussetzungen für eine Veränderung des Werkes zur Verfügung gestellt werden. Werke, die in digitaler Form vorliegen oder in eine digitale Form überführt werden, müssen daher in einem Dateiformat zugänglich gemacht werden, das technisch ermöglicht, was rechtlich durch diese Lizenz erlaubt wird.

Die MuSofT Lizenz schützt die Lizenzgeber davor, dass Lizenznehmer die Nutzung des Werkes - auch in bearbeiteter Form - nachträglich beschränken können. Dazu dient der „Copyleft“-Effekt, der gewährleistet, dass ein Werk, welches dieser Lizenz unterstellt wurde, sowie alle darauf beruhenden Bearbeitungen nur gemäß den Bestimmungen dieser Lizenz genutzt werden dürfen.

Die MuSofT Lizenz wurde für das Hochschulprojekt „MuSofT - Multimedia in der Software Technik“ entwickelt, das im Rahmen der Ausschreibung „Neue Medien in der Bildung“ des BMBF gefördert wurde, und dort erstmals eingesetzt.

1. Abschluss der Lizenz

(a) Dieser Lizenztext stellt ein Angebot auf Abschluss eines Lizenzvertrages unter den nachfolgenden Bedingungen dar. Das Angebot richtet sich an jedermann. Der Lizenzvertrag kommt durch die Ausübung der in Ziffer 2 und 3 genannten Rechte zustande, insbesondere durch die Vervielfältigung oder Verbreitung des Werkes. Der Erwerber dieser Rechte wird im Folgenden als Lizenznehmer bezeichnet.

(b) Für eine bloße Benutzung des Werkes, etwa das private Anhören eines Tonträgers, Lesen eines Buchs oder Betrachten eines Photos, muss dieser Lizenzvertrag nicht abgeschlossen werden. Dies gilt auch für Befugnisse zur Nutzung des Werkes, die sich aus einer gesetzlichen Beschränkung des Urheberrechts ergeben, etwa für das Anfertigen einer Sicherungskopie oder für die Weitergabe eines rechtmäßig erworbenen Vervielfältigungsstückes.

2. Nutzungsrechte

(a) Der Lizenznehmer erwirbt mit Abschluss der Lizenz das zeitlich und räumlich unbeschränkte Recht, das unveränderte Werk zum Zwecke der Forschung und Lehre an Schulen und Hochschulen in nichtkommerzieller Form zu nutzen. Dies beinhaltet das Recht, das Werk in digitaler und analoger Form, online und offline, körperlich und unkörperlich zu verwenden. Die Nutzung zu anderen Zwecken wird durch diese Lizenz nicht gestattet. Die Nutzungserlaubnis erfolgt lizenzgebührenfrei.

(b) Zur Nutzung wird insbesondere das Recht eingeräumt, das Werk zu vervielfältigen, zu verbreiten, zum Download bereitzuhalten oder in anderer Weise öffentlich zugänglich zu machen, vorzutragen, aufzuführen oder in anderer Form öffentlich wiederzugeben.

(c) Wer das Werk nutzt, darf von anderen Lizenznehmern keine Lizenzgebühren für das Werk verlangen.

(d) Die durch diese Lizenz erworbenen Nutzungsrechte dürfen nicht an Dritte weiterübertragen werden. Dritte können die Nutzungsrechte durch den Abschluss dieser Lizenz nur direkt von den Urhebern oder sonstigen Inhabern der ausschließlichen Nutzungsrechte erwerben. Dafür genügt es, dass Dritte das Werk mit dieser Lizenz von einer beliebigen Person erhalten und gemäß Ziffer 1 den Lizenzvertrag abschließen.

3. Bearbeitungsrecht

(a) Der Lizenznehmer hat das Recht, das Werk zu bearbeiten und das bearbeitete Werk nach Maßgabe der Ziffer 2 zu nutzen. Dies umfasst die Befugnis das Werk zu kürzen, neue Bestandteile hinzuzufügen, Teile des Werkes auszutauschen oder es auf andere Weise zu verändern. Das Werk darf in einen anderen Kontext gestellt und seine Aussagen inhaltlich verändert werden.

