UNIVERSITÄT DORTMUND

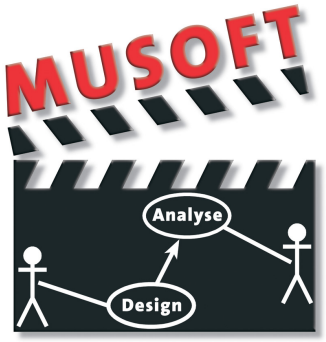
Fachbereich Informatik
Lehrstuhl für Software-Technologie

M E M O Nr. 121

**MuSofT Bericht Nr. 1**

# Ergebnisbericht des Jahres 2001 des Projektes "MuSofT – Multimedia in der SoftwareTechnik"

Ernst-Erich Doberkat Gregor Engels (Hrsg.)




Februar 2002

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

# Ergebnisbericht des Jahres 2001 des Projektes “MuSofT– Multimedia in der SoftwareTechnik”

Ernst-Erich Doberkat Gregor Engels
(Hrsg.)

Februar 2002

## Vorwort

Das Vorhaben *MuSofT – Multimedia in der Softwaretechnik* wird seit dem 1. März 2001 vom Bundesministerium für Buildung und Wissenschaft im Rahmen des Programms Neue Medien in der Bildung gefördert. An diesem Vorhaben nehmen die folgenden Hochschulen teil: Fachhochschule Lübeck, Otto-von-Guericke-Universität Magdeburg, Universität Paderborn, Universität Dortmund, Universität Siegen, Universität Stuttgart und die Universität der Bundeswehr in München. Wir haben uns in diesem Projekt vorgenommen, die Ausbildung in der Softwaretechnik an den Stellen, an denen es sinnvoll erscheint, durch den Einsatz Neuer Medien zu unterstützen. Das Vorhaben wird bis zum Ende des Jahres 2003 laufen.

Mit dieser Sammlung wollen wir nach etwa einjähriger Laufzeit die ersten Projektergebnisse vorstellen. Wir haben uns im Zeitplan des Projekts vorgenommen, gründlich über die verwendeten Konzepte und Lehrinhalte nachzudenken, die Frage der Plattformen gerade am Anfang nicht auszuklammern, aber auch zu überlegen, wie wir die Nachhaltigkeit unserer Entwicklungen schon während der Projektarbeit sichern können. Das alles und noch viel mehr ist in den Beiträgen dieser Sammlung diskutiert. Um von der Struktur gleichförmige Beiträge zu bekommen, haben wir eine grobe Gliederung vorgegeben, um gleichförmige Qualität zu erreichen, haben wir die Beiträge intern begutachten lassen; die Ergebnisse der Begutachtung sind in die vorliegenden Darstellungen einge¤ossen.

So möchten wir denn allen beteiligten Kolleginnen und Kollegen für ihre Bereitschaft danken, nicht nur Beiträge für diesen ersten übergreifenden Projektbericht zu schreiben, sondern ihre Zeit auch als Gutachter zur Verfügung zu stellen (es wird nicht das letzte Mal sein). Klaus Alfert hat als Projektmanager ef£zient dazu beigetragen, daß diese Sammlung geräuschlos und rechtzeitig zustande kam.

Dortmund und Paderborn,
im Februar 2002

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

# Inhaltsverzeichnis

## Autoren

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

Dipl.-Inform. Jörg Pleumann
Lehrstuhl für Software-Technologie
Fachbereich Informatik
Universität Dortmund

Prof. Dr. Silke Seehusen
Dipl.-Ing. (FH) Nina Nissen
Fachhochschule Lübeck

Dipl.-Inform. Peter Aschenbrenner
Prof. Dr. Andy Schürr
Institut für Softwaretechnik
Fakultät für Informatik
Universität der Bundeswehr, München

Prof. Dr.-Ing. Fritz Schmidt
Institut für Kernenergetik und
Energiesysteme
Universität Stuttgart

Dipl.-Inform. Corina Kopka
Lehrstuhl für Software-Technologie
Fachbereich Informatik
Universität Dortmund

# MuSofT-Teilprojekt 1.1 Ergebnisbericht 2001

Gregor Engels, Jan Hendrik Hausmann, Marc Lohmann, Annika Wagner

Das Teilprojekt 1.1 im Projekt MuSofT beschäftigt sich mit der Ausarbeitung von Lehrmaterialien zum Thema Anforderungsde£nition. Besondere Schwerpunkte der dabei entstehenden Lehreinheit liegen auf dem Einsatz von multimedialen Materialien zur Illustration realistischer Anforderungen sowie dem Aufbau von Erfahrungswissen bei den Studierenden.

## Inhaltsverzeichnis

# 1 Einleitung

Das Gebiet der Softwaretechnik beschäftigt sich mit der zielorientierten Entwicklung von Software. Die Ziele der Entwicklung werden dabei in der Anforderungsde£nition vom Anwender vorgegeben. In dieser Phase werden die unterschiedlichen Informationen erfasst, strukturiert und festgehalten, die als Grundlage aller folgenden Entscheidungen im Entwicklungsprozess dienen. Aktuelle Studien belegen, dass Fehler, die in dieser Phase begangen werden, nicht nur schwer zu korrigieren sind, sondern in vielen Fällen auch zum Scheitern ganzer Projekte führen. Trotzdem wird dieses zentrale Thema in der universitären Ausbildung häu£g mit einem geringeren Fokus versehen, als z.B. implementierungsnahe Gebiete wie Algorithmen und Architekturen. Ein Grund hierfür ist mit Sicherheit in der Tatsache zu suchen, dass viele der Probleme, die es in der Anforderungsde£nition zu lösen gilt, die angestammte Domäne des Informatikers verlassen und eher nach Fähigkeiten aus dem sozialwissenschaftlichen bis hin zum ethnologischen Bereich verlangen, da die Schwierigkeiten aus der unpräzisen Kommunikation zwischen den verschiedenen Beteiligten resultieren. Weder die Art der Schwierigkeiten noch Vorgehensweise zu deren Lösung lassen sich in einer Art und Weise formalisieren, wie dies in vielen anderen Bereichen der Informatik möglich ist. Die vorherrschenden universitären Lehrmethoden der Informatik sind daher nur unzureichend auf die Vermittlung solcher Probleme ausgelegt. Es ist deshalb das Ziel der Lehreinheit 1.1, Material zur Verfügung zu stellen, dass diese Vermittlung besser leisten kann. Hierfür sind insbesondere Materialien geeignet, die das Problemfeld (also die Anforderungen der Benutzer) möglichst unge£ltert darstellen. Der Einsatz multimedialer Techniken bietet sich dafür besonders an, da diese in der Lage sind, Originalmaterialien aus der Anforderungsdomäne darzustellen (Pläne, Anleitungen), Kundengespräche und Beobachtungen wiederzugeben (Videos) oder komplexe Abläufe zu visualisieren (Animationen). Weiterhin muss es das Ziel der Einheit sein, beim Studierenden ein Erfahrungswissen aufzubauen, das es ihm erlaubt, in realen Situationen die Problemstellungen zu erkennen und geeignete Methoden zu deren Behebung auszuwählen.

# 2 Inhalt

## 2.1 Inhaltliche Lernziele

Die Lehreinheit Anforderungsde£nition soll Probleme aufzeigen, die typischerweise beim Erfassen, Strukturieren und Notieren der Anforderungen von Benutzern an ein neues Softwaresystem entstehen. Sie soll darüber hinaus den Studierenden die Fähigkeiten vermitteln, diesen Problemen in einem praktischen Kontext zu begegnen.

Die Aufgaben, die bei der Anforderungsde£nition im Einzelnen zu erfüllen sind, lassen sich nach [vL00] wie folgt kategorisieren:

- **Informationsbeschaffung:** Es müssen Techniken zur Beschaffung von Informationen über die Anforderungen bereitgestellt werden. Diese reichen von Beobachtungen über die Analyse existierender Dokumente bis hin zu Interviewtechniken.

- **Domänenanalyse:** Die Umgebung, in der das Softwaresystem eingesetzt werden soll, muss erfasst und verstanden werden. Beteiligte müssen identi£ziert und ihre Ziele ge-

klärt werden. Probleme im existierenden System müssen aufgedeckt und Verbesserungsmöglichkeiten erarbeitet werden.

- **Zielde£nition:** Alternative Modelle für das zu entwickelnde System müssen erarbeitet und ihr Potential im Hinblick auf die zuvor erkannten Probleme muss eingeschätzt werden. Es müssen Grenzen zwischen dem zukünftigen Softwaresystem und den weiter bestehenden Strukturen de£niert werden.
- Verhandlungen: Die alternativen Modelle müssen mit den Beteiligten abgestimmt werden, Widersprüche müssen aufgelöst und Kompromisse erreicht werden.
- **Spezi£kation:** Die erarbeiteten Informationen müssen in einer präzisen Art erfasst werden.
- **Spezi£kationsanalyse:** Es müssen Prüfungen der gesammelten Informationen auf Inkonsistenzen, Widersprüche und Lücken durchgeführt werden. Aufwands- und Risikobewertungen müssen durchgeführt werden.
- **Dokumentation:** Die Entscheidungen, die in der Anforderungsanalyse fallen, müssen nachvollziehbar dokumentiert werden.
- **Evolution:** Die Anpassung an Änderungen, Korrekturen oder neue Ziele muss unterstützt werden.

In der klassischen Lehre wird hauptsächlich die Spezi£kation vorgegebener Situation (z.B. durch den Einsatz von Use-Case-Diagrammen) geübt. Darüber hinaus ist es das Ziel der LE 1.1 Probleme der Domänenanalyse, Zielde£nition und der Spezi£kationsanalyse zu lehren. Dazu sollen die Studierenden *Kenntnisse* zu folgenden Tätigkeiten erwerben:

- Strukturelle Zusammenhänge und Terminologie eines Anwendungsbereiches erfassen,
- Abläufe erkennen, Prozesse identi£zieren und De£zite in diesen aufdecken,
- Ziele zur Verbesserung existierender Abläufe formulieren und diese bewerten,
- funktionale Anforderungen an das zu entwickelnde System formulieren und
- nicht-funktionale Anforderungen in die Spezi£kation einarbeiten.

Dazu müssen die Studierenden folgende *Vorgehensweisen* kennen und an verschiedenen Stellen anwenden lernen:

- Einsatz von pragmatischen Regeln zur Bewertung/Analyse von Problemen
- Erkennen und Entscheiden von Trade-Off-Situationen, Begründen der Entscheidung
- Erkennen von Inkonsistenzen und Lücken in den Anforderungen
- Erstellen von Modellen
- Prüfen von/Fehlersuche in Modellen
- Iterieren über existierende Modelle

## 2.2 Übergeordnete Lernziele

Insbesondere im Bereich der ‘weichen’ Probleme ist ein deduktiver Lehransatz kaum zu realisieren, da allgemein de£nierte Lösungen nicht existieren. Vielmehr ist es entscheidend, den Studierenden einen problemorientierten Zugang zu den Themen der Anforderungsde£nition zu ermöglichen, da erst durch das Erkennen und Erfahren der ‘Unschärfe’ der Probleme auch eine Akzeptanz für die ungewöhnlich ‘weichen’ Lösungsansätze zu erreichen ist. Die Studierenden müssen also Erfahrungswissen im Bereich Anforderungsde£nition sammeln, was normalerweise nur durch die Teilnahme an größeren Softwareprojekten möglich ist, für die ihnen jedoch die Quali£kationen fehlen. Weiterhin soll eine Brücke vom geleiteten Lernen der Grundstudiumsstudierenden hin zu einer selbstständigeren und inhaltlich orientierten Lernform geschlagen werden. Es ist deswegen unser Ziel, einen hohen Anteil der Lehreinheit als Übungen zu gestalten, in denen die Studierenden geleitet und dann zunehmend selbständig die notwendigen Erfahrungen machen können.

# 3 Didaktisches Konzept

Um Lehreinheiten zu konstruieren, die auch übergreifend einsetzbar sind, ist es gemäß [Ker98] und den Ausführungen von KT1 unerlässlich zuerst die Zielgruppe und ihre Eigenschaften genau zu untersuchen. Nur so kann gewährleistet werden, dass die in der Lehreinheit eingesetzten didaktischen Mittel und Methoden geeignet sind, um die Ziele der Einheit zu erreichen. Insofern ist dieses Kapitel eine Art Ist-Analyse für die Lehreinheit selbst.

## 3.1 Analyse der Zielgruppe

Die Lehreinheit Anforderungsde£nition soll zuerst an der Universität Paderborn im Rahmen der Vorlesung Techniken des Softwareentwurfs zum Einsatz kommen, hierbei soll auch die Evaluation der Lehrmaterialien erfolgen. Wir orientieren uns daher an dieser speziellen Zielgruppe, auch wenn wir einen breiteren Einsatzkontext im Blick behalten. Die Hörer dieser Vorlesung setzen sich wie folgt zusammen:

- Informatiker: Grundstudium, 3.Semester, P¤ichtveranstaltung Hochschulstudiengang mit Ziel Diplom, Diplom H1, Master/Bachelor, Vorkenntnisse: OO-Pro-grammierung, grundlegende Modelle (ER-Diagramme, Petri-Netze, Automaten)
- Wirtschaftsinformatiker: Hauptstudium, Vertiefungsveranstaltung Hochschulstudiengang mit Ziel Diplom, Master/Bachelor Vorkenntnisse: OO-Programmierung, Datenbanken, grundlegende Modelle
- Sonstige: Verschiedene Studiengänge sind mit Neben-/Vertiefungsfach Informatik vertreten. Unter anderem Ingenieurwissenschaftler, Wirtschaftingenieure, Mathematiker etc., außerdem Lehramtsstudierenden der Informatik.

Zahlenmäßig bilden jedoch die Gruppen der Infos und WInfos mehr als 90% der Hörerschaft, so dass auf deren Vorwissen aufgebaut wird.

Das allgemeine Leitbild (unabhängig von diesem speziellen Einsatzszenario) für diese Lehreinheit geht von grundlegenden Vorkenntnissen im Bereich der Objektorientierung aus. Ein Einsatz im Anschluss an einen grundlegenden Programmierkurs ist vorstellbar. Hilfreich für den Einsatz der Lehrmaterialien ist weiterhin ein grundlegendes Verständnis für Softwareentwicklungsprozesse. Es ist jedoch weder der universitäre Kontext noch die Bindung ans Grundstudium festgelegt. Die Lehreinheit soll so modular aufgebaut werden, dass einführende Teile in einer Grundlagenveranstaltung verwandt werden können, die Gesamtheit der Materialien jedoch auch die Grundlage für eine Spezialveranstaltung bilden kann (entsprechend dem universitären Hauptstudium). Den Einstz im Kontext von anderen Ausbildungsgängen (Informatik-Serviceveranstaltungen, Fachhochschulen, Weiterbildungsangebote) halten wir wegen der Realistätsnähe des Stoffes für denkbar, weisen aber klar darauf hin, dass die Lehreinheit nicht auf diesen Einsatzzweck hin konstruiert wurde und Anpassungen nötig sein können.

## 3.2 Motivation der Zielgruppe

Da es sich bei der überwiegenden Zahl der Hörer (Informatiker) um Studierende im Grundstudium handelt, muss man von einer extrinsischen Motivation der Teilnehmer ausgehen, d.h. die Teilnahme geschieht nicht aus Interesse am Stoff, sondern weil sie vom Studienplan vorgeschrieben ist. Bei den Wirtschaftsinformatikern ist zwar eine Wahlfreiheit vorhanden, diese ist jedoch eingeschränkt. Es ist allerdings zu beobachten, dass die Wirtschaftsinformatiker in dieser Phase ihres Studiums bereits einen breiteren Horizont und bessere analytische Fähigkeiten haben. Diese Studierenden sind -auch aufgrund von Vorerfahrungen in Projektarbeitdann bereits intrinsisch motiviert, was sich deutlich in ihren Ergebnissen niederschlägt.

## 3.3 Lerngewohnheiten

Die Grundstudiumsstudierende sind an einen geleiteten Studienablauf gewöhnt. Es wird ein regelmäßiger Vorlesungsbesuch erwartet, aber nicht kontrolliert. Vorlesungsbegleitend werden Übungsblätter ausgegeben, deren Bearbeitung und Rückgabe durch die Studierenden optional ist, jedoch mit einem Anreizsystem (Bonus für die Klausurnote) versehen ist. Auch hierbei ist ein klarer Unterschied zwischen den Informatikern (spärliche Abgabe) und den Wirtschaftsinformatikern (sehr detaillierte Bearbeitung) festzustellen. Der Umgang mit kreativen Aufgaben fällt allerdings den Informatikern erfahrungsgemäß leichter. Großes Unbehagen bereitet allen Studierenden der Umgang mit ‘weichem’ Wissen, damit sind Gebiete gemeint, in denen kein exaktes Wissen gelehrt wird, sondern Abwägungen getroffen und Sachverhalte kontextabhängig interpretiert werden müssen. In diesen Bereichen beklagen sich die Studierenden häu£g über Orientierungslosigkeit und eine schlechte Klausurvorbereitung.

Zur Zeit wird von einem Verhältnis von 4 SWS Vorlesung zu 2 SWS Übung sowie einer veranschlagten Zeit von etwa 4 SWS Vor-/Nachbereitung und Übungsbearbeitung ausgegangen.

## 3.4 Lernorte und Ausstattung

Die Studierenden lernen unserer Beobachtung nach überwiegend zu Hause, wenn sie nicht durch die Benutzung zentraler Betriebsmittel (Bibliothek, Poolrechner etc.) einen deutlichen Mehrwert im Lernen in der Universität erkennen. Auch Lerngruppen werden eher im privaten Bereich organisiert. Ein Zugang zum Intra- und Internet ist für alle Studierenden auf dem Campus gewährleistet, die Rechneraussattung der Poolräume ist als ausreichend einzuschätzen (auch wenn in Spitzenzeiten Vollauslastungen auftreten können). Insbesondere ist zur Zeit über Lehrlizenzen eine ausreichende Versorgung mit kommerziellen CASE-Tools (Rational Rose und Together) sichergestellt. Darüber hinaus ist eine sehr gute private Rechnerausstattung (inklusive Internetzugang) bei vielen Studierenden zu verzeichnen.

## 3.5 Lernszenarien

Die im Rahmen unserer Lehreinheit erstellten Materialien können in verschiedenen Organisationsformen eingesetzt werden

- Es können „klassische“ Veranstaltungen mit einem wissensvermittelnden Vorlesungsteil und einem vertiefenden Übungsbetrieb mit diesen Materialien gehalten werden. In dieser Organisationsform können zwar insbesondere die übergreifenden Lernziele nicht voll erreicht werden, aber sie fügt sich am nahtlosesten in die existierenden Strukturen der Universität ein. Das multimediale Format der meisten Materialien schließt jedoch die Distribution in Form von gedruckten Übungszetteln weitgehend aus. Stattdessen ist eine elektronische Distribution über E-Mail sinnvoll. Auch die Abgabe der Übungen soll auf diesem Weg vor sich gehen. Dabei kann die im Rahmen des MuSoft-Projektes zu erstellende Übungsplattform zum Einsatz kommen.

- Neben dem an der Vorlesung orientierten Übungsbetrieb ist auch eine stärker übungszentrierte Lehr-/Lernform denkbar. Bei dieser würde der Ansatz der Problemorientierung stärker zur Geltung kommen. Die Präsenzlehre würde also eher erkannte Probleme thematisieren/vertiefen als zusätzliche Informationen bereitstellen. Auch hierbei käme die Übungsplattform zum Einsatz, wobei der Studierende als Feedback auf eine Aufgabe jedoch auch weitergehende Informationen anfordern kann. Auf diese Weise kann das Lerntempo/die Lerninhalte vom Lernenden mitgestaltet werden. Eine solche Veranstaltung würde jedoch sicherlich eher den Charakter einer Vertiefung haben und nicht unbedingt im Grundstudium angeboten werden.

- Schließlich kann mit der Idee des simulierten Projektes auch eine weitgehend unabhängig von der Präsenzlehre einzusetzende Einheit konstruiert werden, die es Studierenden erlaubt, ihre Fähigkeiten im Bereich der Anforderungsde£nition zu testen/zu erweitern.

Für alle Einsatzgebiete von übungszentrierten Lernmaterialien sind Überlegungen zum Aufwand der Korrektur/Bewertung unerlässlich. Hierbei setzen wir auf eine im Rahmen des Projektes ‘virtuelle Welt’ entwickelte Konzeption auf, die auch eine teilautomatische Korrektur von komplexen Lösungen ermöglicht. Insbesondere ist es mit den dort entwickelten Techniken möglich, Verweise zwischen Mediendaten (z.B. Stellen in einem Video) und in

CASE-Tools erstellten Modellen zu de£nieren, so dass im Modell manifestierte Entscheidungen mit Belegen des Ursprungsmaterials versehen werden können.

# 4 Strukturkonzept

Die Lehreinheit zerfällt in Lehr- und Übungsmaterialien, die jedoch eng miteinander verwoben sind. Im Bereich der Lehrmaterialien sind folgende Module zu unterscheiden:

- **Informationsbeschaffung:** Hier werden die unterschiedlichen Erfassungstechniken thematisiert, die zur Verfügung stehen, um an die (meist impliziten) Anforderungen an das zu erstellende System zu gelangen. Das Spektrum reicht hier von Fragetechniken für Interviews bis hin zur Auswertung von Beobachtungsvideos. Dieses Themengebiet ist am weitesten von der gewohnten Welt des Informatikers entfernt. Vielmehr werden hier insbesondere soziale Fähigkeiten verlangt, die so nicht zur Kernquali£kation eines Informatikers gehören, aber den Erfolg dieser Phase entscheidend bestimmen.

- **Ist-Analyse:** Liegt eine hinreichende Menge an Informationen vor, so ist der erste Schritt zur Strukturierung die Erstellung des Ist-Modelles. Typischerweise zerfällt diese in zwei Teile, die Erstellung eines Modells des Problembereiches und die Geschäftsprozessanalyse.

  - **Modell des Problembereiches:** Das Modell des Problembereiches stellt in der Form von UML Objekt- und Klassendiagrammen die Strukturen dar, die in der Problemdomäne erkannt wurden. Beim Übergang von den typischerweise unstrukturierten und heterogenen Informationen vom Auftraggeber zu einem solchen formalen Modell treten Kon£kte auf, wenn die Beschreibung Homonyme und Synonyme enthielt oder wenn sie sprüchlich oder unvollständig ist. Weiterhin muss man von irrelevanten Details abstrahieren und ein präzises und redundanzfreies Modell erstellen.
  - **Geschäftsprozess:** Neben der Strukturierung der Domäne sind auch die erkannten Abläufe relevant für die Implementierung des zukünftigen Systems. Mit UML Aktivitätendiagrammen ist man in der Lage, solche Geschäftsprozesse zu erfassen. In vielen Firmen liegen heutzutage auch bereits Geschäftsprozessmodelle in verschiedenen Notationen vor, es bieten sich hier also Ansatzpunkte für die Integration verschiedener Notationen.

- **Sollkonzept:** Für das zu entwickelnde System ist die aktuelle Situation natürlich nur der Ausgangspunkt, entscheidend ist jedoch, wo das System bestehende Abläufe unterstützen oder neue ermöglichen soll. Um solche Informationen herauszuarbeiten, benutzt man folgende Techniken:

  - **Zielmodell:** Der Einsatz von Software in Unternehmen dient häu£g verschiedenen Zwecken von unterschiedlichen Beteiligten. Eine Akzeptanz der Software kann nur dann erreicht werden, wenn alle Ansprüche berücksichtigt oder Kompromisse ausgehandelt werden. Zur Erkennung solcher möglichen Widersprüche

werden Zielmodelle aufgestellt und analysiert. Ein erfolgreiches Zielmodell gibt Aufschluss über viele der erforderlichen Funktionalitäten oder nicht-funktionalen Anforderungen.