(b) Veränderungen dürfen die geistigen oder persönlichen Interessen der Urheber nicht beeinträchtigen. Hierbei ist zu berücksichtigen, dass durch die Lizenzierung unter dieser Lizenz auch substantielle Veränderungen des Werkes bewusst in Kauf genommen werden, da die Freiheit zur Veränderung des Werkes eines der Hauptziele dieser Lizenz ist.

(c) Bei einer Bearbeitung des Werkes muss sein Titel verändert werden. Hierfür genügt das Hinzufügen eines Zusatzes, der die Veränderung des Werkes kenntlich macht, etwa der Zusatz einer neuen Versionsnummer. Der Titel des Werkes darf nicht verändert werden, wenn das Werk ansonsten inhaltlich unverändert genutzt wird.

(d) Es wird empfohlen, für jede Bearbeitung des Werkes einen Urhebervermerk zu den bereits bestehenden Vermerken hinzuzufügen.

4. Freigabe von Bearbeitungen und verwandten Schutzrechten („Copyleft“)

(a) Wer bei der Bearbeitung des Werkes ein Urheberrecht erwirbt, muss dieses Recht den Bestimmungen dieser Lizenz unterstellen, wenn er das bearbeitete Werk verbreitet, zum

Download bereithält oder in anderer Weise öffentlich zugänglich macht, vorträgt, aufführt oder in anderer Form öffentlich wiedergibt.

(b) Eine Bearbeitung in diesem Sinne liegt nicht vor, wenn das unveränderte Werk

- mit einem anderen selbständigen Werk verbunden wird. Dies gilt auch dann, wenn die verbundenen Werke als ein Gesamtwerk genutzt werden;
- in eine Datenbank oder ein sonstiges Sammelwerk eingefügt wird;
- eine Datenbank oder ein sonstiges Sammelwerk ist und weitere Elemente eingefügt werden.

In diesen Fällen muss ein deutlicher Hinweis darauf erfolgen, welche Teile des Gesamtwerkes oder Sammelwerkes dieser Lizenz unterstehen.

(c) Ein selbständiges Werk ist ein Werk, das alleine in sinnvoller Weise genutzt werden kann oder das von der Verkehrsanschauung als selbständiges Werk angesehen wird.

(d) Wer bei der Nutzung oder Bearbeitung des Werkes ein verwandtes Schutzrecht erwirbt, zum Beispiel ein Datenbankherstellerrecht oder ein Recht an einer Interpretation des Werkes, muss dieses Recht den Bestimmungen dieser Lizenz unterstellen, wenn er das Werk verbreitet, zum Download bereithält oder in anderer Weise öffentlich zugänglich macht, vorträgt, aufführt oder in anderer Form öffentlich wiedergibt und das verwandte Schutzrecht für diese Nutzungen erforderlich ist.

5. Namensnennung

(a) Wird das Werk in unveränderter Form verbreitet, zum Download bereitgehalten oder in anderer Weise öffentlich zugänglich gemacht, vorgetragen, aufgeführt oder in anderer Form öffentlich wiedergegeben, müssen Namensnennungen von Urhebern und Interpreten in der vorgefundenen Art und Weise übernommen werden. Die Namensnennung hat dann in einer angemessenen und für die jeweilige Nutzungsart üblichen Form zu erfolgen.

(b) Wird das Werk in inhaltlich veränderter Form verbreitet, zum Download bereitgehalten oder in anderer Weise öffentlich zugänglich gemacht, vorgetragen, aufgeführt oder in anderer Form öffentlich wiedergegeben, darf keine Namensnennung von Urhebern oder Interpreten ohne deren ausdrückliche Zustimmung außerhalb der History erfolgen. Übersetzungen gelten als inhaltliche Veränderung in diesem Sinne. Bei bloß formalen Änderungen muss die Namensnennung entsprechend der Nutzung in unveränderter Form erfolgen. Rechtschreibkorrekturen, Formatierungen oder Digitalisierungen sind im Regelfall als bloß formale Änderungen anzusehen.

(c) Dürfen Urheber oder Interpreten wegen einer inhaltlichen Veränderung des Werkes nicht genannt werden, muss bei jeder Nutzung des Werkes ein Hinweis auf die Urheber oder Interpreten des ursprünglichen Werkes in angemessener Form erfolgen. Ein Hinweis in angemessener Form ist jedenfalls dann gegeben, wenn die History den Anforderungen der Ziffer 8 genügt oder in einer Fußnote die Namensnennung mit dem Zusatz „basierend auf einem Werk von“ erfolgt.

(d) Die vorstehenden Ausführungen zur Namensnennung gelten entsprechend für die Inhaber der ausschließlichen Nutzungsrechte, sofern diese im Zusammenhang mit dem Werk genannt werden.

6. Zugänglichmachung von digitalen Daten

(a) Wer das Werk in unveränderter Form verbreitet, zum Download bereithält oder in anderer Weise öffentlich zugänglich macht, vorträgt, aufführt oder in anderer Form öffentlich wiedergibt, muss die zur weiteren Bearbeitung des Werkes erforderlichen digitalen Daten zugänglich machen, soweit er sie mit dem Werk erhalten hat.

(b) Wer das Werk in veränderter Form verbreitet, zum Download bereithält oder in anderer Weise öffentlich zugänglich macht, vorträgt, aufführt oder in anderer Form öffentlich wiedergibt, muss die zur weiteren Bearbeitung des Werkes erforderlichen digitalen Daten in dem Dateiformat zugänglich machen, das er bei der Bearbeitung verwendet hat. Werden keine digitalen Daten bei der Bearbeitung oder Nutzung verwendet, besteht keine Verpflichtung zur Zugänglichmachung solcher Daten.

(c) Zur Bearbeitung sind solche digitale Daten erforderlich, die zur Erstellung oder Bearbeitung des Werkes verwendet wurden. Wird das Werk in ein anderes Dateiformat konvertiert, ist das ursprüngliche Dateiformat zugänglich zu machen, wenn das Dateiformat, in das konvertiert wurde, eine Bearbeitung nicht zulässt.

(d) Die Zugänglichmachung der digitalen Daten kann in folgender Weise erfolgen:

- durch körperliche Übergabe auf einem Datenträger;
- durch Veröffentlichung auf einem im Werk oder in der History exakt angegebenen, der Öffentlichkeit unbeschränkt zugänglichen Teil eines Datennetzes oder
- in einer anderen Form, die einen entsprechend einfachen Zugang ermöglicht.

(e) Die Zugänglichmachung der digitalen Daten darf unter den Voraussetzungen der Ziffer 7 (b) unterbleiben.

7. Sonstige Verpflichtungen

(a) Bei einer Nutzung in körperlicher Form muss eine Kopie dieser Lizenz beigefügt oder eine Internetadresse angegeben werden, bei der der Lizenztext dauerhaft abrufbar ist. Bei unkörperlicher Wiedergabe des Werkes darf eine Wiedergabe der Lizenz unterbleiben, wenn dies untunlich ist. Dies kann der Fall sein bei Vorträgen und Aufführungen, sowie Fernseh- und Rundfunksendungen.

(b) Hinweise auf die Geltung dieser Lizenz und Urheberrechtsvermerke dürfen nicht verändert oder gelöscht werden. Wo ein solcher Hinweis nach der konkreten Art der Nutzung unzumutbar ist, kann er unterbleiben, so etwa in Rundfunksendungen, die nur terrestrisch, via Kabel oder Satellit übertragen werden oder bei der Nutzung des Werkes in der Fernsehwerbung.

(c) Die Nutzung des Werkes darf nicht von der Erfüllung von Verpflichtungen abhängig gemacht werden, die nicht in dieser Lizenz genannt sind.