  - **Use-Cases:** Realistische Softwaresysteme bieten dem Benutzer eine Vielzahl von Funktionen, deren Kombination unterschiedliche Szenarien ergibt und die intern wiederum durch viele kleinere Funktionen realisiert sind. Um eine Strukturierung der funktionalen Anforderungen zu erhalten setzt man Use-Case-Modelle ein. Ein Use-Case fasst dabei die für den Benutzer erkennbaren Funktionalitäten in einem bestimmten Szenario zusammen. Use-Cases werde annotiert durch Use-Case-Beschreibungen, die in semi-formaler Spezi£kation Vor- und Nachbedingungen sowie den Ablauf des Use-Cases festhalten.

- **Modellzusammenhang und Iteration:** Die hier in verschiedene Module unterteilten Vorgänge sind in der Praxis eng miteinander verwoben. Die Zusammenhänge zwischen den erstellten Modellen (Konsistenz) und den verschiedenen (vorher separat betrachteten) Tätigkeiten aufzuzeigen und die Möglichkeiten zur Iteration und Evolution der Modelle aufzuzeigen ist Inhalt des letzten Modules. Hier sollen auch aktuelle Forschungsarbeiten zur Konsistenz von Modellen der Anforderungsanalyse [HH01] sowie der frühen Fehlererkennung angesprochen werden, die aufgrund von automatisierten Prüfungen Hinweise zu einer verbessernden Iteration der Modelle geben können [HHT01].

Für die Illustration der Lehreinheit sowie den Übungsteil setzen wir eine Reihe von Fallstudien ein, die sich in unterschiedlichen Schwierigkeitsgraden den identi£zierten Lernzielen annähern.

- **Fallstudie I -Gesellschaftsspiel-**: Ein Gesellschaftsspiel soll als Computerspiel umgesetzt werden. In einem solchen Szenario ist die Problemdomäne allgemein verständlich, es muss keine Auseinandersetzung mit Fachvokabular statt£nden. Die Dimension des Problems ist gut handhabbar, jedoch auch leicht für Spezialfälle zu erweitern und mit der Spielanleitung liegt typischerweise ein gut vorstrukturiertes Dokument vor, das als Grundlage der Analysen dienen kann. Weiterhin sind die funktionalen Anforderungen im Allgemeinen bei solchen Anwendungen sehr einfach strukturiert. Der Schwerpunkt dieser Einheit liegt auf dem Erkennen von Strukturen im Anwendungsbereich und deren Notation. Zusätzlich sollen dynamische Abläufe erkannt und spezi£ziert werden. Durch die Herausarbeitung von Spezialsituationen kann das Problem der Unvollständigkeit von Spezi£kationen illustriert werden.

- **Fallstudie II -N.N.-**: Eine Steigerung des Schwierigkeitsgrades ist möglich, indem man die Einarbeitung in eine komplexere Domäne verbindet mit facettenreichen funktionalen Anforderungen mehrerer Benutzer an das System. Hierbei ist vorgesehen, auf Dokumentationen über Eigenentwicklungen zurückzugreifen (Entwicklung einer Klausurverwaltung, Entwicklung eines XML-Mailtools)

- **Fallstudie III -N.N.-** Das wohl komplexeste Szenario dieser Lehreinheit repräsentiert eine Anwendungsentwicklung in einem realistischen, industriellen Umfeld. Hierbei soll ein echter Entwicklungsprozess einer kooperierenden Software£rma von uns begleitet

werden, um so Material für eine hinreichend realistische Dokumentation zu erlangen. Aus Gründen der Materialbeschaffung kann auch eine Softwareentwicklung an der Uni begleitet werden. Mit diesen Materialien kann für den Studierenden ein Szenario konstruiert werden, das die volle Bandbreite der Probleme der Anforderungsde£nition enthält.

Zu jeder dieser Fallstudien sollen jeweils diverse multimediale Materialien bereitgestellt werden:

- Videos und Animationen, die Abläufe zeigen/veranschaulichen
- Originaldokumente aus der Anforderungsdomäne
- Modelle, die Teile der Anforderung repräsentieren sollen

Weiterhin sind zu jeder Fallstudie eine Reihe unterschiedlicher Aufgabenfolgen zu erstellen, die den Studierenden dazu bringen sollen

- sich in die Domäne einzuarbeiten (inklusive der selbstständigen Anforderung ergänzender Informationen)
- Probleme aufzuzeigen/Entscheidungen zu treffen,
- Modelle zu erstellen (von Hand, unter Einsatz von CASE-Tools),
- oder gegebene Modelle zu erweitern/überprüfen.

Auch diese Aufgabenfolgen sind in ihrem Schwierigkeitsgrad zu variieren, von einer Version, die dem Studierenden viele Informationen zum von ihm erwarteten Verhalten an die Hand gibt, bis hin zu einem -ursprünglich für das Projekt ‘virtuelle Welt’ entwickelten- Szenario, in dem der Studierende in einer simulierten Projektumgebung mit Problemen konfrontiert wird, die er/sie selbständig bearbeiten soll.

# 5 Lehr/Lernumgebung

Bezüglich der technischen Voraussetzungen stellt die LE 1.1 Anforderungsde£nition nur sehr geringe Anforderungen für den Einsatz der Materialien. Die von uns erstellten Vorlesungsteile werden im Powerpoint-Format von Microsoft erstellt werden, für das unter den gängigsten Betriebssystemen Viewer bzw. Editoren verfügbar und in eigentlich allen Universitäten auch im Einsatz sind. Die multimedialen Teile werden jedoch auch einzeln über das geplante MuSofT-Portal abrufbar sein und werden dabei in ihren Ursprungsformaten vorliegen. Gemäß den Vorgaben des KT4 werden dabei Ablageformate verwandt, die eine hohe Verbreitung haben (z.B. mpeg für Videos). Damit können die Materialien von TP 1.1 ¤exibel an jede Art von technisch unterstützter Lehr/Lernumgebung angepasst werden. Bei der Unterstützung von Übungsaufgaben wurde bereits auf die Möglichkeit zur Entwicklung teilautomatischer Korrekturmodule hingewiesen; dies macht jedoch nur Sinn, wenn eine entsprechende Plattform dafür zu Verfügung gestellt wird. Hier warten wir das Ergebnis der technischen Koordination (KT 4) ab.

# 6 Technische Realisierung

Im aktuellen Stand des Projektes sind Materialien zur ersten Fallstudie entwickelt, eingesetzt und evaluiert worden. Die Begleitung der Entwicklung eines Web-Portales ist als Vorlage für die dritte Fallstudie für die erste Hälfte des kommenden Jahres geplant. Wir wollen dabei möglichst viel Material direkt im Original (z.B. per Video) erfassen, um möglichst realistische Darstellungen zu erhalten. Es ist uns jedoch noch nicht klar, inwieweit dieses Vorhaben gelingt, ob also die Auftraggeber dieser Softwareentwicklung bereit und in der Lage sind, die Dokumentation in dieser Weise durchzuführen. Eine Option ist es daher, die in dieser Fallstudie gewonnenen Erfahrungen in Form von Drehbüchern für nachgestellte Szenen zu nutzen. Bei der Erstellung dieser Drehbücher ist eine besonderer Fokus darauf zu legen, eine Verdichtung der Darstellung zu erreichen (Besprechungen dauern oft mehrere Stunden, als Lehrmaterial sind Szenen von maximal 10 min geeignet), ohne dabei jedoch die Informationsqualität gravierend zu erhöhen, da dies eine Simpli£zierung und Beein¤ussung der Authentizität darstellen und daher dem gewünschten Charakter der Materialien widersprechen würde.

# 7 Evaluierung

Um Fehlentwicklungen zu vermeiden, ist eine beständige Evaluation der erstellten Materialien in der Lehrpraxis Aufgabe des Teilprojektes 1.1. Wir haben dazu eine erste Version der von uns erstellten Materialien im Wintersemester 2001/02 in der Vorlesung ’Techniken des Softwareentwurfes I’ an der Universität Paderborn eingesetzt. Da die Materialien hauptsächlich den handlungsorientierten Zugang unterstützen sollten, wurden sie zur Erstellung von Übungsaufgaben eingesetzt. Um eine anschließende Evaluierung der Ergebnisse und Erfahrungen zu ermöglichen wurden zwei Maßnahmen zur Sicherung der Ergebnisse getroffen: Zum Einen wurden sämtliche von Studierenden angefertigten Lösungen vervielfältigt, um eine genaue Auswertung der gemachten Fehler zu ermöglichen, zum Anderen wurden mit Hilfe eines strukturierten Fragebogens Informationen zur Arbeitsweise der Studierenden und der Beurteilung unseres Materials gesammelt. Eine erste Auswertung dieser Informationen ist verfügbar und es lassen sich positive Effekte im Hinblick auf die verfolgten Lernziele erkennen. Allerdings werden auch Schwächen sowohl der Lehrmaterialien als auch der Übungsaufgaben erkennbar, die es in einer nächsten Iteration auszumerzen gilt. Ein Nachteil dieser Evaluation besteht darin, dass es sich nur um eine Querschnittsuntersuchung handelt und keine Informationen im Vergleich zu einer nicht multimedial unterrichteten Gruppe vorliegen. Für Längsschnittuntersuchungen ist mit dieser Erhebung jedoch eine Basis geschaffen worden, gegen die man die Ergebnisse der in den kommenden Jahren zu entwickelnder MuSoft-Einheiten messen kann. Somit ist in diesem Punkt eine aktive Qualitätsverbesserung ebenso ermöglicht worden wie eine nachträgliche Erfolgsprüfung. Der vorläu£ge Bericht über die Ergebnisse der Evaluation ist voraussichtlich ab Januar 2002 auf dem MuSofT-Server verfügbar.

# 8 Zusammenfassung

Die Lehreinheit 1.1 -Anforderungsde£nition- soll realitätsnahe und handlungsorientierte Materialien für das Erlernen und Erfahren von Problemen und Tätigkeiten bei der Anforderungsde£nition bereitstellen. Die Konzeption der LE ist in diesem Bericht dargestellt, erste Materialien existieren und sind auch eingesetzt und evaluiert worden. Ausgehend von diesen Erfahrungen soll eine Verbesserung der existierenden Materialien sowie die gezielte Entwicklung neuer Lehrmodule im nächsten Jahr statt£nden.

# Literatur


[HH01] HAUSMANN, JAN HENDRIK und REIKO HECKEL: *Use Cases as views: A formal approach to Requirements engineering in the Uni£ed Process*. In: *Informatik 2001 - Tagungsband der GI/OCG Jahrestagung*, Seiten 595–600, Wien, 2001.

[HHT01] HAUSMANN, JAN HENDRIK, REIKO HECKEL und GABI TAENTZER: *Detection of Con¤icting Requirements in a Use Case Driven Approach: A static analysis technique based on graph transformations*. accepted for ICSE 2002, 2001.

[Ker98] KERRES, MICHAEL: *Multimediale und telemediale Lernumgebungen : Konzeption und Entwicklung*. Oldenbourg, Munich, 1998.

[vL00] LAMSWEERDE, AXEL VAN: *Requirements engineering in the year 00: a research perspective*. In: *International Conference on Software Engineering*, Seiten 5–19, 2000.

# Zwischenbericht zu LE 1.2 – Entwicklung von Informationssystemen

Dirk Jesko

Im Rahmen des MuSofT-Projekts wird untersucht, wie die Lehre in verschiedenen Bereichen der Softwaretechnik durch den Einsatz neuer Medien, Präsentations- und Kommunikationstechniken verbessert werden kann. Die Lerneinheit 1.2 befaßt sich dabei mit dem Themengebiet „Datenbanken und Informationssysteme“, welches einen wesentlichen Teil vieler Informatikstudiengänge darstellt. Insbesondere beim Datenbankentwurf ist die Vermittlung praktischer Fähigkeiten von Bedeutung. In herkömmlichen Veranstaltungen ist dies nur unzureichend möglich. Die Lerneinheit 1.2 hat daher das Ziel, die Lehre in diesen Bereichen zu unterstützen. In diesem Bericht werden die Ergebnisse der ersten Projektphase zusammengefaßt. Es wird auf didaktischer Aspekte, eine Abgrenzung der betrachteten Inhalte und deren Strukturierung sowie Aussagen zur technischen Realisierung und zur Evaluation eingegangen. Weiterhin werden Anforderungen an eine Plattform de£niert, die die Verwaltung und Bereitstellung der Materialien unterstützen soll.

## Inhaltsverzeichnis

# 1 Einleitung

Die Entwicklung von Datenbanken 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 deren Umsetzung mit realen Werkzeugen, erfolgt nur in unzureichendem Umfang. Daher sollen in der Lerneinheit 1.2 „Entwicklung von Informationssystemen“ des MuSofT-Projekts insbesondere Materialien zur Visualisierung von und zur praktischen Arbeit mit verschiedenen Modellierungmethoden, Verfahren, etc. aus dem Bereich des Datenbankentwurfs erstellt werden. Die vorhandenen Vorlesungsmaterialien sollen ebenfalls um multimediale Elemente erweitert werden, z.B. Videos, die bestimmte Aspekte der Entwicklung, etwa die Anforderungserfassung, verdeutlichen. Schließlich sollen die bestehenden und neu erstellten Materialien so aufbereitet werden, daß sie auch für ein ergänzendes Selbststudium verwendbar sind, etwa durch Bereitstellung über das Internet.

Der vorliegende Bericht gibt einen Überblick der ersten Projektphase, die sich im wesentlichen mit der Analyse bestehender Materialien, deren Strukturierung und der Auswahl von Werkzeugen für die Realisierung befasste. In Abschnitt 2 wird zunächst ein Überblick des Datenbankentwurfs und der in der Lerneinheit betrachteten Themen gegeben. Daran schließen sich in Abschnitt 3 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 4 gibt einen Überblick dieser Strukturierung und der Fallstudien. Die Erstellung und Bereitstellung der Materialien soll über eine Plattform erfolgen. Anforderungen an diese Plattform aus Sicht der Lerneinheit 1.2 werden in Abschnitt 5 angegeben. Abschnitt 6 gibt eine Übersicht der für die Erstellung der Materialien und deren Präsentation verwendeten Werkzeuge. Abschnitt 7 beschreibt kurz, wie die Einsetzbarkeit der Materialen anhand der Lehrveranstaltungen des Instituts evaluiert werden soll. Schließlich gibt Abschnitt 8 eine Zusammenfassung der Ergebnisse und einen Ausblick der für die nächste Projektphase geplanten Arbeiten.

# 2 Inhalt

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. Thematisch orientieren sich die zu erstellenden Materialien an den am Institut angebotenen Grundlagenvorlesungen „Datenbanken I“ und „Datenmanagement“. Erstere betrachtet wesentlich 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. Dieser Abschnitt gibt einen kurzen inhaltlichen Überblick der behandelten Thematiken. Materialien zu diesen Themen sollen innerhalb der Lerneinheit bereitgestellt werden.

Zunächst werden grundlegendende Konzepte der Datenbankterminologie und -technik, der

historischen Entwicklung von Datenbankmanagementsystemen (DBMS), Gründe für deren Einsatz, sowie Basis-Funktionen und Architekturen von DBMS dargestellt. Ziel ist die Motivation des Datenbankeinsatzes, d.h. welche Vorteile bietet dieser beispielsweise gegenüber der Datenspeicherung in Dateien und für welche Anwendungen ist deren Einsatz sinnvoll. 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. 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£nition mittels einer Datende£nitionssprache (DDL). Die Lerneinheit gibt 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 um-

gesetzt wurde, können Daten eingefügt und verändert werden. Weiterhin ist die De£nition von *Anfragen* oder *Sichten* möglich, um basierend auf den in der Datenbank gespeicherten Relationen, neue zu de£nieren. Zunächst werden der Einsatz spezieller Anfragesprachen motiviert und deren Vor- und Nachteile gegenüber „konventionellen“ Programmiersprachen aufgezeigt. Weiterhin erfolgt die Einführung der theoretischen 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 spezi£zieren 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 Datenbankanfragesprachen 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 De£nition von Integritätsbedingungen und Triggern, sowie die Vergabe von Rechten mittels SQL.

# 3 Didaktisches Konzept

## 3.1 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 ¤exibel sein, d.h. sie müssen derart strukturiert werden, daß eine Zusammenstellung von Materialien für eine bestimmte Zielgruppe möglich ist. Zunächst soll die Umgebung in der 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.

**Beru¤iche Quali£kation/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 P¤icht- bzw. Wahlp¤ichtfach im 3. oder 4. Semester des Grundstudium oder im 5. Semester des Hauptstudiums der Informatikstudiengänge angeboten.

## 3.2 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.

### 3.3 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.

## 4 Strukturkonzept

Der Inhalt der Lerneinheit wurde bereits im Abschnitt 2 vorgestellt, wobei bereits eine gewisse Gliederung in Themenkomplexe erfolgte, die aber für die De£nition 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. Anschließend erfolgt eine kurze Vorstellung der Fallstudien, die in der gesamten Lerneinheit verwendete werden.

### 4.1 Übersicht der Lernmodule

In Abbildung 1 sind die der Lerneinheit zugrunde liegenden Lernmodule dargestellt. Weiterhin wurden diese in gewissem Umfang weiter in Gruppenobjekte untergliedert. Auf eine vollständige Au¤istung 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 der Gruppenobjekte untereinander wurde ebenfalls verzichtet, da dies erst sinnvoll möglich ist, wenn deren Inhalte weitestgehend feststehen.

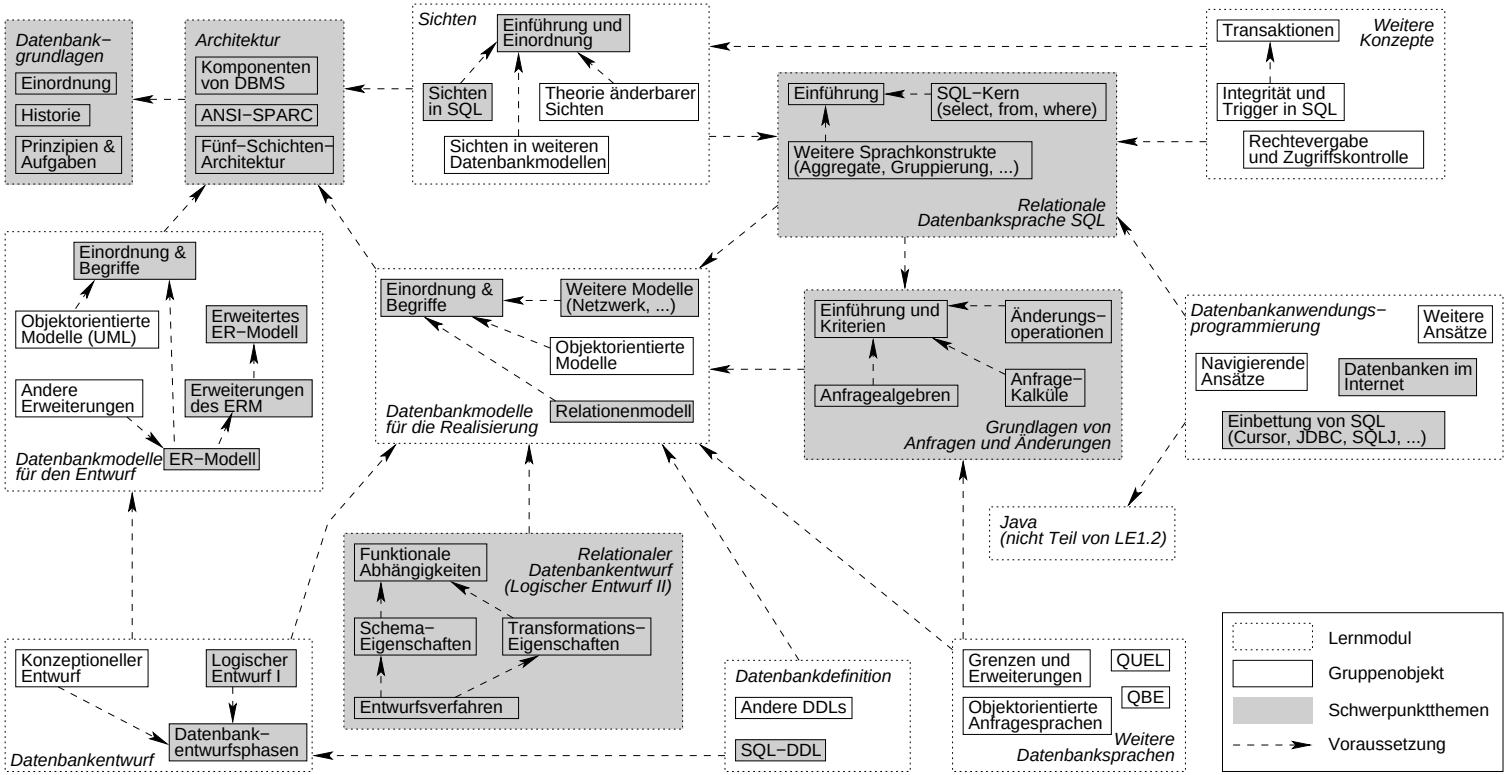



Abbildung 1: Struktur und Abhängigkeiten der Lernmodule und Gruppenobjekte

### 4.2 Fallbeispiele

Im Rahmen der Lerneinheiten sind derzeit zwei Fallbeispiele geplant. Zum einen eine Anwendung im Universitätsbereich [HS00] und zum anderen ein Web-Shop [HSS01].

**Universität:** Dieses Fallbeispiel betrachtet die Verwaltung von Lehrveranstaltungen einer Universität. Dabei werden die Beziehungen zwischen Lehrveranstaltungen, beteiligten Personen, Büchern etc. betrachtet. Das Fallbeispiel ist so angelegt, daß insbesondere die Konzepte der ER-Modellierung (inkl. Erweiterungen) möglichst vollständig enthalten sind.

**Web Shop:** In diesem Beispiel werden die Daten eines Online-Shop verwaltet, d.h. Waren, Kunden, Rechnungen etc. Ergänzend zum Universitätsbeispiel wird der Zugriff auf die Daten über Java-Anwendungen mittels JDBC, SQLJ etc. sowie über das Internet mittels PHP etc. eingegangen.

## 5 Lehr/Lernumgebung

Für die Verwaltung und Bereitstellung der verschiedenen Materialien soll im MuSofT-Projekt eine Plattform bereitgestellt werden. Dieser Abschnitt beschreibt die Anforderungen, die eine derartige Plattform aus Sicht der Lerneinheit 1.2 erfüllen soll. Die Lerneinheit stellt Materialien für verschiedene Bereiche der Lehre (Vorlesungen, Übungen und Selbststudium) bereit. In diesen Bereichen soll die Plattform sowohl die lehrende Person bei der Erstellung und Präsentation der Materialien, aber auch die lernende Person, z.B. beim weitergehenden/vertiefenden Selbststudium oder beim Lösen von Aufgaben, unterstützen. Daher sind folgende wesentliche Komponenten erforderlich:

**Autorenkomponente:** Diese Komponente soll primär Funktionalitäten für die Arbeit mit den Metadaten der Lernobjekte bereitstellen. Dazu gehört die Erstellung (Eingabe) der Metabeschreibungen. Die Erstellung der einzelnen Materialien wird mit externen Werkzeugen durchgeführt (vgl. Abschnitt 6). Es ist daher nicht notwendig, derartige Funktionalitäten zu integrieren. Wünschenswert ist die Möglichkeit, Werkzeuge entsprechend der zu bearbeitenden Materialien aufzurufen. Weitere wesentliche Funktionen sind die Suche basierend auf den Metadaten und die Unterstützung bei der Zusammenstellung von Lernobjekten zu Lernmodulen und -einheiten. Dabei soll u.a. die De£nition einer Reihenfolge für die Präsentation festgelegt werden können. Auch die Verwaltung von verschiedenen Versionen eines Lernobjekts sollte im Hinblick auf die heterogene Zielgruppe unterstützt werden.

**Präsentationskomponente:** Wie bei der Erstellung der Materialien sollen auch für die Präsentation der Materialien externe Werkzeuge eingesetzt werden. Die Aufgabe der Präsentationskomponente ist vorrangig die Bereitstellung/Zusammenstellung der konkreten Materialien, die beispielsweise für eine Veranstaltung benötigt werden, etwa alle Medienobjekte die Teil eines Lernmoduls sind. Weiterhin sollen, beispielsweise basierend auf den Metadaten, Informationen über die notwendige Software bereitgestellt werden. Für das Selbststudium soll im wesentlichen ein Browser für die Präsentation

verwendet werden. Die Präsentationskomponente muß in diesem Fall die Materialien serverseitig bereitstellen, wobei die Struktur weitestgehend basierend auf den Metadaten erstellt werden soll. Schließlich ist Möglichkeit zur Suche auf den Metadaten (und den Materialien) wünschenswert.

**Übungsumgebung:** Diese Komponente soll die Arbeit in betreuten und nicht betreuten Übungen unterstützen. Zum einen sind dafür Werkzeuge bereitzustellen, mit denen die Aufgaben bearbeitet werden können (vgl. dazu Abschnitt 6). Die Plattform muß in diesem Zusammenhang Funktionalitäten anbieten, diese Werkzeuge inkl. der Aufgaben bereitzustellen und die erstellten Lösungen zu verwalten. Letzteres soll vorwiegend personengebunden erfolgen. In diesem Zusammenhang sind zumindest grundlegende Funktionen der Nutzerverwaltung erforderlich. Eine allgemeine Publizierung der Lösungen soll aber auch möglich sein (Präsentationskomponente).

**Datenhaltungskomponente:** Dient der Verwaltung der anfallenden Daten, z.B. die beschreibenden Metadaten der Lernobjekte, die Daten der Nutzerverwaltung, die erstellten Materialien, etc. Zum einen sollten die Objekte oder zumindest die Metadaten zentral gespeichert werden. Zum anderen ist aber auch die Erstellung von Kopien zu unterstützen.

# 6 Technische Realisierung

In Abschnitt 3 wurde bereits erwähnt, daß die Lerneinheit insbesondere Vorlesungen und die praktische Arbeit in Übungen unterstützen soll. Daher wurde untersucht, welche Werkzeuge für die Erstellung der Materialien eingesetzt werden sollen und wie die Materialien in den verschiedenen Zielveranstaltungen präsentiert werden:

**Übung:** Für Übungen ist die Erstellung von Werkzeugen für die verschiedenen Phasen des Datenbankentwurfs geplant. In der ersten Projektphase hat eine entsprechende Evaluierung von Werkzeugen stattgefunden. Dabei hat sich herausgestellt, daß die überwiegende Anzahl verfügbarer Werkzeuge nicht einsetzbar ist, da u.a. keine hinreichenden Schnittstellen für einen Zugriff auf die Funktionalitäten zur Verfügung stehen. Dies ist aber erforderlich, um die Werkzeuge beispielsweise an verwendete Notationen anzupassen oder Funktionen zu deaktivieren oder zusätzliche zu integrieren, z.B. zur Erstellung von Visualisierungen. Daher wurde entschieden, entsprechende Werkzeuge mittels Java selbst zu implementieren. Teilweise sollen dabei existierende Werkzeuge als Grundlage verwendet werden, etwa `Together` (`www.togethersoft.de`) für die Modellierung.

**Vorlesung:** Für die Vorlesung werden weiterhin die vorhandenen Präsentationsfolien verwendet. Im Rahmen des Projektes soll aber eine Aufarbeitung erfolgen, so daß eine Online-Präsentation erfolgen kann. Weiterhin ist die Ergänzung um neue Themen, etwa die Anwendungsentwicklung vorgesehen. Für die Erstellung der Folien wird LaTeX, für die Präsentation PDF verwendet. Die für Übungszwecke erstellten Werkzeuge sollen auch die Generierung von Animationen unterstützen, die in Vorlesungen einsetzbar sind. Beispielsweise soll ein Werkzeug für die Visualisierung von Tabellenstrukturen innerhalb der Datenbank angewendet werden.

**Selbststudium:** Für das Selbststudium werden zum einen die Materialien der Vorlesung in einer für das WWW aufbereiteten Form bereitgestellt. Der vollständige Inhalt der Veranstaltung soll aber nicht in Form von Texten angeboten werden. Vielmehr ist geplant, die Materialien, Folien, Animationen, Video, etc. thematisch und chronologisch zu ordnen und über eine WWW-Seite zur Verfügung zu stellen. Ein erster Prototyp für Folien, Video- und Audioaufzeichnungen wurde bereits erstellt und steht zur Verfügung (`http://wwwiti.cs.uni-magdeburg.de/iti_db/lehre/db1/index.html`). Zusätzlich werden auch die für Übungszwecke entwickelten Werkzeuge bereitgestellt, um eine weitere Bearbeitung der Aufgaben zu gestatten.

# 7 Evaluierung

Die Evaluierung der Lerneinheit wird zunächst im Rahmen der Magdeburger Datenbankvorlesungen erfolgen. Zu diesem Zweck sollen die erstellten Materialien in einigen ausgewählten Übungen bereitgestellt werden (3./4. Semester), um insbesondere die Akzeptanz der erstellten Werkzeuge zu untersuchen. Soweit es die technischen Gegebenheiten zulassen sollen im Sommersemester 2002 und im Wintersemester 02/03 erste Materialien, z.B. Visualisierungen in den jeweils angebotenen Datenbankvorlesungen präsentiert werden. Weiterhin sollen die Materialien anderen Projektpartnern zur Verfügung gestellt werden, damit der Einsatz auch an anderen Universitäten und in anderen Studienfächern erprobt werden kann. Eine endgültige Evaluierung der Lerneinheit soll dann im Sommersemester 2003 erfolgen. Um die Akzeptanz seitens der Studenten festzustellen sind Befragungen, z.B. mittels Fragebogen, in den entsprechenden Übungen zum Ende des Wintersemesters 02/03 geplant.

# 8 Zusammenfassung

In der Lerneinheit 1.2 des MuSofT-Projekts sollen Materialien und Werkzeuge entwickelt werden, die die Lehre im Gebiet des Datenbankentwurfs unterstützen. Dazu wurden zunächst Untersuchungen hinsichtlich der zu bearbeitenden Themen und des Zielpublikums durchgeführt. Im Hinblick auf eine möglichst umfassende Eingliederung in das Gesamtprojekt, welches sich mit der Lehre in der Softwaretechnik im allgemeinen befaßt, wurde eine inhaltliche Orientierung auf Teilbereiche des Datenbankentwurfsprozesses beschlossen. Insbesondere sollen die Phasen des konzeptionellen und des logischen Entwurfs betrachtet werden, in denen sich verschiedene Überschneidungen mit anderen Teilprojekten ergeben, etwa zum Thema Anforderungserfassung oder der Visualisierung von Algorithmen und Verfahren. In der traditionellen Lehre werden insbesondere praktische Fähigkeiten, z.B. der Einsatz von Verfahren und Algorithmen beim Entwurf, nur unzureichend vermittelt, da die zur Verfügung stehende Zeit für Übungen und Praktika nicht ausreichend ist. Im Rahmen der Lerneinheit 1.2 sollen daher insbesondere Werkzeuge für die konzeptionelle Modellierung, den logischen Entwurf, den relationalen Entwurf und den Einsatz von Datenbankanfragesprachen entwickelt werden. Außerdem soll durch den Einsatz von Visualisierungen, etwa von Verfahren des relationalen Datenbankentwurfs, deren Verständnis gefördert werden. Durch die Bereitstellung der Materialien und Werkzeuge über das Internet soll außerdem eine Vertiefung im Selbststudium

gefördert werden. Die weiteren Arbeiten im Projekt werden sich zum einen mit der Realisierung der Werkzeuge befassen. Weiterhin werden bestehende Materialien, z.B. Präsentationsfolien angepaßt und ergänzt sowie entsprechend der vorgesehenen Struktur gegliedert und mit Metadaten versehen.

## Literatur


[Che76] CHEN, P. P.: *The Entity-Relationship Model – Towards a Uni£ed 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. 1999.

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

[HS00] HEUER, ANDREAS und GUNTER SAAKE: *Datenbanken: Konzepte und Sprachen*. Informatik Lehrbuch-Reihe. MITP-Verlag, Bonn, 2. Au¤age Auflage, 2000.

[HSS01] HEUER, ANDREAS, GUNTER SAAKE und KAI-UWE SATTLER: *Datenbanken kompakt*. mitp-Verlag, Bonn, 2001.

# MuSofT-Ergebnis-Bericht der Teilprojekte 1.3 und 2.4

Olaf Scheel, Johannes Magenheim

Dieser Text enthält Informationen über Lerneinheit 1.3: *Softwareengineering in der Informatiklehrerausbildung* und Lerneinheit 2.4: *Dekonstruktion von Softwaresystemen* zum Jahresabschluss 2001.

## Inhaltsverzeichnis



## 1 Einleitung

Im folgenden werden die zwei Lerneinheiten der *AG Didaktik der Informatik Paderborn - Dekonstruktion von Softwaresystemen* und *Softwareengineering in der Informatiklehrerausbildung* - auf dem Stand Dezember 2001 näher beschrieben.

Beide Lerneinheiten sollen innerhalb der Informatiklehrerausbildung an der Universität Paderborn eingesetzt werden. Zum Teil direkt in der Vorlesung bzw. in Seminaren, zum Teil aber auch den Studierenden veranstaltungsbegleitend für die Nacharbeitung des Präsenzteils zur Verfügung stehen.

Für beide Einsatzszenarien soll eine Erkundungsumgebung bereitgestellt werden, in der der Dozent bzw. Studierende selbstständig frei durch die bereitgestellten Materialien navigieren kann, um im Sinne gemäßigter konstruktivistischer Lerntheorien Wissen aufzunehmen.

Innerhalb der Lerneinheiten werden hierzu zwei Fallstudien verwendet: Zum einen die eines kleinen Warenwirtschaftssystems, nämlich eines Schulkiosks, zum zweiten eine Simulation eines Hochregallagers - unter anderem mit einem Modell in Lego-Mindstorms. Die Software für den Schulkiosk existiert bereits und wird seit einiger Zeit von uns in Seminaren eingesetzt. Sie soll innerhalb des MuSofT-Projektes mäßig erweitert und verfeinert werden, aber vor allem um Materialien, die das soziotechnische Umfeld und die zugrunde liegenden Design-/Entwurfsentscheidungen zeigen, ergänzt werden. Das Modell des Hochregallagers mit verschiedenen Zusatzmodulen wird innerhalb des Projektes neu entwickelt. Die Idee beruht darauf, dass der *programmierbare Legobaustein*, der RCX, auch über eine Java-Firmware gesteuert werden kann, so dass eine Programmierung in Java möglich ist. (Der frühere Ansatz mit Fischertechnik wurde von uns mittlerweile verworfen.)

# 2 Inhalt

## 2.1 Lerneinheit 1.3

In der Lerneinheit 1.3 soll ein Modell eines Hochregallager mit verschiedenen Zusatzeinheiten in Lego-Mindstorms zum Einsatz kommen. An diesem Beispiel werden die Konzepte des *soziotechnischen Informatiksystems* und der *Systemorientierten Didaktik* erläutert. Konkret handelt es sich dabei um das Modell einer entsprechenden Anlage aus Lego inklusive dem informatischen Modell (sprich: Steuerungssoftware) und verschiedener Dokumente, die Anwendungsfälle (auch als Videos), UML-Diagramme, Entwurfsalternativen, CRC-Karten usw. zeigen. Die Dokumente werden wie auch in der Lerneinheit 2.4 zu einer Erkundungsumgebung arrangiert. Da das reale Modell nicht jedem/immer zur Verfügung gestellt werden kann, ist es zusätzlich notwendig, eine entsprechende gra£sche Simulation zu erstellen.

Die wichtigen Konzepte dieser Lerneinheit werden überwiegend über die Arbeit an Texten - d.h. lesen, diskutieren, zusammenfassen, wiedergeben - vermittelt werden. Dies soll aber nicht heißen, dass die Plattform in diesen Fällen einen einfache Textsammlung bereitstellt, sondern ebenso Zusammenfassungen, Gegenüberstellungen und Vergleiche verschiedener Ansätze. Diese sollen auch gra£sch illustriert werden.

**Modul 1: Soziotechnisches Informatiksystem**

Innerhalb dieses Moduls werden die verschiedenen Typen von Informatiksystemen vorgestellt. (Einzelplatz, Mehrbenutzer, eingebettete Systeme) Jedes System wird sowohl abstrakt (durch Texte und Gra£ken) charakterisiert, als auch durch konkrete Beispiele (Videos) beschrieben.

Ausgehend von diesen eher nach technischen bzw. funktionalen Aspekten kategorisierten Systemen wird der Begriff des *soziotechnischen Informatiksystems* eingeführt. Zu einem solchen System gehört in Anlehnung an M. Syrbe und G. Ropohl nicht nur die Hard- und Software sondern auch das soziotechnische Umfeld, das durch die mit dem System interagierenden Personen gebildet wird. Ohne hier auf das Konzept näher einzugehen, kann man bei der Analyse solcher System dann zwischen der technischen Repräsentationsebene (Maschinen) der Kommunikationsebene (Gruppe von Menschen) und der Wissensebene (subjektive Sicht des einzelnen) unterscheiden.

**Modul 2: Systemorientierte Didaktik**

Innerhalb dieses Moduls wird das didaktische Grundkonzept der AG Didaktik der Informatik Paderborn, nämlich der Systemorientierten Didaktik der Informatik, beschrieben. Dieses Konzept ist eng verbunden mit dem im vorangegangenen Modul behandelten Begriff des Informatiksystems und dem Konzept der Dekonstruktion, welches in der LE 2.4 erläutert wird, und soll hier zunächst nicht näher beschrieben werden.

Zum Einsatz kommen hier vor allem verschiedene Textdokumente, die Bezüge zu Lerntheorien, Systemtheorie, Technikwissenschaft, Medienwissenschaft und vor allem auch zur Fachwissenschaft der Informatik herstellen. Aus den einzelnen Bezugswissenschaften ergeben sich Anforderungen an die Systemorientierte Didaktik, die anhand von Beispielen mit der Fallstudie verknüpft werden. Dieses Modul wird als erstes in dieser Lerneinheit implementiert und im laufenden Wintersemester erstmalig erprobt.

**Modul 3: Softwaretechnik im Überblick**

Innerhalb dieses Moduls werden diejenigen Teilbereiche der Softwaretechnik eingeführt, die in der Lehrerausbildung eine Rolle spielen. Hier kommen u.U. von den anderen Projektpartnern entwickelte Lernobjekte/-module zum Einsatz. Schulgeeignete Werkzeuge und Methoden sollen vorgestellt werden: Mr. Dobs, CRC-Karten, Object-Game.

**Modul 4: Methoden der Softwaretechnik in den didaktischen Konzepten des Informatikunterrichtes**

Die im vorangegangenen Modul vorgestellten Methoden werden (abhängig von verschiedenen didaktischen Ansätzen) zu Vorgehensmodellen der Softwareentwicklung für den Unterricht zusammengeführt. Diese Vorgehensmodelle werden verglichen, auch mit den in der professionellen Softwareentwicklung angewendeten.

**Modul 5: Berufsbilder in der Softwareentwicklung**

Dieses Modul hängt nur locker mit den anderen in dieser Lerneinheit zusammen, soll aber dadurch, dass die Fallstudie auch in diesem Modul verwendet wird, näher mit den Modulen 1- 4 verknüpft werden. Themen sind hier Frauen in der Informatik, Anforderungen an den Beruf, aber auch Klischeevorstellungen. Zum Einsatz kommen hier Videoaufzeichnungen von Tätigkeiten und Interviews aber auch statistische Daten.

## 2.2 Lerneinheit 2.4

In der Lerneinheit 2.4 soll der oben schon erwähnte Schulkiosk zum Einsatz kommen. Das existierende Softwaresystem wird innerhalb der Einheit einem *Dekonstruktionsprozess* unterworfen.

**Modul 1: Anwendungsfälle und Softwarefunktionen**

Videos visualisieren verschiedene Anwendungsfälle, aber auch Entwurfsentscheidungen, Einsatzszenarien und Konsequenzen. Der Zusammenhang mit dem Softwareprodukt wird mit Hilfe eines gra£schen Debuggers und einem Object-Browser (z.B. Mr Dobs) analysiert. Durch Letzteren wird die Analyse und Manipulation von Java-Objekten zur Laufzeit möglich.

**Modul 2: Reengineering**

Dieses Modul beschäftigt sich mit der Skalierbarkeit, Funktionserweiterung und Anpassung des Schulkiosks. Bildschirmvideos und andere Dokumente beschreiben verschiedene Entwurfs- und Designentscheidungen, aber auch Einsatzszenarien. Die Erweiterungen werden mit Hilfe einer Entwicklungsumgebung (*together*) ausgeführt.

**Modul 3: Praxisevaluation eines Informatiksystems**

Die Auswirkungen von Informationssystemen auf Arbeitsumgebungen werden am Beispiel untersucht. Veränderungen von Arbeitsplätzen, Rollen, Funktionen, Aufgaben und Tätigkeiten werden aufgezeigt.

**Modul 4: Dekonstruktion impliziter Sichtweisen und Interessen**

Wie wirken sich unterschiedliche Interessen und Sichtweisen von Entwicklern, Kunden, Nutzern und Auftraggebern auf die Gestaltung eines Informatiksystems aus? Welche Interessen werden in der Software sichtbar? Am Beispiel wird dies durch den Vergleich von zwei Designalternativen verdeutlicht.

**Modul 5: Wechselwirkungen zwischen komplexen Informatiksystemen und Gesellschaft**

Im Modul 3 wurden die Auswirkungen auf das nahe Umfeld eines Informatiksystems untersucht. In diesem Modul sollen gesamtgesellschaftliche Auswirkungen verdeutlicht werden. Zu diesem Zweck wird ein E-Commerce-Szenario für den Schulkiosk entwickelt und auf die aktuellen Entwicklungen im Internet bezogen.

# 3 Didaktisches Konzept

## 3.1 Einsatzszenario

Wie oben schon erwähnt sollen die von uns erstellten Lerneinheiten in unseren Veranstaltungen *Didaktik der Informatik II* und *Modellieren im Informatikunterricht* eingesetzt werden. Der Schwerpunkt wird hier auf dem Seminar *Modellieren im IU* liegen, in der Didaktik-Vorlesung werden nur Teile der Lerneinheit 1.3 eingesetzt werden. Da unsere Veranstaltungen i.d.R. Seminare sind, oder wegen relativ geringer Studierenden-Zahlen auch sonst teilweise Seminar-Charakter haben, ist für unsere Lerneinheiten eine Aufbereitung für einen reinen Dozenten-Vortrag innerhalb einer Vorlesung wenig interessant. Wir favorisieren eine Erkundungsumgebung genannte Ansammlung von Materialien zu bestimmten Themenkomplexen, durch die der Nutzer (Student oder Dozent) frei navigieren, eine eigenständige Auswahl und ein individuelles Lern-Arrangement bilden kann. Dies wird nach der bisherigen Planung der Integrationsplattform bedeuten, dass viele unserer Materialien für den Übungsplattform genannten Teil des MuSofT-Servers aufbereitet werden.

## 3.2 Lernvorrausetzungen

Die Teilnehmer der genannten Veranstaltungen be£nden sich im (Lehramt-) Hauptstudium, sie haben sowohl einführende Veranstaltungen im Bereich der Softwareentwicklung und Modellierung (*Softwareentwicklung I*, *Softwareentwicklung II*, *Modellierung*) als auch die einführende Veranstaltung in die Didaktik der Informatik (*Didaktik der Informatik I*) bereits besucht. Das innerhalb dieser Veranstaltungen vermittelte Wissen wird vorausgesetzt, wobei davon ausgegangen wird, dass sowohl keine tiefer gehenden Programmier- und Entwurfserfahrungen als auch wenig Kenntnisse über das Verhältnis professioneller Softwareentwickung zu Softwareentwicklung in der Schule vorhanden sind. (Z.B. wegen der noch fehlenden Schulpraktischen Studien.)

## 3.3 Lernziele

Die Lernziele der beiden Lerneinheiten richten sich zunächst einmal nach den Veranstaltungen in die diese eingebettet werden. Da Lerneinheiten im Sinne des Projektes bezüglich ihres Themas in sich abgeschlossen sind, gehen die Lernziele aber über konkrete Ziele der einzelnen Veranstaltungen hinaus. Im folgenden werden nur kognitive Lernziele genannt, wobei immer auch Handlungskompetenz erworben werden soll.

### 3.3.1 Lerneinheit 1.3

- Verständnis des Begriffs *Soziotechnisches Informatiksystem*
- Kenntnisse über die *Systemorientierte Didaktik*
- Fähigkeit über grundlegende Werkzeuge, Methoden und Konzepte der Softwaretechnik (hier werden u.U. Module der Projektpartner verwendet) anzuwenden. Speziell CRC-Karten, einfache Klassen- und Sequenzdiagramme.
- Kenntnis eines für die Schulinformatik bezüglich der Systemorientierten Didaktik geeigneten Vorgehensmodells. In welcher Reihenfolge sollte man Konzepte vermitteln? Wie £ndet man geeignete Beispiele für den Unterricht?

### 3.3.2 Lerneinheit 2.4

- Fähigkeit, ein Softwareprodukt zu *dekonsturieren*.
- Fähigkeit, einen gra£schen Debugger gezielt einzusetzen.
- Kenntnis über die Auswahl geeigneter Anwendungsfälle.
- Fähigkeit, aus Softwaresystemen auf implizite Intentionen der Entwickler zu schließen
- Fähigkeit, gesellschaftliche Auswirkungen vorauszusehen und zu beurteilen

## 3.4 Lernformen

Es soll sowohl der *Frontalunterricht* (in Vorlesungen), das *Unterrichtsgespräch* (in Seminaren und Übungen) als auch das Selbststudium (in der Nachbereitung und in Übungen) durch Materialien unterstützt werden. Aus diesem Grund scheidet eine reine Präsentation an einen passiven Rezipienten aus. Die einzelnen Lernobjekte sollen hingegen in einer *Erkundungsumgebung* miteinander vernetzter, zum Teil interaktiver, Dokumente angeboten werden.

Das Konzept der beiden von uns zu verwirklichenden Lerneinheiten ist ein handlungsorientiertes. Aus diesem Grund werden von uns für jedes Modul entsprechende Fragestellungen in Form von Erkundungsaufträgen gesucht. Bei einzelnen Modulen (z.B. 1.4 Modul 2) ist dies nicht trivial, da es sich um abstrakte Konzepte handelt, die vorwiegend anhand von Texten erschlossen werden müssen. Somit eignen sich nicht alle Module gleichermaßen als Erkundungsumgebung, einzelne eher Präsentation von Materialien. Es wird aber angestrebt, dass

die Eignung der Module für die verschiedenen Arbeitsweisen möglichst homogen ist. Es erscheint uns außerdem zweckmäßig, die *Granularität* von Modulen zu verfeinern, da nur eine hohe Granularität einen Einsatz der Materialien in verschiedenen Lernkontexten ermöglichen wird.