(d) Wer im Zusammenhang mit der Nutzung des Werkes sonstige Schutzrechte erwirbt, insbesondere Patente, Marken, Geschmacksmuster und Gebrauchsmuster, darf mittels dieser Schutzrechte keine zusätzlichen Verpflichtungen für die Nutzung des Werkes aufstellen. So ist es etwa nicht zulässig, für eine fortentwickelte Version des Werkes ein Patent anzumelden und für die Nutzung des fortentwickelten Werkes mittels der Patentlizenz Bedingungen aufzustellen, die über die Bedingungen dieser Lizenz hinausgehen.

(e) Die Nutzung des Werkes darf nicht durch technische Schutzmaßnahmen, insbesondere Kopierschutzvorrichtungen und ähnliche Vorrichtungen, verhindert oder erschwert werden, es sei denn, die Nutzung des Werkes wird zugleich ohne solche Vorrichtungen ermöglicht.

8. History

(a) Die History soll Informationen über das Werk, zum Beispiel über seinen Titel, die Urheber und andere Rechtsinhaber, das Veröffentlichungsdatum, vorgenommene Veränderungen und insbesondere den erlaubten Nutzungszweck enthalten.

(b) Ist dem Werk eine History beigefügt, so muss die History bei der Nutzung des Werkes mit den enthaltenen Informationen weitergegeben werden. Insoweit findet Ziffer 7 (a) entsprechende Anwendung.

(c) Ist dem Werk keine History beigefügt, muss bei der Nutzung einer Bearbeitung des Werkes eine History erstellt und weitergegeben werden. Die zu erstellende History muss zumindest die Informationen über das Werk enthalten, die das Werk selbst enthält oder beim Erwerb des Werkes einfach erkennbar waren. Ziffer 7 (a) findet entsprechende Anwendung.

(d) Bei einer Bearbeitung des Werkes muss in der History so genau wie möglich angegeben werden, wo der Ersteller der Bearbeitung das unveränderte Werk erhalten hat. Hierfür genügt die Angabe einer Internetadresse. Das Datum der Veränderung muss in der History vermerkt werden. Veränderungen des Werkes können in der History durch eine kurze Beschreibung dokumentiert werden.

(e) Sofern ein Rechtsinhaber wünscht, dass er vor der Nutzung des Werkes benachrichtigt wird, etwa um eine aktualisierte Version zur Verfügung zu stellen, kann er einen entsprechenden Hinweis in der History aufnehmen. Es wird empfohlen, diesem Wunsch nachzukommen.

(f) Die History darf nur nach den Bestimmungen dieser Ziffer geändert werden.

9. Beendigung der Rechte bei Zuwiderhandlung

(a) Jede Verletzung der Verpflichtungen aus dieser Lizenz beendet automatisch die Nutzungsrechte des Zuwiderhandelnden.

(b) Die Nutzungsrechte Dritter, die das Werk oder Rechte an dem Werk von dem Zuwiderhandelnden erworben haben, bestehen weiter.

10. Haftung und Gewährleistung

(a) Die Haftung der Lizenzgeber ist auf das arglistige Verschweigen von Rechtsmängeln beschränkt.

(b) Dieser Haftungshinweis bezieht sich ausschließlich auf die Einräumung von Rechten durch diese Lizenz. Die Haftung und Gewährleistung für andere Leistungen, etwa die Verbreitung von Werkstücken, richtet sich nach den gesetzlichen Bestimmungen oder individuellen Vereinbarungen.

11. Neue Versionen dieser Lizenz

Das Kompetenznetzwerk Universitätsverbund MultiMedia NRW kann diese Lizenz in Abstimmung mit dem Projekt MuSofT aktualisieren, soweit eine Veränderung der rechtlichen oder tatsächlichen Umstände dies erfordert. Der Lizenzgeber überlässt dem Kompetenznetzwerk Universitätsverbund MultiMedia NRW die Bestimmung des Inhalts künftiger Versionen dieser Lizenz. Die Bestimmung erfolgt durch öffentliche Bekanntgabe des Lizenztextes. Künftige Versionen müssen den Grundprinzipien dieser Lizenz entsprechen. Soweit ein Werk nicht ausdrücklich einer bestimmten Version dieser Lizenz unterstellt ist, gilt die jeweils aktuellste Version.