# 4 Strukturkonzept

## 4.1 Lerneinheit 1.3

### 4.1.1 Modul 1: Soziotechnisches Informatiksystem

- Typen von Informatiksystemen
  - Einzelplatz
  - Mehrbenutzer
  - Eingebettete Systeme
- Theoretische Bezüge
  - Allgemeine Technikphilosophie
  - Systemtheorie
  - Theorie technischer Systeme
  - Systemgestaltung, Softwareentwicklung
- Zu jedem System werden Dokumente angeboten, die folgende Aspekte beinhalten:
  - Sicht auf die Hardware bzw. den technisch apparativen Teil des Mediums
  - Sicht auf die Benutzungsober¤äche (GUI) und die damit induzierten sozialen Interaktionen bzw. die interaktive Seite des Mediums
  - Sicht auf die Vernetzung der Informatiksysteme bzw. die kommunikativen Aspekte der computerbasierten Medien
  - Sicht auf die Software und die damit während der Softwareentwicklung realisierten Operationalisierungen von antizipierten sozialen Handlungen im Kontext des Informatiksystems
  - Sicht auf die Software im Sinne der dort realisierten informatischen Ideen und Konzepte der Softwaretechnik einschließlich der verwendeten Algorithmen
  - Sicht auf die Software im Hinblick auf die Semiotik und die verwendeten symbolischen Repräsentationen
  - Sicht auf Software im Hinblick auf von ihr erzeugte virtuelle Realität
  - Sicht auf Software im Hinblick auf die Datenbasis, deren Struktur, Inhalt und Repräsentationsformen.

#### 4.1.2 Modul 2: Systemorientierte Didaktik

- Exkurs Bildungstheoretische Didaktik
- Exkurs Lerntheorien
- Systemorientierte Didaktik
- Beispiele
- Java und Systemorientierte Didaktik

#### 4.1.3 Modul 3: Softwaretechnik im Überblick

- Überblick: Softwaretechnik in der Lehrerausbildung
  - Planungsphase
  - De£nitionsphase
  - Entwurf
  - Implementierung
  - Abnahme und Einführung
  - Wartung
- Schulgeeignete Werkzeuge
  - Mr. Dobs
  - CRC-Karten
  - Object-Game

#### 4.1.4 Modul 4: Methoden der Softwaretechnik in den didaktischen Konzepten des Informatikunterrichtes

- Vorgehensmodelle für die Entwicklung von Software im Unterricht
- Vergleich Vorgehensmodelle im IU und in der professionellen SE

#### 4.1.5 Modul 5: Berufsbilder in der Softwareentwicklung

- Typische Berufsbilder
- Frauen in informationstechnischen Berufen
- Statistische Daten

## 4.2 Lerneinheit 2.4

### 4.2.1 Modul 1: Anwendungsfälle und Softwarefunktionen

- Anwendungsfälle
- Use Cases
- Akteure
- Diagramme
- Programmanalyse mit einem Object Browser
- Einsatzszenarien
- Konsequenzen

### 4.2.2 Modul 2: Reengineering

- Skalierung
- Anpassung

### 4.2.3 Modul 3: Praxisevaluation eines Informatiksystems

- Evaluationsmethoden
- Auswirkungen von Informatiksystemen
- Beispiele

### 4.2.4 Modul 4: Dekonstruktion impliziter Sichtweisen und Interessen

- Sichtweisen und Interessen
- Design-Alternativen
- Beispiele

### 4.2.5 Modul 5: Wechselwirkungen zwischen komplexen Informatiksystemen und Gesellschaft

- Szenario: E-Commerce
- Gesellschaftliche Konsequenzen
- Beispiele

# 5 Lehr/Lernumgebung

Aus Sicht der *AG Didaktik der Informatik Paderborn* sollte die Plattform nicht nur präsentieren, sondern vor allem arrangieren. Die Studierenden sollten selbstständig im veranstaltungsbegleitenden Kontext mit den Materialien arbeiten können. Aus diesem Grund ist für uns vor allem der Teil der MuSofT-Plattform interessant, der vom KT4 bisher *Übungsplattform* genannt wird. Inwieweit dort unser Konzept einer *Erkundungsumgebung* verwirklicht werden kann, ist fraglich, sollte aber möglichst bald an einem konkreten Beispiel evaluiert werden. Die von uns erstellten Materialien werden im Wesentlichen mit Hilfe eines Webbrowser mit Standard-Plugins vollständig nutzbar sein. Durch die Plattform zu unterstützende Dateiformate sind hier: HTML, PDF, Java, Javascript, GIF/PNG/JPEG, MPEG und Flash. Die Plattform sollte (nach Maßgabe des Dozenten) selbstständig aus den vorhandenen Lernobjekten Module und Einheiten zusammensetzen können und diese in einem einheitlichen Layout anbieten können. Außerdem sollte ein Mini-Redaktionssystem integriert sein, d.h. nicht nur Dozenten, sondern auch Studenten sollten sich an der Plattform anmelden können. Studenten sollte dann eine andere Sicht auf die Materialien präsentiert werden als dem Dozenten der jeweiligen Veranstaltung, der dann dafür zuständig wäre, Materialien und Funktionen aus dem Pool für den Studierendenbereich freizuschalten. Die Präsentation der Lerneinheiten sollte auch zum Selbststudium der Studierenden nicht in sequentieller Form geschehen, sondern eher in Form einer offenen Erkundungsumgebung in der der Student sich selbst die benötigten Materialien zusammenstellen kann. Die Plattform sollte ihn dabei unterstützen.

Die Lernobjekte, oder Teile von Lernobjekten, sollten untereinander durch typisierte Verweise verknüpft werden können, die auch die synchrone Präsentation mehrerer Objekte erlauben (z.B. zwei Videos parallel, Use-Case zum Quelltext oder UMLDiagramm...).

Es sollte eine systemweite Suche nach Schlagworten, aber auch nach Medientypen und Metadaten möglich sein. Wir unterstützen dabei das von KT2 und KT4 mit unserer Beteiligung erarbeitete Metadatenmodell, welches im Wesentlichen ein auf MuSofT-Bedürfnisse zugeschnittener LOM-Dialekt sein wird.

Zusätzlich zur Integrationsplattform werden aber sowohl auf Nutzer, wie auch auf Autorenseite verschiedene Entwicklungs- und Analysewerkzeuge benötigt. (Mr. Dobs, together//J, ...)

# 6 Technische Realisierung

In dieser Phase der Projektdurchführung steht innerhalb der *AG Didaktik der Informatik Paderborn* zunächst die Entwicklung des Hochregallagers in Lego-Mindstorms im Vordergrund. Nach Fertigstellung werden die verschiedenen oben beschriebenen Dokumente rund um dieses System erstellt werden.

Parallel werden Materialsammlungen für die anderen Teilmodule angelegt: Sammlungen von Basistexten, Einführenden Dokumenten und Textzusammenfassungen werden als Materialpool innerhalb der zu schaffenden Erkundungsumgebung angelegt.

Als erstes Teilmodul wird das Modul 2 der Lerneinheit 1.3 implementiert und zu Beginn des neuen Jahres erstmalig erprobt. In diesem Zusammenhang wird auch das Arrangement von Dokumenten evaluiert werden, so dass ein Konzept für die Erarbeitung der Dokumente

der anderen Module erstellt werden kann.

# 7 Evaluierung

Ein genaues Evaluationskonzept ist bisher noch nicht erarbeitet worden. Geplant ist aber, dass zu jedem von uns eingesetzten Modul eine Eingangs- und Schlussbefragung durchgeführt wird, die den Zusatz-Nutzen der eingesetzten Materialien und das Arrangement auf der Plattform evaluiert. Zu diesem Zweck wird eine von uns schon bei früheren Veranstaltungen eingesetzte webgestützte Befragungssoftware (grafstat) eingesetzt werden.

# Lerneinheit 2.1 – Software-Architektur

Jörg Pleumann

Dieser Bericht gibt den aktuellen Stand des Konzeptes der MuSofT-Lerneinheit 2.1 wieder, die sich mit dem Thema „Software-Architektur“ beschäftigt. Er faßt die Ergebnisse der ersten neun Monate des Projektes sowie die weitere Vorgehensweise zusammen.

## Inhaltsverzeichnis



## 1 Einleitung

Dieser Bericht gibt den aktuellen Stand der MuSofT-Lerneinheit 2.1 wieder, die sich mit dem Thema „Software-Architektur“ beschäftigt.

Ausgangspunkt der Lerneinheit und damit maßgebend für dieses Konzept ist 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 eher theoretische Vermittlung der Eigenschaften bestimmter Architekturen oder architektureller Stile in Vorlesungen befähigt die Studierenden nicht automatisch zum Entwurf komplexer Software-Systeme – genau wie das 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 in diesem Bericht beschriebene Lerneinheit soll den Studierenden helfen, die gewünschten praktischen Erfahrungen mit Architekturen zu sammeln. Dazu stellt sie als Kernelement ein spezielles Werkzeug bereit, das zum Bearbeiten von Software-Architekturen mit graphischen Mitteln dient – quasi einen Architektur-Editor, der in besonderem Maße für die Lehre in der Softwaretechnik geeignet ist. Der Editor kann gleichermaßen in Übungen und Praktika Verwendung £nden und bietet die nötige Infrastruktur, den Studierenden Aufgaben im Bereich des Entwurfs von Software-Architekturen zu stellen, die anschließend durch eine automatische (Vor-) Korrektur oder einen menschlichen Tutor bewertet werden können. Zudem können die graphischen Möglichkeiten des Editors zur komfortablen und übersichtlichen Visualisierung von Architekturen genutzt werden, so daß das Werkzeug auch im Vorlesungsbetrieb einsetzbar ist und dort Tafelanschrieb oder Overhead-Folien ergänzen oder gar ersetzen kann.

# 2 Inhalt

Dieser Abschnitt gibt einen Überblick über den geplanten Inhalt der Lerneinheit. Der Inhalt gliedert sich grob in sechs verschiedene Teilbereiche – Grundlagen, Notation, Architekturstile, Fallbeispiele, formale Methoden und Übungen – , denen jeweils einer der folgenden Unterabschnitte gewidmet ist. Während sich die ersten fünf Teilbereiche relativ klar voneinander abgegrenzen lassen und auch in dieser Weise Eingang in die Lerneinheit £nden sollen, ist das für den Bereich der Aufgaben nicht der Fall. Diese werden entweder – je nach Bedarf – in die anderen Teilbereiche aufgenommen oder komplett in den Übungsbetrieb von MuSofT ausgegliedert.

Jeder Teilbereich deckt sich in etwa mit dem, was innerhalb von MuSofT als Lernmodul bezeichnet wird. Da die Lerneinheit jedoch eher als geschlossene Lernumgebung und weniger als lose Sammlung von Vorlesungsmaterial geplant ist und auch keiner der inhaltlichen Teilbereiche ohne den Editor existieren kann, läßt sich eine strikte Trennung in unabhängige Module nicht wirklich vornehmen.

## 2.1 Grundlagen

Auch wenn die Unterstützung des Selbststudiums nicht das Hauptziel von MuSofT ist, scheint es doch sinnvoll, gewisse Grundbegriffe und -ideen aus dem Bereich der Software-Architektur mit in die Lerneinheit aufzunehmen. Diese Grundbegriffe sollen den Studierenden in komprimierter Form als eine Art Referenz oder „Online-Hilfe“ zur Verfügung stehen und im Einzelfall zum schnellen Nachschlagen, z.B. während der Bearbeitung von Aufgaben, genutzt werden können. Sie sollen außerdem dazu beitragen, die Lerneinheit etwas „abzurunden“ und sie nicht als bloße Sammlung von Beispielen und Aufgaben erscheinen zu lassen. Dieser Teil der Lerneinheit soll und kann jedoch keinesfalls eine Vorlesung, ein Lehrbuch oder ein Skript ersetzen und wird dementsprechend dimensioniert sein.

Insgesamt enthält der Teilbereich etwa das, was auch die einleitenden Kapitel verschiedener Bücher [BMR+96, SG95] oder Artikel [GS94, HHK+96, PW92, SW99] zum Thema Software-Architektur enthalten, jedoch in komprimierter Form. Bei der Wahl von Begriffen

sollen soweit wie möglich die Vorgaben aus dem IEEE Standard für Software-Architekturen [MEH01] Verwendung £nden.

## 2.2 Notation

Der zweite Teilbereich führt die Grundelemente einer Architekturbeschreibung und die zugehörige graphische Notation innerhalb des Editors ein. Er dient den Studierenden somit gleichzeitig als Referenz zur Verwendung des Editors während der praktischen Arbeit an Aufgaben. Als Notation wird eine Erweiterung der UML verwendet, die zur Beschreibung von Architekturen geeignet ist. Diese Erweiterung geht unter anderem auf Ideen aus Real-Time Object-Oriented Modeling (ROOM) zurück [SGW94] und wird Eingang in die Version 2.0 der UML £nden.

## 2.3 Architekturstile

Der dritte Teilbereich wird den Sinn von Architekturstilen erläutern und einige der bekannteren Stile vorstellen. Zu jedem Stil werden das zulässige Vokabular (Kapseln, Konnektoren etc.) sowie die erlaubten strukturellen Muster angegeben. Es wird die generelle Arbeitsweise erläutert, und es werden Invarianten, Vor- und Nachteile, Beispielanwendungen sowie Spezialisierungen aufgezeigt. Damit gleicht der Abschnitt ein wenig den üblichen Katalogen von Entwurfsmustern [BMR$^+$96, GHJV96], mit dem Unterschied, daß sich der Architektur-Katalog auf einer höheren Abstraktionsebene bewegt.

## 2.4 Fallstudien

Dieser Teilbereich stellt einige ausgewählte Architekturen in Fallstudien vor. Sinn des Abschnitts ist es zum einen, den Studierenden ein Gefühl für die Verwendung der im vorangehenden Abschnitt eingeführten Architekturstile zu vermitteln sowie ihnen die Vor- und Nachteile der Verwendung eines Stils bei konkreten Problemstellungen vor Augen zu führen. Zum anderen soll den Studierenden verdeutlicht werden, daß Architekturstile selten in ihrer puren Form auftreten, tatsächliche Architekturen oft Kombinationen von Stilen enthalten oder gar mehrere disjunkte architekturelle Sichten auf ein System möglich sind.

## 2.5 Formale Methoden

Auf dem Gebiet der Software-Architektur ist im Laufe der Jahre eine Reihe von formalen Sprachen und Spezi£kationsansätzen mit unterschiedlichen Schwerpunkten entwickelt worden. Zu den bekannteren gehören *UniCon* und *Wright* [SG95], *Rapide* [LKA$^+$95] und die *Chemical Abstract Machine (CHAM)* [IW95].

Mit der Wahl der UML und der bereits erwähnten Erweiterungen für Architekturen als Notation der Lerneinheit scheint es jedoch nicht sinnvoll, einen oder mehrere dieser Ansätze als Basis einer formalen Beschreibung zu verwenden. Stattdessen sollen Zustandsdiagramme zur Beschreibung dynamischer Aspekte von Architekturen eingesetzt werden, wie dies auch

der kommende UML-Standard 2.0 vorsieht. Zudem soll noch geprüft werden, ob eine Verwendung der *Object Constraint Language (OCL)* zur Beschreibung spezieller Eigenschaften von und Anforderungen an Architekturen dienen kann.

### 2.6 Übungen und Praktikum

Wie bereits erwähnt, soll die Lerneinheit in jedem Fall schwerpunktmäßig die praktische Arbeit mit Architekturen erlauben – also verschiedene Arten von Aufgaben unterstützen oder sich sogar in einen Praktikumsbetrieb eingliedern lassen.

## 3 Didaktisches Konzept

Dieser Abschnitt beschreibt einige der didaktischen Überlegungen, die im Rahmen der Lerneinheit durchgeführt wurden. Er orientiert sich an den Vorgaben von Koordinationsteam 1.

### 3.1 Leitbild

Die Lerneinheit richtet sich an Haupt- oder Nebenfachinformatiker, die – vermutlich durch eine Softwaretechnik-Vorlesung oder eine entsprechende Spezialvorlesung – ein Interesse an Software-Architekturen haben. Sie soll aber nicht auf Universitäts- oder FH-Studierende festgelegt sein, sondern zumindestens theoretisch auch innerhalb der beru¤ichen oder privaten Weiterbildung Verwendung £nden können.

Es wird angenommen, daß die Lernenden eine Quali£kation im Bereich des Entwurfs oder der Implementierung von größeren oder langlebigen Software-Systemen anstreben, also solchen Systemen, bei denen die Wahl einer geeigneten Architektur besonders wichtig ist. Eine alternative Zielgruppe stellen Lernende dar, die mit der Wartung größerer Altsysteme beschäftigt sind, zu denen keine explizite Beschreibung der Architektur existiert. Diese Lernenden würden hier die Quali£kation anstreben, Architekturen in vorhandenen Systemen zu erkennen.

Die Lerneinheit setzt Vorkenntnisse im Bereich der Programmierung voraus, wobei sich diese Vorkenntnisse wünschenswerterweise auf Systeme erstrecken, die den Umfang einer durchschnittlichen Erstsemester-Übungsaufgabe überschreiten. Anderenfalls dürften die Lernenden Schwierigkeiten haben, überhaupt den Sinn von Architekturen und deren Beschreibungen einzusehen. Vorausgesetzt werden auch Grundkenntnisse der UML, da sich sowohl die graphische Darstellung der Architekturen als auch die verwendeten Formalismen der UML bedienen werden.

### 3.2 Lernziele

Die Lernenden sollen durch die Lerneinheit in die Lage versetzt werden, praktische Erfahrungen mit Software-Architekturen zu sammeln. Nach dem vollständigen Durcharbeiten der Lerneinheit sollten sie folgende Quali£kationen erworben haben:

- Sie sollten wissen, warum und auf welche Weise die Wahl einer (un-) geeigneten Software-Architektur die Qualität eines Software-Systems beein¤ußt.

- Sie sollten bewährte Architekturen, deren Eigenschaften sowie Vor- und Nachteile kennen. Sie sollten wissen, wann der Einsatz einer bestimmten Architektur sinnvoll ist und wann nicht, und auf der Basis dieses Wissens ein Software-System entwerfen können.

- Zur Kenntnis der Eigenschaften von Architekturen zählt auch die Fähigkeit, die traditionell eher unbewußt eingesetzen architekturellen Idiome benennen zu können und so die Kommunikation innerhalb eines Entwickler-Teams oder zwischen dem Entwickler und dem Kunden zu verbessern.

- Die Lernenden sollten in der Lage sein, die Architektur zu erkennen, die einem vorhandenen System zugrunde liegt. Es sollte ihnen bewußt sein, daß ein System auch durch eine Kombination von Architekturen beschrieben werden kann oder daß es möglicherweise sogar mehrere disjunkte architekturelle Sichten auf ein System gibt.

## 3.3 Lernszenario

Die traditionelle Lehre wird im wesentlichen durch die Editor-Komponente des Systems unterstützt. In einer Vorlesung kann der Editor zur Visualisierung von zuvor erstellten Architektur-Beschreibungen eingesetzt werden und so den traditionellen Tafelanschrieb oder Overhead-Folien ergänzen bzw. ersetzen. Ein Vorteil dieser Vorgehensweise besteht darin, daß der Editor über eine Art „Zoom-Mechanismus“ aktuell interessante Bestandteile einer (z.B. hierarchisch gegliederten) Architektur in den Vordergrund rücken und so für mehr Überblick sorgen kann. Im Übungsbetrieb kann der gleiche Editor von Lernenden verwendet werden, um Architektur-Beschreibungen zu erstellen bzw. zu analysieren. Beim Einsatz innerhalb eines Praktikums müßte der Entwurf der System-Architektur mit dem Editor an einer geeigneten Stelle in den Software-Entwicklungsprozeß eingebunden werden.

Zur Unterstützung des Selbststudiums bzw. zum Nacharbeiten des Vorlesungsstoffs soll hypermedial aufbereitetes Lehrmaterial erstellt werden, das auf geeignete Weise mit dem Editor verknüpft wird. Insbesondere sollen Aufgaben existieren, die ein Lernender mit Hilfe des Editors bearbeitet und anhand derer er selbst oder ein Tutor seinen Fortschritt einschätzen kann.

Das System soll in der Lage sein, einfache Aufgaben automatisch zu korrigieren. Zu diesem Zweck wird der Editor insbesondere einen Mechanismus enthalten, der erwünschte oder unerwünschte Eigenschaften einer Architektur-Beschreibung prüfen und entsprechend darauf reagieren kann. Über die Lade- / Speicherschnittstelle des Editors ist auch eine Abgabe von Lösungen an einen menschlichen Tutor möglich. Es wird jedoch keine spezielle Infrastruktur zur Verwaltung des Übungsbetriebes bereitgestellt – dies ist Aufgabe der Integrationsplattform.

## 3.4 Didaktischer Ansatz

Das didaktische Konzept der Lerneinheit liegt nicht völlig fest – es kann vom Lehrenden durch die Art und Weise, wie er den Architektur-Editor verwendet, variiert werden. Das geplante „Komplett-Angebot“ aus Editor, vorde£nierten Architekturen, einführendem Material und Aufgaben läßt sich jedoch durch die Begriffe „fallbasiertes Lernen“ (da Architekturen

an Fallbeispielen verdeutlicht werden) und „oszillierend zwischen konstruktiven und dekonstruktiven Phasen“ (da Architektur-Beschreibungen in den Übungen sowohl erstellt als auch analysiert werden sollen) charakterisieren. Wenn die technische Umsetzung wie in Kapitel 5 beschrieben funktioniert, sollte auch entdeckendes Lernen realisierbar sein.

Die generelle Abfolge der Inhalte orientiert sich an Architektur-Kursen, wie sie von Shaw und Garlan [SG95] beschrieben sind. Eine zusätzliche hypermediale Vernetzung von Inhalten £ndet z.B. dann statt, wenn Architekturen verglichen oder für ein bestimmtes Problem als Alternativen zueinander vorgestellt werden.

# 4 Strukturkonzept

Dieser Abschnitt beleuchtet die inhaltliche Struktur der Lerneinheit etwa genauer. Die Struktur folgt den in Abschnitt 2 vorgestellten groben inhaltlichen Teilbereichen, so daß der Abschnitt im wesentlichen eine feinere Sicht auf diese Teilbereiche bietet.

## 4.1 Grundlagen

Die wesentliche Frage, die im ersten Teilbereich geklärt wird, lautet: „Was ist Software-Architektur?“ Die Frage gliedert sich zum Beispiel in folgende Teilaspekte:

- Warum ist Software-Architektur wichtig?
- Auf welchem Abstraktions-Niveau bewegt man sich, wenn man über die Architektur eines Software-Systems redet?
- Wie gliedert sich Software-Architektur in den Entwicklungsprozeß ein?
- Wie kann Software-Architektur zu einer höheren Qualität von Software oder zu einem ef£zienteren Entwicklungsprozeß beitragen?
- Wie kann Software-Architektur zu einer verbesserten Kommunikation beitragen?
- Welchen Nutzen haben formale Beschreibungen von Software-Architekturen?
- Wie kann Software-Architektur zu einer verbesserten Wartung eines Software-Systems beitragen?
- Wie läßt sich die Idee der Architekturstile, die gelegentlich auch als Architekturmuster [BMR$^+$96] bezeichnet werden, von Entwurfsmustern abgrenzen?

## 4.2 Notation

Zu den Konzepten und Notationselementen, die im zweiten Teilbereich eingeführt werden, gehören folgende:

- Kapsel (Capsule) – Ein Element einer Architekturbeschreibung, in dem Berechnungen statt£nden oder Daten abgelegt werden. In den klassischen Arbeiten zu Software-Architektur werden Kapseln als Komponenten bezeichnet.

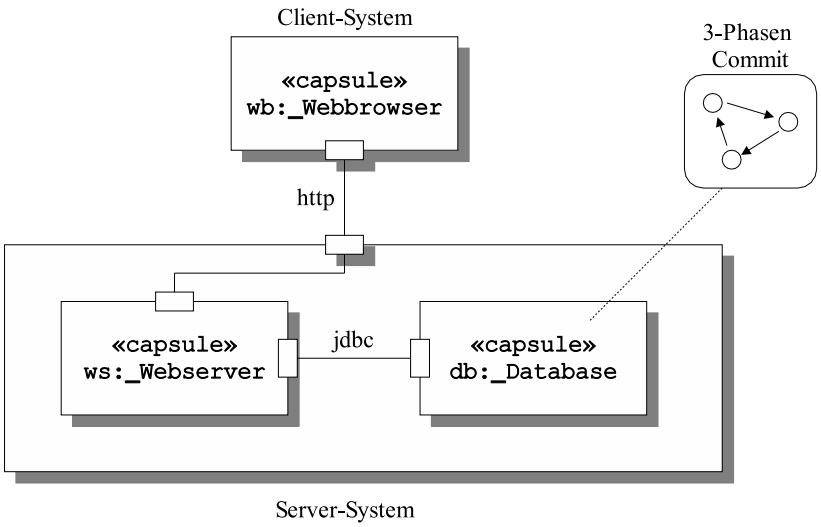



Abbildung 1: Graphische Notation des Editors

- Anschluß (Port) – Eine Schnittstelle zwischen dem Innenleben einer Kapsel und ihrer Außenwelt. Jegliche Interaktion mit anderen Kapseln £ndet über diese Schnittstelle statt.
- Konnektor (Connector) – Eine Verbindung zwischen mehreren Kapseln bzw. Anschlüssen, die Daten- oder Kontroll¤uß realisiert.
- Protokoll (Protocol) – Beschreibt das Verhalten sowohl von Kapseln als auch von Konnektoren.
- Rolle (Role) – Bezeichnet das Ende eines Konnektors.

Neben diesen speziellen graphischen Konstrukten zur Beschreibung einer statischen Systemstruktur wird das Verhalten von Komponenten mit Hilfe von Zustandsdiagrammen beschrieben. Abbildung 1 skizziert, wie die Architektur eines imaginären Software-Systems aus Datenbank mit Web-Schnittstelle auf der Basis dieser Notation aussehen könnte.

## 4.3 Architekturstile

Innerhalb des dritten Teilbereiches sollen die folgenden Stile ausführlich vorgestellt werden:

- Pipes and Filters – Dieser Stil setzt sich aus aktiven, datenverarbeitende Einheiten (den Filtern) und eher passiven, Daten¤uß herstellenden Verbindungen (den Pipes) zusammen. Bei der Spezialisierung „Pipeline“ liegt eine streng lineare Folge von Filtern vor. Konsumiert ein Filter seine Eingabe komplett, bevor er die Ausgabe erzeugt, degeneriert der Stil zu einer „Batch Sequential“-Architektur. Ein Beispiel für eine *Pipes and*

*Filters*-Architektur ist die Art und Weise, wie in einer Unix-Shell Befehle verknüpft werden können.

- Repository – Bei diesem Stil operiert eine Menge von unabhängigen Komponenten auf einer zentralen Datenstruktur. Bestimmt die Art und Reihenfolge der Anfragen das Scheduling der durchgeführten Operarationen, dann liegt die Spezialisierung „Datenbank“ vor. Wird das Scheduling hingegen durch den Zustand der zentralen Datenstruktur bestimmt, handelt es sich um eine „Blackboard“-Architektur.

- Hierarchical Layers – Folgt ein System diesem Stil, wird es aus einer Reihe von Schichten unterschiedlichen Abstraktionsgrades aufgebaut. Jede Schicht erbringt eine Menge von Diensten für die nächsthöhere Schicht und bedient sich dabei der Dienste, die von der nächsttieferen Schicht angeboten werden. In manchen Systemen sind die Schichten halbtransparent, das heißt, eine Schicht kann sich aller tieferliegenden Schichten bedienen. Ein vielen Studenten bekanntes Beispiel für eine Schichtenarchitektur ist das ISO/OSI-Referenzmodell für Telekommunikationssysteme [Tan96].

- Implicit Invocation – In Systemen, die nach diesem Stil aufgebaut sind, liegt eine lose Kopplung der Komponenten vor: Kommunikation zwischen den Komponenten geschieht nicht direkt über den Aufruf von Prozeduren, sondern indirekt über Nachrichten. Eine Komponente kann bei einer anderen Komponente das Interesse an einem bestimmten Ereignis anmelden und die Adresse einer Prozedur hinterlegen. Tritt das Ereignis zu einem späteren Zeitpunkt auf, wird die Methode aufgerufen. Hier liegt eine deutliche Verwandtschaft zum Entwurfsmuster „Listener“ [BMR+96, GHJV96] vor. Beispiele für den Stil £nden sich in vielen modernen Fenstersystemen, etwa der Swing-Bibliothek von Java [CWH01].

Die Stile sind so gewählt, daß sie zu den Fallstudien passen, die im weiteren Verlauf der Lerneinheit vorgestellt werden. Abschnitt 4.4 enthält weitere Informationen dazu. Weitere, ebenfalls technisch interessante Stile sind *Interpreter* und *Rule-Based System*. Diese könnten zusätzlich in den Katalog aufgenommen werden.

Die Stile *Main Program and Subroutine* und *Abstract Data Type/Object-Orientation* scheinen, obwohl bei Garlan und Shaw [GS94, SG95] aufgeführt, nicht zum Abstraktionsniveau der Architekturen zu passen, da sie sich eher auf der Ebene des Entwurfs bzw. sogar der Implementierung bewegen. Sie sollen deshalb keinen Eingang in den Katalog £nden oder nur kurze Erwähnung £nden. Gegen letzteren Stil spricht zudem, daß Objektorientierung sowohl in MuSofT als auch im heutigen Alltag der Software-Entwicklung Standard ist und somit nicht mehr unbedingt den Rang eines eigenen Stils verdient.

## 4.4 Fallbeispiele

Die folgenden Fallstudien sind derzeit geplant, wobei „N.N.“ bedeutet, daß eine entsprechende Fallstudie noch gesucht wird:

- Das KWIC-System – Hier sollen die beiden ursprünglichen Varianten von Parnas [Par72] bzw. die vier Varianten von Garlan und Shaw [GS94, SG95] dargestellt und verglichen werden.

- Ein Compiler – Hier soll die traditionelle, auf Pipes and Filters basierende Architektur schrittweise in eine Architektur überführt werden, die auf einem zentralen Repository basiert. Auch dieses Beispiel entstammt Garlan und Shaw [GS94, SG95], wird jedoch auch ähnlich von Perry und Wolf [PW92] verwendet.
- N.N. – Ein Beispiel für eine Schichtenarchitektur. Hier wurde noch keine Entscheidung getroffen, gute Kandidaten sind aber das XWindow-System oder ein (nicht zu kompliziertes) Betriebsystem, dessen Architektur publiziert ist (z.B. Dijkstras THE-System). In beiden Fällen ist es vielleicht auch möglich, einen „Gegenkandidaten“ zu £nden, bei dem sich die Wahl der Architektur im Nachhinein als falsch erwiesen und somit zum Mißerfolg des Systems beigetragen hat.
- N.N. – Ein Beispiel für eine gemischte Architektur und ein weiteres für ein System, das verschiedene architekturelle Sichten zuläßt. Während gemischte Architekturen eher der Regelfall sind und somit auch zahlreich in der Literatur vorkommen, liegt für den zweiten Fall noch kein Beispiel vor.
- N.N. – Ein Beispiel für eine domänenspezi£sche oder Referenzarchitektur aus der Wirtschaft, um zu zeigen, wie im nicht-akademischen Umfeld mit Architekturen umgegangen wird. Gute Kandidaten sind hier IBM, da dort generell viel publiziert wird, und sd&m, deren softwaretechnische Basis im Buch von Dehnert [DS91] zu £nden ist.

Bei den mit „N.N.“ gekennzeichneten bietet sich eventuell auch die Übernahme eines Fallbeispiels aus einem anderen MuSofT-Teilprojekt an. Dies wird derzeit noch überprüft.

## 4.5 Formale Methoden

Bei der Behandlung formaler Methoden stehen State Charts und (vermutlich) die OCL im Mittelpunkt. Zu den Fragestellungen, die in diesem Kontext betrachtet werden sollen, gehören die folgenden:

- Wie können State Charts genutzt werden, um das Verhalten eines Systems oder von einzelnen Komponenten zu beschreiben?
- Wie kann die OCL genutzt werden, um Teile des Systems mit speziellen Eigenschaften (etwa dem Speicherverbrauch o.ä.) auszuzeichnen oder Anforderungen eines Kunden an das Gesamtsystem auszudrücken?

## 4.6 Übungen und Praktikum

Wie bereits erwähnt, soll die Lerneinheit in jedem Fall schwerpunktmäßig die praktische Arbeit mit Architekturen erlauben – also verschiedene Arten von Aufgaben unterstützen oder sich sogar in einen Praktikumsbetrieb eingliedern lassen. Die folgenden Arten von Aufgaben sind geplant:

- Verständnisaufgaben – Beantworten von Fragen zu bestimmten Architekturen, z.B. ob in einer gegebenen Schichtenarchitektur ein bestimmter Prozeduraufruf erlaubt ist oder nicht (sehr einfaches Beispiel!).

- Analytische Aufgaben – Erkennen von Architektur-Stilen oder Kombinationen von Stilen in einer gegebenen Architektur.

- Konstruktive Aufgaben – Erstellen einer Architektur(-beschreibung), die einem bestimmten Stil oder bestimmten Anforderungen folgt.

- Formale Aufgaben – Wenn formale Spezi£kationsmethoden von der Lerneinheit unterstützt werden, dann sollten diese auch Eingang in die Übungen £nden. Zum Beispiel könnten gegebene, mit UML modellierte Architekturen mit OCL-Contraints dekoriert werden, um bestimmte Anforderungen wiederzugeben. Ebenso kann den Studierenden sicherlich zugemutet werden, das Verhalten eines einfachen Systems mit Zustandsdiagrammen zu modellieren und anschließend „ablaufen“ zu lassen.

Genauere Szenarien zum Übungsbetrieb oder gar einzelne Aufgaben sind zum jetzigen Zeitpunkt noch nicht bekannt. Dieser Aspekt der Lerneinheit ist unter anderem davon abhängig, wie sich die Integrationsplattform weiter entwickelt.

# 5 Lehr/Lernumgebung

Dieser Abschnitt beschreibt einige der grundlegenden technischen Ideen zur multimedialen Umsetzung des geplanten Inhalts. Prinzipiell besteht die technische Umsetzung der Lerneinheit aus zwei Komponenten:

- Editor-Werkzeug – Der Editor dient zum Visualisieren und Bearbeiten von Software-Architekturen. Je nach Betrachtungsweise handelt es sich dabei um ein einfaches CASE-Tool oder ein 2D-Vektorgra£k-Programm, bei dem komplexere Elemente (Kapseln und Konnektoren) de£niert und auf der Basis vorgegebener Anforderungen oder Regeln (Architektur-Stile) verbunden werden können.

- Hypertext-System – Das Hypertext-System dient zur Vermittlung des in den Abschnitten 2 und 4 beschriebenen Lehrstoffes. Die Nutzung multimedialer Möglichkeiten beschränkt sich auf Text, Gra£ken und Hyperlinks, wobei man durchaus überlegen kann, im Einzelfall Videos o.ä. mit aufzunehmen, wenn sich ein sinnvoller Anwendungsfall £ndet. Falls ein Teil der Übungen über das Hypertext-System realisiert werden soll (z.B. klassische Aufgabenzettel), dann werden auch Formular-Elemente benötigt.

Diese beiden Kompoenenten sollen zu einer einzigen Anwendung verbunden werden, die dann die eigentliche Lerneinheit darstellt. Das könnte zum Beispiel so geschehen, daß ein Fenster der Anwendung zur Bearbeitung von Architekturen dient und ein anderes den Lehrstoff enthält. Abbildung 2 skizziert dies. Zwischen beiden Arten von Fenstern sollen komplexe Querbezüge möglich sein:

- Das Anklicken bestimmter Hyperlinks soll (z.B. Skript-gesteuert) Elemente der angezeigten Architektur hervorheben oder verändern können. Damit lassen sich etwa „Guided Tours“ durch die in Abschnitt 2 beschriebenen Architekturen und Stile realisieren.

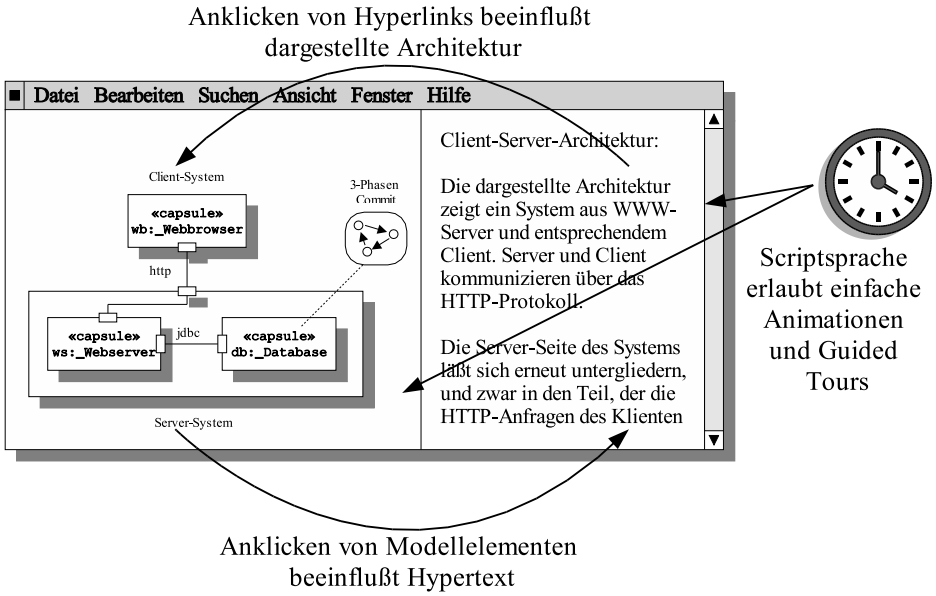



Abbildung 2: Beziehungen zwischen Editor und Hypertext-System

- Über komplexere Skripte sollen begrenzte Animationen (eher eine Art „Dia-Show“) realisiert werden können, zum Beispiel eine, die das Umwandeln des auf Pipes and Filters basierenden Compilers in eine Variante zeigt, die ein zentrales Repository nutzt, und diesen Vorgang mit geeigneten Kommentaren begleitet.
- Bei bestimmten Aktionen oder Veränderungen in der Architektur durch den Studierenden sollen analog bestimmte Inhalte im Hypertext-Bereich angezeigt werden können:
  - Beim Klick auf die Elemente einer der Fallstudien könnte so zum Beispiel direkt ein Text angezeigt werden, der das entsprechende Element erläutert („Tool-Tips“ zu einer Architektur). Damit wäre eine Unterstützung für entdeckendes Lernen gegeben.
  - Auf der Basis von Eigenschaften einer vom Studierenden erstellen Architektur könnten bestimmte Inhalte im Hypertext-Fenster angezeigt werden. Auf dieser Basis ließe sich zum Beispiel die automatische Korrektur einfacher Übungsaufgaben innerhalb des Werkzeugs realisieren.
- Erstellte Architekturen können gespeichert und zu einem späteren Zeitpunkt wieder geladen werden. Dies ist gleichzeitig die Schnittstelle zum traditionellen Übungsbetrieb, der durch die Integrationsplattform organisiert wird und die Korrektur von Lösungen durch einen menschlichen Tutor oder eine Automatik vorsieht.

# 6 Technische Realisierung

Den Absprachen der beiden Plenumstreffen folgend soll die technische Umsetzung der Lernumgebung in Java geschehen. Dies garantiert eine Einsetzbarkeit auf einer maximalen Zahl von Plattformen und kommt damit der Nachhaltigkeit des Projektes zugute. Für die beiden größeren Komponenten der Lernumgebung, Editor und Hypertext-System, können eventuell bestehende Bibliotheken Verwendung £nden Für das Hypertext-System bietet es sich an, ein bestehendes Anzeigeelement für HTML- oder RTF-Dateien einzusetzen oder sich direkt des Online-Hilfe-Mechanismus bedienen, der Swing-Anwendungen zur Verfügung steht.

Als mögliche Basis des Editors wurde zunächst das „Open Source“-Modellierungswerkzeug ArgoUML untersucht, da derzeit kein Java-basiertes Werkzeug für Software-Architekturen existiert, das den oben genannten Anforderungen genügt. ArgoUML hätte im Rahmen von MuSofT um die bereits erwähnte graphische Notation für Architekturen erweitert werden müssen. Leider stellte sich diese Variante als nicht tragfähig heraus, da die Anpassung sehr aufwendig geworden wäre. Es wird also ein eigener Architektur-Editor auf der Basis einer bestehenden Java-Bibliothek zur Visualisierung und Bearbeitung von Graphen etc. entstehen. Im Januar 2001 fand ein MuSofT-interner Workshop statt der die Anwendbarkeit von bestehenden Bibliotheken beleuchtet und gegen die Alternative einer vollständigen MuSofT-Eigenentwicklung abgewogen hat. Als mögliche Kandidaten sind die Bibliothen JHotDraw und YFiles zur weiteren Evaluation ausgewählt worden. Eine Entscheidung zu diesem Thema wird möglicherweise auf oder nach dem Plenumstreffen im Frühjahr 2002 fallen.

# 7 Evaluierung

Da die Entwicklung der Lerneinheit inkrementell geschieht, stehen bereits frühzeitig Teile der Funktionalität zur Verfügung, die praktisch eingesetzt und evaluiert werden können. Beispielsweise werden die rein visualisierenden Elemente zu Architekturstilen und Fallstudien bereits zum Wintersemester 2002 im Vorlesungsbetrieb einsetzbar sein. Weitere Teile werden evaluiert, sobald sie zur Verfügung stehen. Während der letzten Monate des Projektes, also im Winter 2003, £ndet eine Evaluation der gesamten Lerneinheit statt.

Für den Einsatz bzw. die Evaluation der Lerneinheit sind folgende Veranstaltungen an der Universität Dortmund geeignet:

- Die Vorlesung „Softwaretechnik“, wo der Editor im wesentlichen als Visualisierungskomponente eingesetzt wird.
- Der Übungsbetrieb zur gleichen Vorlesung. Hier nutzen die Studierenden den Editor zur Lösung von Aufgaben und verwenden das hypermediale Lehrmaterial zum Nacharbeiten des Vorlesungsstoffes.
- Das Software-Praktikum. In dieser Veranstaltung werden bereits jetzt CASE-Tools projektbegleitend eingesetzt, jedoch bewegen sich diese auf der Ebene von UML-Analyse- und Entwurfsdiagrammen. Der Editor könnte hier helfen, frühzeitig eine Architektur für die zu entwickelnde Anwendung auszuwählen, die anschließend mit den bisherigen Werkzeugen verfeinert werden kann.

Für den Einsatz in Vorlesungen an der Universität Dortmund ist die Zielgruppe – nämlich die der Lehrenden – so klein, so daß sich eine Evaluation über Fragebögen nicht anbietet. Der Stichprobenumfang ist zu gering, um eine quantitative Aussage zu erlauben. Hier wird also der subjektive Eindruck des Lehrenden genügen müssen, daß die Präsentation von Architekturen mit dem Editor „besser“ oder „schlechter“ als das bisherige bisherigen System (Tafelanschrieb oder Folien) war. Eventuell ändert sich dies, wenn das Werkzeug eine größere Verbreitung (innerhalb und außerhalb von MuSofT) £ndet.

Im Übungsbetrieb und im Praktikum ist es sehr wohl möglich, die Studenten zu ihren Erfahrungen mit dem Werkzeug zu befragen. Speziell am Software-Praktikum nehmen so viele Studenten teil, daß sich möglicherweise anbietet, ein kontrolliertes Experiment durchzuführen, um den Ein¤uß der Verwendung des Werkzeugs auf den Projektfortschritt (Einhaltung des Zeitplans, gefundene Fehler in späteren Phasen) zu ermitteln.

# 8 Zusammenfassung

Software-Architektur ist ein eher praktisches Thema innerhalb der Softwaretechnik. Das muß bei der Lehre berücksichtigt werden. Neben der Vermittlung von reinem Theoriewissen in Vorlesungen, die selbstverständlich unverzichtbar ist, muß den Studierenden eine Möglichkeit gegeben werden, eigene Erfahrungen auf dem Gebiet der Software-Architektur zu sammeln oder von den Erfahrungen derer zu pro£tieren, die sich zuvor mit dem Thema beschäftigt haben.

Die Lerneinheit, die innerhalb von Teilprojekt 2.1 entwickelt wird, versucht eine Infrastruktur zu schaffen, die diesem Wunsch Rechnung trägt: Das Sammeln eigener Erfahrungen wird durch die Editor-Komponente ermöglicht, die eine Vertiefung des Vorlesungsstoffes in Übungen ermöglicht. Das Hypertext-System erlaubt dabei jederzeit einen Rückgriff auf den Vorlesungsstoff oder weiterführende Informationen. Das Erfahrungswissen zahlreicher Software-Ingenierure wird – ähnlich Entwurfsmustern – durch Architekturstile kodi£ziert, von denen die Lerneinheit eine kleine Auswahl vermittelt. Die Fallstudien erlauben zudem einen Blick darauf, wie sich Architekturstile auf die Eigenschaften verschiedener realer und imaginärer Projekte auswirken. Dem berechtigten Wunsch nach präziseren Beschreibungen als sie natürlichsprachlich möglich sind kommen die formalen Anteile der Lerneinheit nach, die mit State Charts und der OCL auf Standard-Formalismen der UML basieren.

Das erste Projektjahr war im wesentlichen von einer Einarbeitung in das Thema Software-Architektur, einer Auswahl der multimedial umzusetzenden Aspekte und einer Planung der Lerneinheit geprägt. Zu den Arbeiten, die sich in der ersten Jahreshälfte 2002 anschließen werden, gehören unter anderem folgende:

- Verfeinerung des Konzeptes, Treffen der endgültigen Entscheidungen bezüglich der technischen Realisierung und des Inhaltes.
- Implementierung des Editors für Struktur- und Zustandsdiagramme, Bereitstellen von Funktionen zum Laden, Speichern und Drucken der Diagramme.
- Benutzen dieses Editors zum Erstellen der graphischen Anteile (Diagramme) des Kapitels über Architekturstile.

Damit stehen die grundlegenden Elemente des Editors sowie ein Teil der Beispiele und Fallstudien zu Architektur-Stilen zum Wintersemester 2002 bereit. Sie können in dieser Form bereits in Vorlesungen oder für Übungsaufgaben genutzt werden. Im weiteren Verlauf Laufe des Jahres 2002 und des abschließenden Projektjahres 2003 wird die Lerneinheit dann einen Ausbau und die nötige technische und inhaltliche Abrundung erfahren.

Insgesamt – so hofft der Autor – wird sich aus den in diesem Bericht genannten Zutaten eine Lerneinheit ergeben, die den Studierenden einen multimedialen Ein- und Überblick der Software-Architektur gibt und sowohl Vorlesungen als auch Übungs- und Praktikumsbetrieb der Softwaretechnik gleichermaßen bereichert.

## Literatur


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

[CWH01] CAMPIONE, MARY, KATHY WALRATH und ALISON HUML: *The Java Tutorial, Third Edition*. Addison Wesley, 2001.

[DC94] DEAN, THOMAS R. und JAMES R. CORDY: *A Syntactic Theory of Software Architecture*. IEEE Transactions on Software Engineering, 21(4):302–313, 1994.

[DS91] DEHNERT, ERNST und JOHANNES SIEDERSLEBEN: *Software-Engineering*. Springer, 1991.

[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.

[HHK+96] HOFMANN, CHRISTOPH, ECKART HORN, WOLFGANG KELLER, KLAUS RENZEL und MONIKA SCHMIDT: *The Field of Software Architecture*. Technischer Bericht TUM-I9641, 1996.