Anhang: Wie unterstelle ich ein Werk der MuSofT Lizenz?

Um ein Werk nach den Bestimmungen dieser Lizenz zur freien Nutzung durch jedermann zur Verfügung zu stellen, muss dem Werk der folgende Hinweis in gut wahrnehmbarer Weise beigefügt werden. Es wird darüber hinaus empfohlen, einen Urhebervermerk aufzunehmen, der das Jahr der ersten Veröffentlichung sowie den Inhaber der ausschließlichen Nutzungsrechte (Name oder allgemein verständliche Abkürzung) enthält.

```
 Copyright (C) 20[jj] [Name des Inhabers der ausschließlichen
Nutzungsrechte].

Dieses Werk kann durch jedermann zum Zwecke der Forschung und
Lehre an Schulen und Hochschulen in nichtkommerzieller Form
gemäß den Bestimmungen der MuSofT Lizenz genutzt werden.

Die Lizenzbedingungen können unter http://www.uvm.nrw.de/
opencontent oder unter http://www.musoft.org abgerufen sowie
bei der Geschäftsstelle des Kompetenznetzwerkes
Universitätsverbund MultiMedia NRW, Universitätsstraße 11,
D-58097 Hagen, schriftlich angefordert werden.
```

/120/ Volker Gruhn, Ursula Wellen
Autonomies in a Software Process Landscape
Januar 2002

/121/ Ernst-Erich Doberkat, Gregor Engels (Hrsg.)
Ergebnisbericht des Jahres 2001
des Projektes "MuSofT – Multimedia in der SoftwareTechnik"
Februrar 2002

/122/ Ernst-Erich Doberkat, Gregor Engels, Jan Hendrik Hausmann, Mark Lohmann, Christof Veltmann
Anforderungen an eine eLearning-Plattform — Innovation und Integration —
April 2002

/123/ Ernst-Erich Doberkat
Pipes and Filters: Modelling a Software Architecture Through Relations
Juni 2002

/124/ Volker Gruhn, Lothar Schöpe
Integration von Legacy-Systemen mit Eletronic Commerce Anwendungen
Juni 2002

/125/ Ernst-Erich Doberkat
A Remark on A. Edalat's Paper *Semi-Pullbacks and Bisimulations in Categories of Markov-Processes*
Juli 2002

/126/ Alexander Fronk
Towards the algebraic analysis of hyperlink structures
August 2002

/127/ Markus Alvermann, Martin Ernst, Tamara Flatt, Urs Helmig, Thorsten Langer
Ingo Röpling, Clemens Schäfer, Nikolai Schreier, Olga Shtern
Ursula Wellen, Dirk Peters, Volker Gruhn
Project Group Chairware Final Report
August 2002

/128/ Timo Albert, Zahir Amiri, Dino Hasanbegovic, Narcisse Kemogne Kamdem,
Christian Kotthoff, Dennis Müller, Matthias Niggemeier, Andre Pavlenko, Stefan Pinschke,
Alireza Salemi, Bastian Schlich, Alexander Schmitz
Volker Gruhn, Lothar Schöpe, Ursula Wellen
Zwischenbericht der Projektgruppe Com42Bill (PG 411)
September 2002

/129/ Alexander Fronk
An Approach to Algebraic Semantics of Object-Oriented Languages
Oktober 2002

/130/ Ernst-Erich Doberkat
Semi-Pullbacks and Bisimulations in Categories of Stochastic Relations
November 2002

/131 Yalda Ariana, Oliver Effner, Marcel Gleis, Martin Krzysiak,
Jens Lauert, Thomas Louis, Carsten Röttgers, Kai Schwaighofer,
Martin Testrot, Uwe Ulrich, Xingguang Yuan
Prof. Dr. Volker Gruhn, Sami Beydeda
Endbericht der PG nightshift:
Dokumentation der verteilten Geschäftsprozesse im FBI und Umsetzung von Teilen dieser Prozesse im Rahmen eines FBI-Intranets basierend auf WAP- und Java-Technologie
Februar 2003

/132/ Ernst-Erich Doberkat, Eugenio G. Omodeo
ER Modelling from First Relational Principles
Februar 2003

/133/ Klaus Alfert, Ernst-Erich Doberkat, Gregor Engels (Hrsg.)
Ergebnisbericht des Jahres 2002 des Projektes "MuSofT – Multimedia in der SoftwareTechnik"
März 2003

/134/ Ernst-Erich Doberkat
Tracing Relations Probabilistically
März 2003

/135/ Timo Albert, Zahir Amiri, Dino Hasanbegovic, Narcisse Kemogne Kamdem,
Christian Kotthoff, Dennis Müller, Matthias Niggemeier,
Andre Pavlenko, Alireza Salemi,Bastian Schlich, Alexander Schmitz,
Volker Gruhn, Lothar Schöpe, Ursula Wellen
Endbericht der Projektgruppe Com42Bill (PG 411)
März 2003

/136/ Klaus Alfert
Vitruv: Specifying Temporal Aspects of Multimedia Presentations —
A Transformational Approach based on Intervals
April 2003

/137/ Klaus Alfert, Jörg Pleumann, Jens Schröder
A Framework for Lightweight Object-Oriented Design Tools
April 2003

/138/ K. Alfert, A. Fronk , Ch. Veltmann (Hrsg.)
Stefan Borggraefe, Leonore Brinker, Evgenij Golkov, Rafael Hosenberg, Bastian Krol, Daniel Mölle,
Markus Niehammer, Ulf Schellbach, Oliver Szymanski, Tobias Wolf, Yue Zhang
Endbericht der Projektgruppe 415: Konzeption und Implementierung eines digitalen und hypermedialen Automobilcockpits (HyCop)
Mai 2003

/139/ Volker Gruhn, Malte Hülder, Sami Beydeda (Hrsg.)
Endbericht der Projektgruppe 409: Entwicklung von ortsbasierten Diensten für UMTS-Mobilfunkgeräte (mCube)
Mai 2003

/140/ Ernst-Erich Doberkat
Congruences for Stochastic Relations
Juli 2003

/141/ Marion Kamphans, Sigrid Metz-Göckel, Anja Tigges
Wie Geschlechteraspekte in die digitalen Medien integriert werden können – das BMBF-Projekt „MuSofT“
September 2003

/142/ Ernst-Erich Doberkat
Semi-Pullbacks for Stochastic Relations over Analytic Spaces
Januar 2004

/143/ Volker Gruhn, Lothar Schöpe (Hrsg.)
1. Workshop des Verbundforschungsprojektes Mobile Spedition im Web – SpiW
Oktober 2002

/144/ Ernst-Erich Doberkat
Stochastic Relations Interpreting Modal Logic
Oktober 2003

/145/ Alexander Fronk, Ernst-Erich Doberkat, Johannes Bergemann, Ulrich-Walter Gans
Ein interdisziplinäres methodisches Vorgehen zur Gestaltung webbasierter Studieneinheiten für die Altertumswissenschaften
November 2003

/146/ Ernst-Erich Doberkat
Factoring Stochastic Relations
Januar 2004

/147/ Ernst-Erich Doberkat
Characterizing the Eilenberg-Moore Algebras for a Monad of Stochastic Relations
March 2004

/148/ Ernst-Erich Doberkat, Gregor Engels, Corina Kopka (Hrsg.) Abschlussbericht des Projektes “MuSofT – Multimedia in der SoftwareTechnik”
April 2004