[IW95] INVERARDI, PAOLA und ALEXANDER L. WOLF: *Formal Speci£cation and Analysis of Software Architecture using the Chemical Abstract Machine*. IEEE Transactions on Software Engineering, 21(4):373–386, 1995.

[LKA+95] LUCKHAM, DAVID C., JOHN J. KENNEY, LARRY M. AUGUSTIN, JAMES VERA, DOUG BRYAN und WALTER MANN: *Speci£cation and Analysis of System Architecture Using Rapide*. IEEE Transactions on Software Engineering, 21(4):336–355, 1995.

[MEH01] MAIER, MARK W., DAVID EMERY und RICH HILLIARD: *Software Architecture: Introducing IEEE Standard 1471*. IEEE Computer, 34(4):107–109, 2001.

[Par72] PARNAS, DAVID L.: *On the Criteria to Be Used in Decomposing Systems into Modules*. Communications of the ACM, 15(12):1053–1058, 1972.

[PW92] PERRY, DAVID E. und ALEXANDER L. WOLF: *Foundations for the Study of Software Architecture*. ACM SIGSOFT Software Engineering Notes, 17(4):40–52, 1992.

[SDK+95] SHAW, MARY, ROBERT DELINE, DANIEL KLEIN, THEODORE ROSS, DAVID YOUNG und GREGORY ZELESNIK: *Abstractions for Software Architecture and Tools to Support Them*. IEEE Transactions on Software Engineering, 21(3):314–335, 1995.

[SG95] SHAW, MARY und DAVID GARLAN: *Software Architecture - Perspectives on an Emerging Discipline*. Prentice Hall, 1995.

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

[SW99] STAFFORD, JUDITH A. und ALEXANDER L. WOLF: *Architecture-Based Software Engineering*. Internal Report CU-CS-891-99, 1999.

[Tan96] TANENBAUM, ANDREW S.: *Computer Networks, Third Edition*. Prentice Hall, 1996.

# Lerneinheit Entwurfsmuster – Jahresbericht 2001 MuSofT Teilprojekt 2.2

S. Seehusen, N. Nissen

Das inhaltliche und das didaktische Konzept der im Projekt MuSofT entwickelten Lerneinheit Entwurfsmuster wird vorgestellt. Die empfohlenen Lernszenarien werden dargestellt und ein Konzept für eine konkrete Lernumgebung der Studierenden entwickelt. Die Lernumgebung soll die bisherige scharfe Trennung zwischen Präsenzstudium und virtueller Lehre aufheben und die Lernformen durchlässig gestalten. In der technischen Realisierung werden diverse Werkzeuge zur Erstellung der verschiedenen Medienobjekte ausgewählt. Die Medienobjekte werden mit XML zu einer Lerneinheit integriert.

## Inhaltsverzeichnis



## 1 Einführung

Entwurfsmuster sind ein wichtiges Konzept der Softwaretechnik. Es dient insbesondere der Vermittlung von Lösungen von häu£g 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 u.a. 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 Inhaltliches Konzept

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 u.a. [GHJV95] und [BMR+96]. Es gibt weiterhin diverse Bücher, auch keine Lehrbücher, in denen weitere Entwurfsmuster beschrieben und kategorisiert sind, u.a. [SSRB00]. Daneben gibt es diverse Literatur zu programmiersprachspezi£schen Anwendung, z.B. in [Gra98] oder [Coo00] und entsprechende Web-Sites.

Die Lerneinheit Entwurfsmuster soll eine allgemeine Einführung in Entwurfsmuster und in die Beschreibungsstruktur von Entwurfsmustern geben. Desweiteren werden wichtige Kategorien von Entwurfsmustern wie Architektur-, Erzeugungs-, Struktur- und Verhaltensmuster sowie Muster zur Arbeitsorganisation eingeführt. Zu diesen ausgewählten Kategorien werden repräsentative und häu£g verwendete Entwurfsmuster präsentiert. Die Auswahl der Entwurfsmuster ist in Kapitel 4 aufgeführt.

Zu jedem Entwurfsmuster wird ein vollständiges Anwendungsbeispiel einschließlich Klassen- und Sequenzdiagramm und Quellcode in Java gegeben, in dem ein Teil des Systems nach dem Entwurfsmuster entworfen und implementiert wurde.

Auch die Kombination von verschiedenen Entwurfsmustern wird anhand eines durchgängigen größeren Beispiels diskutiert.

## 3 Didaktisches Konzept

Es werden die didaktischen Vorüberlegungen zum didaktischen Konzept der Lerneinheit Entwurfsmuster vorgestellt.

## 3.1 Leitbild

Das Leitbild umfasst eine typisierende Beschreibung der lernenden Person, sowohl in ihrem Kontext in der Ausbildung (Studiengang, Studienfach), als auch in ihrem Kontext der späteren beru¤ichen Praxis.

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

- **Studienfach**: 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.

  Ein Teil der Lernmodule wird z.B. in Lübeck im 4. Semester in der Lehrveranstaltung Programmiertechniken der Studienrichtung Informatik des Studiengangs Elektrotechnik (genauer: Kommunikations-, Informations- und Medientechnik (KIM)) zum Thema Entwurfsmuster eingesetzt. Ein weiterer Teil der Lernmodule kann z.B. in einer entsprechenden Vertiefungsveranstaltung angeboten werden.

- **Angestrebte beru¤iche Quali£kationen**, beru¤iche Einsatzfelder:

  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.

- **Individuelle kognitive und methodische Vorkenntnisse**:

  Es werden grundlegende Kenntnisse in objektorientiertem Entwurf und objektorientierter Programmierung vorausgesetzt, einschließlich UML und Java.

  Wird die Lerneinheit in der Weiterbildung eingesetzt, wird die Fähigkeit zum Selbststudium vorausgesetzt.

- Art der **Ausbildungsinstitution** (z.B. Universität, Fachhochschule, Weiterbildungszentrum eines Unternehmens):

  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.

## 3.2 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.

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 Qualitätssichungsprozess fördern.

Durch die Kenntnis von Entwurfsmustern sollen die Studierenden komplexe Bibliotheken wie z.B. einige Java-APIs einfacher nutzen können.

## 3.3 Lernszenario

Die erstellten Lernmodule werden im wesentlichen in zwei unterschiedlichen Szenarien eingesetzt.

#### Lernszenario traditionelle Lehre

Im ersten Szenario werden die erstellten Materialien, insbesondere die Animationen, in der Präsenzvorlesung zur Erläuterung der Lehrinhalte eingesetzt. In der Übung bzw. 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.

#### 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 stof¤ichen Inhalte eingesetzt, die von besonderem Interesse für die lernende Person ist.

Da es pro präsentiertem Entwurfsmuster eine relativ ausführliche Einführung und Beschreibung sowie im Referenzteil eine konzentrierte Beschreibung gibt, kann die lernende Person je nach Vorkenntnissen die entsprechende Präsentation auswählen.

#### Lehr-/Lernmaterial

Das Lehr- und Lernmaterial besteht für die Vorlesung aus Folien, Texten, Bildern, Gra£ken 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 bearbeitet werden können. Diese Tests werden in der Regel von jeweils einer Einzelperson durchgeführt.

Des weiteren werden größere Aufgaben angeboten, die in der Regel eine konstruktive Entwicklungsarbeit umfassen und eine Gruppenarbeit von Studierenden erfordern.

#### Betreuung der Studierenden

Die Studierenden werden in der traditionellen Lehre im Praktikum von Laborpersonal bzw. Tutoren oder Tutorinnen bei der Bearbeitung ihrer Entwicklungsaufgaben betreut. Email übernimmt dabei einen Teil der Sprechstunden.

Bei der Bearbeitung der einfachen Testaufgaben werden die Studierenden in der Regel nicht betreut, sondern sind auf eine maschinell erzeugte Antwort angewiesen.

### 3.4 Didaktischer Ansatz

Die Lerneinheit und die Lernszenarien verknüpfen mehrere didaktische Ansätze, die teilweise auch in den verschiedenen Teilen der Lerneinheit explizit sichtbar sind.

Die Vorlesung ist an die Fachsystematik angelehnt. Des weiteren enthält sie relativ viele analytische Elemente. Das begleitende Praktikum bzw. Übung ist teilweise analytisch und in den größeren Aufgaben auch konstruktiv in dem Sinne angelegt, dass ein Teil eines Produktes konstruktiv erstellt wird.

#### Vernetzung der Lerninhalte

Die zu lernenden Inhalte werden in den ersten zwei Teilen inhaltlich aufeinander aufgebaut. Diese Teile sind stark mit dem Referenzteil (Teil 5) und auch mit der zusammenhängenden Beschreibung des durchgängigen Beispiels (Teil 3) vernetzt. Vom Referenzteil gibt es entsprechende Rückverweise auf die einführenden Teile 1 und 2.

In den einzelnen Teilen, insbesondere im Teil Ausblick, wird auf vertiefende Inhalte verwiesen, die teilweise auch im Web verfügbar sind. Insbesondere soll auf andere Lernmodule verwiesen werden, die im Projekt MuSofT entwickelt werden.

## 4 Struktur der Lerneinheit

Die Lerneinheit wird in 5 Teile gegliedert, die jeweils aus Lernmodulen (LM, nach der MuSofT-Terminologie) bestehen. Die Teile 3 und 5 haben Referenzcharakter. Auf diese Teile wird aus den anderen Lernmodulen verwiesen.

### Teil 1: Einführung in Entwurfsmuster

- LM 1.1: Allgemeine Einführung
- LM 1.2: Einführendes Entwurfsmuster

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

## Teil 2: Entwurfsmuster

Es wird eine Auswahl weiterer Entwurfsmuster eingeführt, wobei jeweils eins oder mehrere aus verschiedenen Kategorien wie Architektur-, Erzeugungs-, Struktur- und Verhaltensmuster sowie Muster zur Arbeitsorganisation gewählt werden. Ein weiteres Auswahlkriterium ist die Verwendung für interaktive Programme.

Jedes Entwurfsmuster wird in einem eigenen Lernmodul dargestellt. Folgende Muster sind geplant:

Erzeugungsmuster:

- LM 2.1: Singleton
- LM 2.2: Fabrikmethode

Verhaltensmuster:

- LM 2.3: Strategie
- LM 2.4: Beobachter (Observer)
- LM 2.5: Befehl (Command)
- LM 2.6: MVC
- LM 2.7: Iterator
- LM 2.8: Besucher (Visitor)
- LM 2.9: Memento (evtl.)

Strukturmuster:

- LM 2.10: Kompositium
- LM 2.11: Adapter
- LM 2.12: Bridge

Arbeitsorganisation:

- LM 2.13: Delegation
- LM 2.14: Master-Slave
- LM 2.15: Blackboard
- LM 2.16: Broker (evtl.)

Zu der Beschreibung jedes Entwurfsmusters gehört

- ein Einführungsbeispiel,
- Beschreibung nach dem 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,
- Aufgaben

### Teil 3: Durchgängiges Beispiel

So weit wie möglich wird ein durchgängiges Beispiel zur Demonstration der Anwendung der Entwurfsmuster einschließlich Implementierung in Java eingesetzt. Als Beispiel ist die Realisierung eines Kommunikationswerkzeuges mit den Funktionen Bulletin Board, Chat und WhiteBoard geplant. In diesem Teil 3, dem durchgängigen Beispiel, wird das Beispiel zusammenhängend dargestellt. In Teil 2 wird pro Entwurfsmuster jeweils ein Teilaspekt näher beleuchtet. Der Teil 3 dient als Überblick und als Referenz zum durchgängigen Beispiel. Er besteht zunächst aus einem Lernmodul:

- LM 3.1: Beispiel Kommunikationswerkzeug

### Teil 4: Ausblick

- LM 4.1: Es 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.

### Teil 5: Referenzteil

In Teil 2 werden die einzelnen Entwurfsmuster einführend beschrieben. In diesem Teil 5, dem Referenzteil, wird jedes Entwurfsmuster noch einmal nach einem festen Beschreibungsschema kurz beschrieben. Es ist kein Lernmodul im engeren Sinne, da es einen Referenzcharakter besitzt. Es ist jedoch ein Bestandteil der Lerneinheit und kann insbesondere auch zur Wiederholung oder zur Auswahl von Entwurfsmustern genutzt werden.

- LM 5.1: Referenzhandbuch

### 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 z.B. wiederholend sein, es können Analyseaufgaben sein, sie können eine Re¤exion 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 £nden und frei beschreiben muss.

# 5 Lehr- und Lernumgebung

Die Lehr- und Lernumgebung besteht aus einer verteilten Umgebung, die aus verschiedenen Mengen von Werkzeugen bestehen kann, die jeweils eine unterschiedliche Nutzung unterstützen.

Im Kapitel 6 wird genauer auf die Umgebung zur Entwicklung und späteren Weiterentwicklung und Aktualisierung der Lerneinheit eingegangen.

## 5.1 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 spezi£sche Arbeitsumgebungen z.B. zur Unterstützung der virtuellen Gruppenarbeit (z.B. 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.

## 5.2 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 (min. 100 Mb/sec) vorausgesetzt.

Als Präsentationswerkzeuge sollten Web-Browser, z.B. Netscape, mit entsprechenden Plugins für Animationen, Audio, Video und 3D ausreichen.

Zur Lehrumgebung gehört auch eine relativ einfache Möglichkeit, die Lerneinheit zu aktualisieren.

## 5.3 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 u.a. folgende Werk-

zeuge enthält:

- Web-Browser, z.B. Netscape, mit einigen “Standard”-Plugins, mit Java-Plugin,
- UML-Editor, möglichst mit XMI-Ausgabeformat, z.B. argouml,
- JDK 2,
- Linux empfohlen, Windows auch möglich,
- HTML-Editor für Aufschreiben von Lösungen der Aufgaben,
- evtl. Softwareentwicklungsumgebung wie z.B. Together,
- Werkzeuge zur asynchronen Kommunikation empfohlen wie email, news und Bulletin Board und
- Werkzeuge zur synchronen Kommunikation empfohlen wie z.B. chat, Netmeeting, CU-seeMe 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.

Die Liste der konkreten Werkzeuge gibt einen Ist-Stand wieder und wird gegebenenfalls aktualisiert.

# 6 Technische Realisierung

## 6.1 L3-XML und Lernpfade

Für die Texte der Lernmodule und die Integration der Medienobjekte wird die XML-DTD L3-XML [SLK00] verwendet und gegebenenfalls weiterentwickelt. Aus einer L3-XML-Beschreibung wird mit dem an der FH Lübeck entwickelten Werkzeug xml2html [SL01] eine Menge von vernetzten HTML-Seiten generiert.

Die De£nition von verschiedenen Lernpfaden wird in L3-XML explizit angegeben. Die Lernpfade werden durch xml2html auf entsprechende Verweisketten abgebildet.

Zur Unterstützung der Navigation und der Übersicht der Position in der Lerneinheit wird mit dem entwickelten Werkzeug eine Übersichtskarte über die ganze Lerneinheit mit entsprechenden Verweisen in einer sensitiven Map generiert. Desweiteren wird pro “Lerneinheitseite” eine Kurzgraphik generiert.

## 6.2 Auswahl Entwicklungswerkzeuge

Für die Beschreibung der Entwurfsmuster werden im wesentlichen die Medien Text, Bilder, Animationen, Video und Audio verwendet, wobei der Schwerpunkt auf Text, Bilder und Animationen 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.

#### 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 und PDF bzw. PostScript verwendet. Diese Formate sind zur Zeit 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 z.B. xemacs und Html-Kit verwendet, welches ein kostenloser Editor zur Erstellung jeglicher XHtml- bzw. XML-Dateien ist. Dort ist bereits der Html-Validator Tidy integriert, um die Korrektheit der Dateien sicherzustellen. 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 werden mehrere Entwicklungswerkzeuge verwendet, u.a. die Macromedia-Produkte Freehand 10 und Fireworks 4, da sie mit den übrigen verwendeten Programmen am besten kooperieren.

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. Gra£ken können optimiert und mit erweiterten Interaktivitätselementen versehen werden. Auch dieses Werkzeug arbeitet gut mit Flash zusammen, um dort die erstellten Graphiken in Animationen zu integrieren.

Für die Bearbeitung von Photos wird Photoshop von Adobe verwendet, das für Photobearbeitung hervorragend geeignet und weit verbreitet ist.

Die Studierenden benötigen für die Darstellung der Bilder selbst keine zusätzlichen Werkzeuge, außer den ohnehin verwendeten XHtml-fähigen Browser.

#### Entwicklungswerkzeuge für Animationen

Wie bereits schon angegeben, werden für die Erstellung der Animationen Flash 5 und in einigen Fällen auch Director 8.5 verwendet, die gut mit den übrigen Werkzeugen zusammenarbeiten können. Da beide Programme in der neusten Version auch für 3D-Objekte geeignet sind, werden gegebenenfalls 3D-Modelle bzw. -Animationen in der Lerneinheit eingesetzt.

| Zweck | Entwicklungswerkzeug | Ausgabedatei-Format |
|---|---|---|
| Erstellen der Lerneinheit | Html-Kit<br>xml2html (selbstentwickelt) | XML, XHtml, VRML<br>XHtml |
| Bildbearbeitung | Freehand 10, Fireworks 4, Gimp, x£g | eps, gif, jpeg, png |
| Photobearbeitung | Photoshop 6 | eps, gif, jpeg, png |
| Videobearbeitung | Premiere 6, After Effects 5 | avi, mpeg, Quicktime |
| Animationserstellung | Flash 5, Director 8.5,<br>3D Studio Max, Maya | <br>¤ash, schockwave, VRML |
| Hardcopy | Acrobat, LaTeX | pdf, ps |
| Sounderstellung | Sound Forge XP | wav, au, mpeg, mp3 |
| UML-Diagramme | ArgoUML, Poseidon, Together | xmi, gif |

Abbildung 1: Übersicht Entwicklungswerkzeuge

Für die Erstellung dieser 3D-Modelle werden die Entwicklungswerkzeuge 3D Studio Max und Maya verwendet, die beide einen Export£lter für Director integriert haben.

Um die Darstellung der Animationen bei den Studierenden sicher zu stellen, werden ein Flash-PlugIn und eventuell ein VRML-Viewer benötigt.

#### Entwicklungswerkzeuge für Video und Audio

Für die Erstellung und Bearbeitung von Videos wird das weit verbreitete WerkzeugPremiere von der Firma Adobe eingesetzt. Zur Optimierung der erstellten Videos wird After Effects von Adobe verwendet. Als Ausgabeformate sind die drei häu£g verwendeten Formate avi, mpeg und Quicktime vorgesehen. Zum Abspielen werden die dazu gehörigen Player (Medien-, Mpeg- und Realtime-Player) benötigt.

#### Zusammenfassung aller Entwicklungswerkzeuge

Die zur Erstellung der Lerneinheit verwendeten Entwicklungswerkzeuge werden zusammenfassend in Abbildung 1 aufgeführt.

#### Darstellungswerkzeuge

Es wird von den Studierenden folgendes zur Darstellung der Lerneinheit benötigt:

- XHtml-fähigen Browser
- Acrobat-Reader, eventuell GhostView
- Flash-PlugIn
- Java-PlugIn
- VRML-Browser
- Medien-, Mpeg- und Realtime-Player

## 6.3 Aufgabenverwaltung

Die Beispiele und Übungsaufgaben der zu entwickelnden Lerneinheit sollen zusätzlich mit einer Aufgabensammlung verwaltet werden. Der Prototyp einer solchen Aufgabensammlung wurde bereits in einer Diplomarbeit zum Thema “Parametrisierbare Aufgabensammlung für Online-Kurse” [Nis01] erstellt.

Mit diesem Prototyp ist es bereits möglich, Aufgaben der unterschiedlichsten Art zu verwalten und in einer Lerneinheit zu präsentieren. Es bietet dem Benutzer beziehungsweise der Benutzerin über eine Eingabeeinheit des Systems die Möglichkeit, die Daten einer Aufgabe in das System einzufügen, diese zu ändern, anzuzeigen oder zu löschen.

Mit Hilfe verschiedener Parameter, die bezüglich der Frage einer Aufgabe angegeben werden können, konstruiert das System aus einer Aufgabe unterschiedliche Aufgabenstellungen, die alle dem selben Muster entsprechen. Das System unterstützt MultipleChoice-Aufgaben, einfache Texteingaben und Drag&Drop-Aufgaben, sowie Freitexteingaben, Programmieraufgaben und Gruppenaufgaben. Durch die Vielzahl unterstützter Aufgabentypen und die Parametrisierbarkeit der Aufgaben gibt es diverse Variationsmöglichkeiten bei der Erstellung von Aufgaben zur Verbesserung der Ef£zienz der Lehre.

Das System ist mit zwei verschiedenen Datenbasen verbunden. Eine eigens für dies System entwickelte XML-Base und eine relationale MySQL-Datenbank, die auf der Anfragesprache SQL basiert, stehen in direkter Verbindung. Für die Nutzung der XML-Base liegt eine Spezi£kation der Aufgaben in Form einer DTD vor.

Das System ist ein ¤exibles System, das auf verschiedene Arten genutzt werden kann. Es existiert eine gra£sche Benutzungsober¤äche für die Eingabeeinheit, jedoch kann das System auch von der Konsole aus gestartet werden. Ebenso ist die Integration in anderen Programmen, wie zum Beispiel in einem Lerneinheiten-Generator wie xml2html, möglich. Diese Programme können Aufgaben nach einem bestimmten Muster oder einer bestimmten Aufgabennummer aus den jeweiligen Datenbasen holen und diese in die zu generierende Lerneinheit integrieren. Außer den Aufgabendaten liefert das System auch die Präsentation innerhalb der Lerneinheit. Die Darstellung der Aufgabe ist über weitere Parameter der Lerneinheit anpassbar.

Insgesamt bietet das System viele Möglichkeiten in Bezug auf den Umgang mit den Aufgabendaten und der Verwaltung von Aufgaben.

In den einzelnen präsentierten Lernmodulen werden die jeweils ausgewählten Aufgaben derart integriert, dass der Lernarbeitsplatz nur über einen HTML- und JavaScript-fähigen Browser hinsichtlich der Aufgaben verfügen muss.

# 7 Evaluierung

Die Lernmodule werden in einem ersten Test an der Fachhochschule Lübeck im 4. Semester in der Lehrveranstaltung Programmiertechniken der Studienrichtung Informatik des Studiengangs Elektrotechnik (genauer: Kommunikations-, Informations- und Medientechnik (KIM)) eingesetzt. In der Präsenzveranstaltung werden parallel zu den entwickelten Lernmodulen weitere Einheiten zur Programmierung in Java für C++-Kundige eingesetzt. Den inhaltlichen Kern der Lehrveranstaltung bilden allgemeine Programmiertechniken, und hier insbesondere

die Techniken, die als Entwurfsmuster beschrieben sind.

Sobald die ersten Lernmodule erstellt sind, werden sie in der Lehrveranstaltung eingesetzt und es £ndet eine erste formative Evaluation statt mit dem Ziel, die Qualität der Lernmodule zu verbessern.

Eine abschließende Evaluation ist für das zweite Halbjahr 2003 geplant, die nach einem im Projekt abzustimmenden Konzept durchgeführt wird.

# 8 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 zur Weiterbildung in der Phase des Selbststudiums.

Zur Entwicklung der Lerneinheit wird eine Vielzahl von Werkzeugen eingesetzt. Die Arbeitsumgebung der Studierenden jedoch besteht aus Standard-Werkzeugen, die für die Ausbildung kostenlos zur Verfügung stehen und einfach installierbar sein sollten.

# Literatur


[BMR+96] BUSCHMANN, FRANK, REGINE MEUNIER, HANS ROHNERT, PETER SOMMERLAD und MICHAEL STAL: *Pattern-Oriented Software Architecture - A System of Patterns*. Wiley, New York, 1996.

[Coo00] COOPER, JAMES W.: *Java Design Patterns, A Tutorial*. Addison-Wesley, 2000.

[GHJV95] GAMMA, ERICH, RICHARD HELM, RALPH JOHNSON und JOHN VLISSIDES: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

[Gra98] GRAND, MARK: *Patterns in Java*, Band 1. Wiley, 1998.

[Nis01] NISSEN, NINA: *Parametrisierbare Aufgabensammlung für Online-Kurse*. Diplomarbeit, FH Lübeck, FB Elektrotechnik, 2001.

[SL01] SEEHUSEN, SILKE und CARSTEN LECON: *Entwicklung von Online-Kursen für den längerfristigen Einsatz*. In: *Informatik 2001 Jahrestagung GI/OCG, Workshop Virtuelle Lernräume*, Seiten 1131–1135, 2001.

[SLK00] SEEHUSEN, SILKE, CARSTEN LECON und CAY KABEN: *Speci£cation of Learning Trails in Virtual Courses*. In: *FIE 2000, Frontiers in Education Conference, Kansas City, MO, USA*, 2000.

[SSRB00] SCHMIDT, DOUGLAS, MICHAEL STAL, HANS ROHNERT und FRANK BUSCHMANN: *Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects*. Wiley, 2000.

# MuSofT-Ergebnis-Bericht des Teilprojekts 2.3

Peter Aschenbrenner, Andy Schürr

## Inhaltsverzeichnis



## 1 Einleitung

Im Rahmen des MuSofT-Projektes hat das Teilprojekt 2.3 das Ziel, eine multimedial aufbereitete Lerneinheit zum Thema “Algorithmen und Datenstrukturen” zu entwickeln, 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. Man £ndet 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 wie z.B. [CCA01], [Ani01a], [Sor01], [Bub01]. Einen guten Überblick über den Stand der Technik vor einigen Jahren bietet [Sta98], einen guten Überblick über aktuelle Arbeiten in diesem Bereich £ndet man in [Dag01]. Allerdings handelt es sich in nahezu allen Fällen um Visualisierungen mit im wesentlichen fest vorgegebenem Ablauf, die entweder überhaupt keine Interaktionsmöglichkeiten oder keine ¤üssigen Animationen bieten. Damit wird ein Teil des möglichen Potentials der

multimedialen Unterstützung von Lehrveranstaltungen verschenkt [JTS93, ALS94, Whi97, PBS93, Mye90].

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 gra£schen 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 gra£sche Debugging selbst erstellter Programme von Studenten wird mithilfe von Visualisierungsbausteinen und automatischer Überprüfung von Invarianten von Datenstrukturen unterstützt.

Dafür wurde in der ersten Projektphase festgelegt, welche Inhalte in welcher Form und welchem Umfang für welchen Zuhörerkreis unterrichtet werden sollen. Die aktuelle Fassung wird in Abschnitt 2 bezüglich ihrer Inhalte vorgestellt, während sich Abschnitt 3 mit der Strukturierung und Anpassung an verschiedene Zuhörerkreise befasst. Diese Zuhörerkreise werden zusammen mit den ihnen zu vermittelnden Fähigkeiten in Abschnitt 4 näher beschrieben.

In einer zweiten inzwischen abgeschlossenen Projektphase wurde etwa ein Dutzend Werkzeuge evaluiert, die als Basis für die Entwicklung ¤üssiger, interaktiver Animationen komplexer Datenstrukturen dienen könnten. Bei dieser Evaluierung haben sich zwei Kandidaten herauskristallisiert, die wir im weiteren Projektverlauf einsetzen werden. Die sich daraus ergebenden Konsequenzen für die Lernumgebung der Studierenden werden kurz in Abschnitt 5 angerissen, weitere technische Realisierungsdetails werden im Abschnitt 6 diskutiert.

## 2 Inhalt

Die Vollversion der Lerneinheit 2.3 (vgl. Abschnitt 4.1) wird dem Studierenden im wesentlichen folgende Lehrinhalte vermitteln:

- 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 gra£sche 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 Programmveri£kation
- O-Kalkül zur Charakterisierung von Laufzeit- und Speicherplatzverhalten

Zur Motivation der behandelten Themen und zur Demonstration der Praxisnähe der Vorlesung werden wir ein durchgängiges Beispiel verwenden, die Bewältigung alltäglicher Probleme einer £ktiven Speditions£rma "Blitz AG". Such- und Sortieralgorithmen lassen sich am Beispiel terminlich gebundener Aufträge oder Kundenstammdaten diskutieren; für die Behandlung von Standardalgorithmen auf Graphen eignen sich die Planung einer Hierarchie von Verteilerzentren (Suche eines minimalen Spannbaumes), die Planung von Auslieferungsrouten von Verteilerzentren (Handlungsreisendenproblem), ...; für die Behandlung der dynamischen Programmierung kann man schließlich die Berechnung optimaler LKW-Beladungen als Variante des bekannten Rucksackproblems heranziehen.

Bei der Auswahl der oben genannten Themen sowie bei der Art der Präsentation haben und werden wir uns insbesondere auf folgende Quellen abstützen:

- das sehr kompakte und insbesondere für Ingenieurstudiengänge geschriebene Buch *Algorithmen und Datenstrukturen* [BOK99] von Bernd Owsnicki-Klewe, das in einem etwas unkonventionellen Schreibstil nahezu alle oben aufgeführten Themen (abgesehen von der gra£schen Modellierung von Datenstrukturen und dem systematischen Testen) abdeckt
- Teile der Kapitel II und III des gerade neu erschienenen Buches *Algorithmen und Datenstrukturen. Eine Einführung mit Java.* eines MuSofT-Projektpartners [GS01]
- das *Skriptum Informatik* [HJA00] von Hans-Jürgen Appelrath und Jochen Ludewig, das eine allgemeine Einführung in die Informatik mit einer knappen Darstellung eines Großteils der oben aufgeführten Themenstellungen verbindet

Als weiterführende Literatur empfehlen wir den Studierenden [AVA96], [Sed92], [Wir83], [Bal99], [AS01], [TO96].

# 3 Strukturkonzept

Entsprechend der Zusammensetzung der Studenten aus mehreren Gruppen mit unterschiedlichen Lernzielen (siehe Abschnitt 4) wird der Inhalt der Lerneinheit 2.3 in zwei Unterrichtsschienen LE2.3a und LE2.3b aufgeteilt (Abbildung 1), die jeweils aus fünf Lernmodulen bestehen. Die Lernmodule der Unterrichtsschiene LE2.3b setzen dabei meist Lerninhalte von

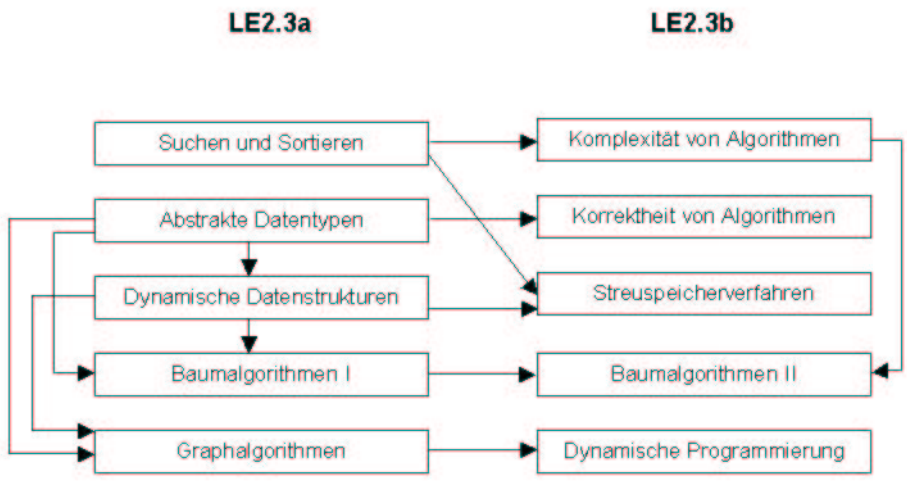



Abbildung 1: Aufteilung und Abhängigkeiten des Lerninhaltes in LE2.3a und LE2.3b

Modulen der Unterrichtsschiene LE2.3a voraus. Sie vertiefen und ergänzen die allen in Abschnitt 4.1 aufgeführten Studierenden vermittelten Lehrinhalte und sind für Studierende der Studiengänge Informatik und Wirtschaftsinformatik konzipiert.

Abbildung 1 zeigt, dass die gegebene Aufteilung in die zwei Zeitschienen keine starre Reihenfolge festlegt, sondern verschiedene Möglichkeiten zur zeitlichen Organisation und Auswahl der Lehrmaterialien offen lässt wie z.B.

- die Behandlung aller Kapitel in der kanonischen Reihenfolge,
- die Streichung bestimmter Kapitel aus LE2.3b,
- die verzahnte Unterrichtung bestimmter Kapitel, wie z.B. bei “Suchen und Sortieren” und “Komplexität von Algorithmen” oder
- die Konzipierung von LE2.3b als eigene Lehrveranstaltung, die entweder zeitlich leicht versetzt im selben Semester (Trimester) oder im folgenden Semester (Trimester) durchgeführt wird.

Die Abhängigkeitsstruktur zwischen den verschiedenen Lerninhalten £ndet sich auch innerhalb von Abschnitten wieder: Zum Beispiel besteht das Kapitel “Baumalgorithmen I” aus je einem Abschnitt für sortierte Binärbäume, AVL-Bäume usw. Jeder dieser Abschnitte besteht aus einer festen Anzahl von Unterabschnitten, die durch das Schema vorgegeben werden, gemäß dem alle Datenstrukturen präsentiert werden (wie z.B. Komplexität, Korrektheit

...). Einzelne dieser Unterabschnitte können aus Zeitgründen entfallen oder deshalb, weil die dafür notwendigen Voraussetzungen (noch) nicht geschaffen wurden.

Die Abhängigkeiten, die zwischen den Kapiteln bestehen, können noch genauer dargestellt werden als Abhängigkeiten zwischen Abschnitten innerhalb dieser Kapitel. Abbildung 2 zeigt einen Teil dieser verfeinerten Darstellung.

Gerade diese genaue Betrachtung der Abhängigkeiten hat gezeigt, dass wie oben beschrieben genügend Spielraum für die geforderte Variantenbildung der Lerneinheit 2.3 bleibt und wo dieser Spielraum ausgeschöpft werden kann (s.o.).

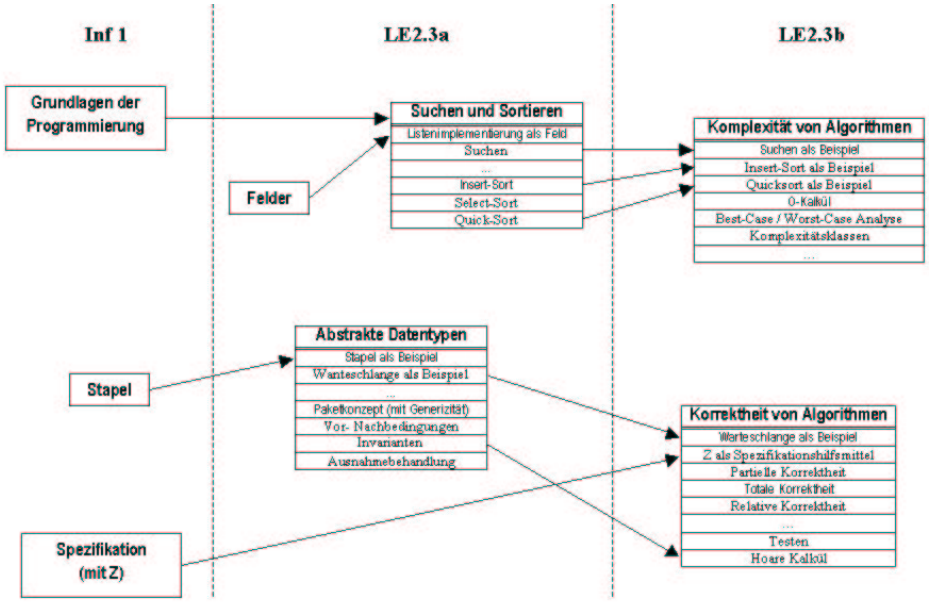



Abbildung 2: Teilmenge der verfeinerten Abhängigkeiten

# 4 Didaktisches Konzept

## 4.1 Leitbild

An der Universität der Bundeswehr wird die Lerneinheit 2.3 zuerst in der Vorlesung “Einführung in die Informatik II” (INF2) erprobt, die im Wintertrimester 2002 statt£ndet. Diese Vorlesung ist der 2. Teil einer dreisemestrigen Vorlesungsreihe im Grundstudium: In “Einführung in die Informatik I” (INF1) werden u.a. Grundlagen der Programmierung (mit verschiedenen Programmierparadigmen), grundlegende Begriffe zu Syntax und Semantik von Programmiersprachen sowie einfache Datentypen und Rechenstrukturen eingeführt. INF2 besteht — nach einem grundlegenden Redesign im Rahmen dieses Projektes — aus den Stof£nhalten, die be-

reits in Abschnitt 2 beschrieben wurden und um deren Unterrichtung es in der Lerneinheit 2.3 primär geht. In “Einführung in die Informatik III” (INF 3) wird die objektorientierte Programmierung vertieft und es werden sowohl Grundlagen der Softwaretechnik als auch der maschinennahen Programmierung vermittelt. Diese Vorlesungsreihe wird von drei Gruppen von Studenten[1] besucht:

1. Informatikstudenten nehmen an dem Vorlesungszyklus INF1, INF2 und INF3 in vollem Umfange teil.

2. Wirtschaftsinformatikstudenten nehmen an INF1 und INF2 ebenfalls in vollem Umfange teil, in INF3 aber nicht an den Lernmodulen zu maschinennaher Programmierung.

3. Elektrotechnikstudenten besuchen nur INF1 im vollen Umfange; in INF2 sind für diese Studenten nur die grundlegenden Lernmodule von Interesse, in INF3 überspringen sie die maschinennahe Programmierung.

Deswegen wird es die Lerneinheit 2.3 in zunächst zwei verschiedenen Ausbaustufen (Varianten) für (1) den Studiengang Informatik sowie (2) für andere Studiengänge mit Nebenfach Informatik geben; eine weitere Kompaktvariante (einsetzbar im Rahmen einer 2-stündigen Überblicksvorlesung über das Gebiet der Informatik) für Studierende anderer Fakultäten befindet sich in Planung. Als Voraussetzung für alle drei oben aufgeführten Gruppen von Studierenden genügen zunächst Grundkenntnisse der imperativen und objektorientierten Programmierung; für die Vollversion der Lerneinheit werden zusätzlich elementare Kenntnisse der mathematischen Logik bzw. diskreten Mathematik benötigt.

## 4.2 Lernziele

Auf die inhaltlichen Lernziele wurde bereits in Abschnitt 2 eingegangen. Auf einer etwas höheren Ebene geht es darum, den Studenten folgende Kompetenzen zu vermitteln:

- vertiefte Programmierkenntnisse und -erfahrungen beginnend mit präziser Beschreibung einer Aufgabenstellung über den Entwurf der Problemlösung hin zum systematischen Test oder Beweis der Korrektheit der gewählten Lösung

- Fähigkeiten zur Auswahl oder Anpassung geeigneter Algorithmen und Datenstrukturen aus dem Grundrepertoir der Informatik für eine gegebene Problemstellung

- ein bildliches Verständnis von Algorithmen und Datenstrukturen (hier versprechen wir uns einen besonderen Vorteil von den Animationen), und zwar durchgängig vom Entwurf mit UML über die Implementierung bis zum Testen und Debugging

## 4.3 Lernszenario

Das Angebot der klassischen akademischen Lehre wird in der Lerneinheit 2.3 zusätzlich zur generellen multimedialen Gestaltung besonders durch die Möglichkeit der Interaktion erweitert:

[1] An der Universität der Bundeswehr gibt es bislang keine Studentinnen.

- *Unterstützung von Lehrformen*: in erster Linie wird die Lerneinheit 2.3 so angelegt, dass die Lehrinhalte zunächst durch eine Vorlesung (oder Selbststudium) vermittelt und durch seminarartige (dreistündige) Übungen mit hohem Eigenanteil der Studierenden vertieft werden; ggf. können diese Lehrformen durch ein Proseminar ergänzt werden, in dem von den Studierenden nach einem vorgegebenen Schema weitere Algorithmen und Datenstrukturen präsentiert werden.

- *Einsatz durch die lehrende Person*: die Lerneinheit 2.3 soll sowohl als mediales Element der Präsenzveranstaltung (z.B. in der Vorlesung) eingesetzt werden als auch interaktiv (sequentiell und/oder erkundend hypermedial) zur individuellen oder teambezogenen Bearbeitung. Beim Einsatz vorgegebener Lernmodule und den dazu gehörigen Animationen im Rahmen neuer Lehrveranstaltungen sollen Anpassungen durch den Dozenten an eigene Vorlieben oder spezielle Erfordernisse ohne großen technischen Aufwand möglich sein.

- *Nutzung durch den Studierenden*: Der Student soll die Lerneinheit 2.3 beim Wissenserwerb (in der Vorlesung), bei der Vertiefung (in den Übungen und Seminaren) und in der Selbstlernphase (interaktives Erkunden) nutzen können.

- *Lehr-/Lernmaterial*: die Lerneinheit 2.3 wird in der Endausbaustufe ihren Nutzern folgende Lehrmittel zur Verfügung stellen:

  - für Vorlesungen: Folien, Präsentationen, Animationen von Datenstrukturen, Vorlagen für Illustrationen
  - für Übungen: Übungsaufgaben mit interaktiven Animationen, Vorschläge für traditionelle Übungsblätter, Demonstrations- und Debuggingumgebung
  - evtl. für Proseminare: Vorstellung einzelner Algorithmen und Beispiele für den Bau eigener Animationen, Demonstrations- und Debuggingumgebung

# 5 Lehr/Lernumgebung

Die Teilnehmer an einer Lehrveranstaltung, in der die Lerneinheit 2.3 als Ganzes oder einzelne ihrer Module eingesetzt werden, benötigen eine Laufzeitplattform, die neben den üblichen Werkzeugen für die Präsentation von Lehrmaterialien (Folien, Filmausschnitte, Web-Seiten, . . . ) eine Java-Entwicklungsumgebung, eine Laufzeitumgebung für Animationen und ein einfaches UML-CASE-Tool (oder einen Gra£keditor) enthält. Weitere Details zu den Voraussetzungen für die Entwicklung neuer Animationen und vor allem für den Einsatz von Animationen £ndet man im folgenden Abschnitt 6, der sich mit der Auswahl von Werkzeugen zur Erstellung interaktiver Animationen befasst. Für die Erstellung neuer Varianten der Lerneinheit 2.3 ist eine Plattform erforderlich, die

- die benötigten Dokumente für Lernmodule verwaltet,

- über Metadaten die Zuordnung dieser Dokumente zu Modulen und deren Zuordnung zu Lerneinheiten unterstützt,

- den Zugriff auf und die Erstellung von verschiedene(n) Varianten bzw. Versionen einer Lerneinheit für unterschiedliche Benutzerpro£le erlaubt

- und mindestens auch die Erfordernisse der Laufzeitplattform (s.o.) erfüllt.

# 6 Technische Realisierung

Bei der multimedialen Unterstützung der Lerneinheit 2.3 spielt die Erstellung kon£gurierbarer, interaktiv erforschbarer Animationen, die für die Bearbeitung von Übungsaufgaben und das Debugging erstellter Programme eingesetzt werden, eine herausragende Rolle.

Deshalb war die Suche nach geeigneten Animationsentwicklungs- und ablaufumgebungen ein zentrales Thema der ersten Projektphase. Um zielgerichtet nach geeigneten Animationswerkzeugen suchen und diese evaluieren zu können, wurde zuerst eine Liste von Minimalanforderungen in tabellarischer Form erstellt, die aus ca. 50 Unterpunkten bestand; diese Unterpunkte wurden dann in drei Gruppen verdichtet: *Eingabe-/ Ausgabe-Möglichkeiten*, *Entwicklungsumgebung* und *Rahmenbedingungen*. Details können hier nicht ausgebreitet werden. Für die Ermittlung dieser Anforderungen wurden zum einen eine ganze Reihe von Szenarien mit Standard- Anwendungsfällen durchgespielt, zum anderen die Zerlegung der bereitzustellenden Animationsumgebung in funktionale Einheiten mit entsprechenden Abhängigkeiten untersucht. Weitere Details £nden sich in einem ausführlichen Evaluationsbericht, der auf Anfrage zur Verfügung gestellt wird.

In der Gra£k (Abbildung 3) wird eine vereinfachte Sicht auf die Systemstruktur der Lerneinheit 2.3 dargestellt. Der Benutzer des Systems ist entweder ein Dozent, der für eine gegebene Animation neue Beispiele ausarbeitet oder die voreingestellte Art der Darstellung abändert, oder ein Studierender, der für die Durchführung einer Übungsaufgabe ein fertiges Animationsbeispiel benutzt oder die Animationsumgebung für das Debugging eigener Beispiele einsetzt.

Wenn ein Dozent eine neue Datenstruktur - etwa einen AVL-Baum - spezi£ziert, implementiert er die Datenstruktur als Graph und erlaubte (oder notwendige) Zustandsübergänge der Datenstruktur (wie Ausgleichs-Rotationen beim AVL-Baum) als “Graphtransformationen” (siehe Abbildung 3). Zur Zeit wird das Graphtransformationswerkzeug PROGRES [PRO01] (auf Linux) eingesetzt; ein späterer Wechsel zu Fujaba [Fuj01] (auf Windows) ist jedoch noch in Diskussion. Das Graphtransformationswerkzeug generiert ablauffähigen “Code für Datenstruktur-Editor, ...”.

Das “Programm der Studenten” ist ein ablauffähiges Programm in Java oder Ada, in dem an geeigneter Stelle Teile dieses generierten Codes (z.B. Schnittstellen-Operationen der Datenstruktur) aufgerufen werden. Der doppelte Pfeil zwischen den zwei beschriebenen Komponenten in Abbildung 3 deutet an, dass Rückgabewerte an das “Programm der Studenten” zurückgeliefert werden können. Der “Code für Datenstruktur-Editor, ...” basiert (für die Speicherung der jeweiligen Datenstruktur) im Falle von PROGRES auf einer Art “Graphdatenbank”.

Die “kon£gurierbare Visualisierung, ...” ist der Teil der Software, der im Rahmen der Lerneinheit 2.3 entweder entsprechend erweitert oder neu implementiert werden soll. Er soll in Zukunft nicht nur die interaktiv-explorative Visualisierung ermöglichen, sondern auch eine ¤üssige Form der Animation, die wir z.B. für die Darstellung einer Doppelrotation eines AVL-

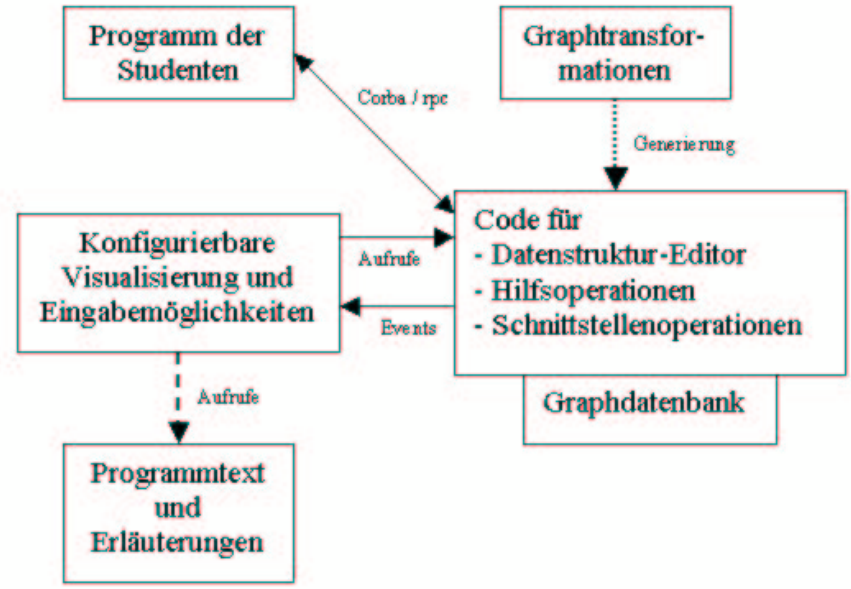



Abbildung 3: Lerneinheit 2.3 - Bestandteile der Animationsumgebung

Baumes für besonders wichtig halten. Dabei werden einfache Datenstrukturen wie Listen und Bäume als Spezialfälle von Graphen aufgefasst, so dass man die hier entstehende Funktionsgruppe auch “Graphbrowser” nennen könnte. Die zentrale Visualisierungskomponente wird eine zur Animation parallele Anzeige von “Programmtext und Erläuterungen” steuern, womit beispielsweise der passende Programmcode, die Anzeige zugehöriger HTML Seiten oder die Ausgabe von Fehlermeldungen gemeint sein kann.

Als technische Realisierung der Schnittstelle zwischen dem “Programm der Studenten” und dem generierten “Code für Datenstruktur-Editor, ...” werden Corba oder Java RPC ins Auge gefasst, weil auch ein verteilter Einsatz (z.B. “Programm der Studenten” auf Windows, “Graphdatenbank” auf Linux) unterstützt werden soll. Die Kommunikation zwischen dem generierten “Code für Datenstruktur-Editor, ...” und der Visualisierungskomponente kann über Eventsteuerung und Aufrufe geschehen.

Der Grobentwurf des beschriebenen Systems ergab verschiedene Kriterien, nach denen folgende Werkzeuge zuerst ausgewählt und danach einer näheren Analyse unterzogen wurden: PROGRES [PRO01] , Animal [Ani01b] , Stagecast [Sta01] , Fujaba [Fuj01], Amulet [Amu01], Tilcon [Til01] , Simul8 [Sim01], AGD [AGD01], DDD [DDD01] , yFiles [YFi01].

Dabei stellte es sich heraus, dass das Graphtransformationssystem PROGRES und die Graphenanimationsumgebung yFiles am ehesten die aufgestellten Anforderungen für die Entwicklung von Animationen erfüllen. Möglicherweise wird eine Kombination beider Systeme sinnvoll sein, in der die bislang verwendete kommerzielle Graphvisualisierungsbibliothek JViews [JVi01] in PROGRES durch yFiles ersetzt wird. Im Rahmen einer weiteren Evaluation von PROGRES und insbesondere von yFiles werden wir in der nächsten Projektphase dieser Frage anhand einiger ausgewählter Beispiele für Datenstrukturanimationen weiter nachgehen. Weitere Details zur Evaluation der oben genannten Systeme und umfangreiches dabei gewonnenes tabellarisches Material können in einem in Kürze erscheinenden Bericht nachgelesen werden.

# 7 Evaluierung

Die erste Fassung der Lerneinheit 2.3 kommt bereits im Wintertrimester 2002 an der UniBw München zum Einsatz. Anstelle von Java (der Sprache, auf die man sich in MuSofT geeinigt hat) wird in der Vorlesung “Einführung in die Informatik II” jedoch die Programmiersprache ADA verwendet. Das heißt, dass die Lerneinheit so konzipiert wird, 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. Auf diese Weise wird vor der Fertigstellung der ersten Java-Fassung der Lerneinheit für das Projekt MuSofT schon eine Erprobung ihrer Konzeption statt£nden. Die ersten Animationen werden noch meistens passiv sein (in der Vorlesung) oder auf relativ einfache Interaktionsbeispiele mit Sortieralgorithmen beschränkt bleiben, die im Internet frei verfügbar sind.

Gemäß der Planung soll im nächsten Jahr eine Lehrveranstaltung mit den neu zu entwickelnden, eigenen Animationen wiederholt werden. Der Einsatz an anderen Universitäten, z.B. den Projektpartnern ist in Diskussion. Nachdem sich die Stoffauswahl der Lerneinheit u.a. an [GS01] anlehnt, ist mit der Universität Magdeburg eine Kooperation oder Übernahme am problemlosesten vorstellbar.

# 8 Zusammenfassung

Bisher wurden die Lerninhalte der Lerneinheit 2.3 - auch in ihrer Struktur und ihren Abhängigkeiten - spezi£ziert und nach einer ausführlichen Evaluierung einer Menge von Werkzeugen zur Animationsunterstützung zwei Kandidaten gefunden: PROGRES und yFiles.

Die nächsten Schritte sind die Erstellung einer Level-0 Fassung der Lehrmaterialien (Vorlesungsfolien, Übungsaufgaben, einfache fertige Animationen) inklusive einer weiteren Ausarbeitung des Fallbeispiels der £ktiven Speditions£rma “Blitz AG” und, parallel dazu, die weitere Evaluierung von PROGRES und yFiles anhand von ausgewählten Beispielen für Datenstrukturanimationen.

## Literatur


[AGD01] http://ftp.mpi-sb.mpg.de/AGD/, Dez. 2001.

[ALS94] A. LAWRENCE, A. BADRE und J. STASKO: *Empirically evaluating the use of animations to teach algorithms.* In: *Proceedings IEEE Symposium on Visual Languages*, Seiten 48–54, 1994.

[Amu01] http://www-2.cs.cmu.edu/afs/cs/project/amulet/www/amulet-home.html, Dez. 2001.

[Ani01a] http://www.informatik.uni-siegen.de/db/animations.php3?lang=en, Dez. 2001.

[Ani01b] http://www.informatik.uni-siegen.de/~inf/Software/Animal/indexDE.html, Dez. 2001.

[AS01] ANDREAS SOLYMOSI, ULRICH GRUDE: *Grundkurs Algorithmen und Datenstrukturen. Eine Einführung in die praktische Informatik mit Java.* Vieweg Verlag, 2001.

[AVA96] ALFRED V. AHO, JEFFREY D. ULLMAN: *Informatik. Datenstrukturen und Konzepte der Abstraktion.* International Thomson Publishing, 1996.

[Bal99] BALZERT, HELMUT: *Lehrbuch Grundlagen der Informatik.* Spektrum Akademischer Verlag, 1999.

[BOK99] BERND OWSNICKI-KLEWE, BERND WISSNER: *Algorithmen und Datenstrukturen.* Wißner-Verlag, 1999.

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

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

[Dag01] http://www.dagstuhl.de/DATA/Seminars/01/#01211, Dez. 2001.

[DDD01] http://www.gnu.org/software/ddd/, Dez. 2001.

[Fuj01] http://www.fujaba.de/, Dez. 2001.

[GS01] GUNTER SAAKE, KAI-UWE SATTLER: *Algorithmen und Datenstrukturen. Eine Einführung mit Java.* dpunkt-Verlag, Heidelberg, 2001.

[HJA00] HANS-JÜRGEN APPELRATH, JOCHEN LUDEWIG: *Skriptum Informatik. Eine konventionelle Einführung.* Teubner Verlag, 2000.

[JTS93] J. T. STASKO, A. M. BADRE, C. LEWIS: *Do algorithm animations assist learning ? An empirical study and analysis.* In: *PROCEEDINGS INTERCHI'93 Conference on Human Factors in Computer Systems*, Seiten 61–66, 1993.

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

[Mye90] MYERS, B.A.: *Taxonomies of Visual Programming and Program Visualization*. Journal of Visual Languages and Computing, 1(1):97–123, 1990.

[PBS93] PRICE, BLAINE A., RONALD M. BAECKER und IAN S. SMALL: *A Principled Taxonomy of Software Visualization*. Journal of Visual Languages and Computing, 4(3):211–266, 1993.

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

[Sed92] SEDGEWICK, ROBERT: *Algorithmen*. Addison-Wesley, 1992.

[Sim01] `http://www.simul8.com/`, Dez. 2001.

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

[Sta98] STASKO, JOHN: *Software Visualization*. MIT Press, 1998.

[Sta01] `http://www.stagecast.com/`, Dez. 2001.

[Til01] `http://www.tilcon.com/`, Dez. 2001.

[TO96] THOMAS OTTMANN, PETER WIDMAYER: *Algorithmen und Datenstrukturen*. Spektrum Akad. Vlg., Hdg., 1996.

[Whi97] WHITLEY, K. N.: *Visual Programming Languages and the Empirical Evidence For and Against*. Journal of Visual Languages and Computing, 8:109–142, 1997.

[Wir83] WIRTH, NIKLAUS: *Algorithmen und Datenstrukturen*. Teubner Verlag, 1983.

[YFi01] `http://www.yworks.de/`, Dez. 2001.

# Arbeiten zu LE 3.1 ”V-Modell” und LE 3.2 ”Qualitätsmanagement” in 2001

Fritz Schmidt

**Kurzfassung:**

Die Lehreinheiten 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 Tayloring auf aktuelle Projekte angepasst werden können. Dadurch können die Lernmodule in den verschiedensten Kontexten wiederverwendet werden. Die Lernmodule werden ergänzt durch Beispielanwendungen, die mit den Systemen erzeugt wurden.

**Abstract:**

Teaching units 3.1 and 3.2 of the project MuSofT deal with quality management (QM) in software engineering. Teaching unit 3.1 concentrates on the process of software development. Teaching 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. Links are provided to systems which can be taylored to handle concrete projects. Most examples are generated using these systems. As a consequence the learning modules can be reused in many different contexts.

## 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, sind prozeßorientiertes und produktorientiertes Qualitätsmanagement notwendig.

Unter **prozeßorientiertem** Qualitätsmanagement meinen wir den Erstellungsprozeß der Software. Ihm ist LE 3.1 ’V-Modell’ gewidmet. Er umfaßt Methoden, Werkzeuge, Richtlinien und Standards. Sie bilden sowohl eine Unternehmens- als auch eine Projektkultur. Ziel ist es, den Produktionsprozeß hin zu einer optimalen Qualität zu verändern. Wir gehen also von einem dynamischen Qualitätsoptimum aus. Der Qualitätsprozeß dient der permanenten Adaptierung.

Unter **produktorientiertem** Qualitätsmanagement verstehen wir, Software-Pro-dukte und Zwischenergebnisse auf vorher festgelegte Qualitätsmerkmale zu überprüfen. Dies ist Schwerpunkt der LE 3.2 ”Qualitätsmanagement”. Bei Anwendungs-Soft-ware gehören Gütebedingungen und Prüfbestimmungen dazu. Diese können Gegenstand einer Zerti£zierung sein, d.h. einer Bestätigung durch anerkannte (akkreditierte) Stellen, daß bestimmte Normen eingehalten werden.

Die Lehreinheiten basieren primär auf unseren Erfahrungen bei der Durchführung von Qualitätsmanagement-Maßnahmen bei großen Industrie Projekten. Diese Erfahrungen werden ergänzt durch theoretische Überlegungen, die der in Kapitel 9 angegebenen Literatur entnommen wurden. Dabei wurden nur dann eigenständige Formulierungen gewählt, wenn dies unseren pädagogischen Zielen dienlich erschien.

Es sind insgesamt 14 Lehrmodule geplant. Zielgruppe sind Ingenieure des Faches ”Angewandte Informatik” und Informatiker mit Nebenfach Maschinenbau. Die Lehrmodule können zum Teil einzeln verwendet werden und in andere Vorlesungen eingebaut werden. Sie sind zunächst für Präsenzveranstaltungen gedacht, sollen es aber auch erlauben, Kenntnisse aus vergangenen Vorlesungen aufzufrischen (beru¤iche Weiterbildung der Alumni). Durch die Internetpräsentation sollen folgende Ziele erreicht werden:

- Bearbeitung des Stoffes unabhängig von der Vorlesung (z.B. bei Abwesenheit von Studenten oder zum Auffrischen)
- Erweiterung der Präsentationen um die Dimensionen Zeit (Abläufe), Raum (Hintergrund und Links) und Interaktion (Experimente).

- Verbindung zu Prozessmodellen und Testsystemen, die an konkrete Projekte angepasst werden können.

In der Typisierung von MuSofT handelt es sich vor allem um level 0 Module (Übersicht), die durch Anwendungsteile ergänzt werden. Deswegen werden die Materialien so zur Verfügung gestellt, daß sie leicht in andere Veranstaltungen integriert oder für eine konkrete Veranstaltung neu kombiniert werden können.

# 2 Inhalt der Lerneinheiten "V-Modell" und "Qualitätsmanagement"

Wie schon einleitend gesagt sind 14 Lehrmodule in zwei Lerneinheiten vorgesehen. In diesem Kapitel werden kurz ihre Inhalte beschrieben und eine mögliche Anordnung in einer Vorlesung vorgestellt. Diese Anordnung unterscheidet sich vom ursprünglichen Plan und ist ein erstes Ergebnis der Evaluierung des Konzeptes anhand ausgewählter Teile in einer Vorlesung im Wintersemester 01/02.

**Lehrmodule 1 - 7 [LE 3.1]**

1. Fehler und ihre Kosten

   Software ist wie jedes Ingenieurprodukt fehlerbehaftet. Fehler verursachen Kosten und müssen daher auf ein Minimum beschränkt werden. Dies erreicht man durch Qualitätssicherung beim Prozess der Softwareerstellung (Prozeßqualität) und durch Prüfung der Software (Produktqualität). Bei der Prüfung großer Softwaresysteme ist nur das Auftreten von Fehlern festzustellen. Ein Nachweis der Fehlerfreiheit ist nicht möglich (Parnas).

2. Prozeßqualität und Produktqualität

   Unter der Annahme, daß ein Nachweis der Fehlerfreiheit nicht möglich ist, werden prinzipielle Wege zur Erzielung von Prozeß- und Produktqualität vorgestellt

3. Das Capability Maturity Modell (CMM)

   Das Capability Maturity Modell (CMM) ist eines der wichtigsten internationalen Modelle zur Erzielung vonVerbesserung der Prozeßqualität. An Hand dieses Modelles werden Grundüberlegungen zur Prozeßqualität erläutert. erläutert. Modi£kationen des Modells hin zum Personal Capability Maturity Modell (P-CCM) und zum Personal Software Process (PSP) zeigen praktische Relevanz.

4. Das V-Modell im Überblick

   Das V-Modell ist der Entwicklungsstandard für IT-Systeme des Bundes. Er besteht aus drei Teilen:

   - **Vorgehensmodell** (Was ist zu tun?),
   - **Methodenzuordnung** (Wie ist etwas zu tun?)

- **Funktionale Werkzeuganforderungen** (Womit ist etwas zu tun?)

Kern des Modelles ist die Beschreibung des IT-Entwicklungsprozesses über ein **Vorgehensmodell** (Prozessmodell), wofür abkürzend der Begriff ”V-Modell” benutzt wird. Im Begriff ”V-Modell” werden die Teile Methodenzuordnung und funktionale Werkzeuganforderungen mit eingeschlossen. Im V-Modell wird der Entwicklungsprozeß als eine Folge von Tätigkeiten, **Aktivitäten**, und deren Ergebnissen, den **Produkten**, beschrieben. Das V-Modell wurde zunächst für für strukturierte Methoden entwickelt und erst im V-Modell 97 um Möglichkeiten der Objektorientierung ergänzt. Es kann daher sowohl auf Projekte, die nach dem Wasserfallmodell als auch auf solche, die ein iterativ-inkrementelles Vorgehen erfordern, zugeschnitten werden.

5. V-Modell - Anwendungen

   Es wird die Anwendung des V-Modelles für die Erstellung grosser Software Pakete erläutert. Basis ist Erstellung eines Projekthandbuches am Beispiel des Projektes Integriertes Mess- und Informationssystem des Bundes (IMIS). Als Prozessmodell wurde ein iterativ-inkrementelles Vorgehen nach dem Rational Uni£ed Prozess (RUP) ausgewählt.

6. Prozessbeschreibung SSDA

   Es wird die Anwendung des V-Modelles für die Erstellung von Seminar-, Studien- und Diplomarbeiten erläutert. Prozessmodell ist das Phasenmodell in seiner Ausprägung Wasserfallmodell. Ziel der Anwendung ist es studentische Arbeiten transparenter zu machen und das Risiko ihres Scheiterns zu verringern. Durch Anwendung eines Vorgehensmodelles auf ein studentisches Projekt sollen gleichzeitig Aufwand und Changen modernen Qualitätsmanagements vermittelt werden. Einen ersten Satz von Ablaufdiagrammen und Dokumenten £ndet man unter folgendem Link:

   `http://www-is.ike.uni-stuttgart.de/intra/q-management/index.html`

7. Der Rational Uni£ed Prozess im V-Modell

   Das V-Modell ist in der Lage, auch objektorientiertes Vorgehen wie es etwa im UP vorgeschlagen wird, zu unterstützen. Es wird gezeigt, wie solch ein Tayloring zu erfolgen hat und wie die Entwickler- und die Managersicht trotzdem vereinigt werden können. Folgende Schwerpunkte sind vorgesehen

   - Rational Uni£ed Process (RUP)
   - Alternative Prozessansätze
   - Die 6 Grundprinzipien des RUP und die Zusammenhänge zum V-Modell
   - Integrationsmöglichkeiten des RUP in das V-Modell ‘97
   - Tailoring
   - Pragmatischer Ansatz zur Durchführung großer Projekte

**Lehrmodule 8 - 14 [LE 3.2]**

8. ISO-Normen

   Die ISO 9000-Familie beschreibt einen Rahmen, um ein QM-System in einer Organisation einzuführen und zu betreiben. Die Einführung eines QM-Systems macht es notwendig, sich über alle Vorgänge, Verantwortlichkeiten, Verhaltensweisen und Einstellungen der Mitarbeiter klar zu werden. Diese Dinge müssen offengelegt und dokumentiert werden. Dabei stellt sich oft heraus, daß einzelne Maßnahmen nicht angemessen sind und geändert werden müssen. Dies ist eine große Chance für jedes Unternehmen, seine Abläufe zu optimieren. In der neuen ISO 9000 ( Fassung 2000) liegen die Schwerpunkte bei Zufriedenstellung der Kunden, Prozessmodellierung und Weiterentwicklung der Qualitätsstandards eines Unternehmens. Für die Qualität der Softwareentwicklung ergeben sich dadurch die Schwerpunkte Abnahmetest (LM 13), Vorgehensmodell (LM 4) und Entwicklungsmodell (LM 3). Lehrmodul 8 hat als ein Ziel, diese Lehrmodule vorzubereiten.

9. Prozessmodelle Übersicht

   Prozessmodelle - häu£g auch Vorgehensmodelle genannt - haben zum Ziel, den Prozess der Entwicklung von Softwaresystemen zu strukturieren und planbar zu machen. Sie bilden damit die Grundlage des prozessorientierten Qualitätsmanagements. Durch Tayloring kann aus einem Prozessmodell der organisatorische Rahmen der Softwareentwicklung innerhalb eines konkreten Projektes entwickelt werden. Anhand des Wasserfallmodells werden die grundlegenden Festlegungen eingeführt (Aktivitäten, Produkte einschliesslich Layout und Qualitätskriterien, Quali£kationen, Rollen und Entwicklungsumgebung). Es werden die Schwächen des Phasenmodells aufgezeigt und alternative Modelle skizziert.

10. Prüfung von SW-Komponenten - Einzeltests

    Ergebnisse der Aktivitäten eines Prozessmodelles sind Produkte, die vorgegebenen Qualitätsmerkmalen genügen müssen. Handelt es sich bei den Produkten um Software, so erfolgt der Nachweis des Erreichens der Qualitätsziele durch Tests. Es werden die Grundüberlegungen des Testens von kleinen Softwareeinheiten (Klassen, ADT-Module, Fkt-Module) erläutert. Dabei gehen wir von den Software Qualitätsmerkmalen der ISO 9126 aus

11. Prüfung von komponentenbasierten Systemen - Integrationstests

    Die Produktqualität komponentenbasierter Systeme wird durch die Qualität der Komponenten und der Beziehungen zwischen den Komponenten im System bestimmt. Die Prüfung der Produktqualität eines SW-Systems erfolgt zeitlich nacheinander in 3 Teststufen: Integrationstest, Systemtest und Abnahmetest. In diesem Modul wird die Problematik der Integration und der Systemtests diskutiert.

12. Prüfung von Simulationsprogrammen - Funktionstests

    Durch den Funktionstest wird überprüft, ob alle in der Produktde£niton geforderten Funktionen vorhanden und wie vorgesehen realisiert sind. Aus dem P¤ichtenheft werden die Testsequenzen übernommen und/oder mit funktionalen Testverfahren hergeleitet. Funktionstests sind daher häu£g Bestandteil des Systemtests. In der Lehreinheit liegt

der Schwerpunkt auf dem Test von Simulationskomponenten (Funktions-Modulen). Ihr Einsatz ist nur bei Einhaltung der ihnen zugrundeliegenden Modellannahmen und bei bereitstellung konsistenter Daten möglich. Um dies zu gewährleisten muss man die dem Modul zugrundeliegende Ontologie kennen und daraus Aussagen zur Semantik seiner Schnittstelle machen

13. Testumgebungen - Abnahmetests

    Die Entwicklung grosser Softwaresysteme endet in der Regel mit einer Werkabnahme. Dadurch wird nachgewiesen, dass in einer gemeinsam mit dem Auftraggeber de£nierten Umgebung die vertraglich festgelegten Funktionalitäten des Systemes realisiert wurden. Die Funktionalitäten werden über Testitems beschrieben. Testprozeduren beschreiben darauf aufbauend die zum Test notwendigen Voraussetzungen, die Durchführungsschritte und das erwartete Verhalten. Im Lehrmodul wird versucht die Verbindung des Abnahmetests mit der Spezi£kation zu verdeutlichen und Erfahrungen aus Werkabnahmen grosser objektorientierter Systeme zu vermitteln.

14. Risikomanagement

    Risikomanagement versucht die Auswirkungen heutiger Entscheidungen auf die Zukunft abzuschätzen. Es verbessert die Wahrscheinlichkeit, die Softwareprodukte dem Kunden mit den richtigen Inhalten in der geforderten Qualität und im geplanten Budget zu liefern. Alle Risiken haben Ein¤uss auf 3 elementare Risiko-Elemente:

    - Zeitrahmen
    - Kostenrahmen
    - Funktionalität/Qualität

    Es werden Verfahren zur Minimierung dieser Risiken vorgestellt

# 3 Didaktisches Konzept

Im Rahmen der Vorarbeiten zu MuSofT haben wir gemeinsam mit dem Institut für Berufs-, Wirtschafts- und Technikpädagogik der Universität Stuttgart eine ausführliche Studie über Möglichkeiten computerunterstützten Lernens durchgeführt. Dort £nden sich auch Vorschläge für eine didaktisch sinnvolle Verwendung der neuen Medien. Die Ergebnisse dieser Studie sind im Netz verfügbar (Möglichkeiten computerunterstützten Lernens: `http://www.ike.uni-stuttgart.de/~www_wn/lehre`). Hier werden einige der daraus gezogenen Schlussfolgerungen für unsere Beiträge zu MuSofT dargestellt.

## 3.1 Multimedia-Technologien zur verbesserten Aufbereitung von Lehrinhalten

Multimediatechnologien erlauben es, Lehrinhalte neu aufzubereiten und dem Lernenden in einer seinen Kenntnissen und Bedürfnissen angepaßten Form zu vermitteln.

Die Einführung von Multimediatechnologien setzt die Verwendung des PC in der Ausbildung voraus. Der Rechner ist zugleich Anbieter von Lehrmaterial und seiner Experimentierung, Bibliothek, Auskunftsterminal und Kommunikationszentrum. Zum Lehrmaterial gehören Multimedia-Kurse(interaktive), Videos, Computer Based Training, Simulationspakete, Experimentiersoftware, Animationen. Sie sollen die bisher üblichen Kursunterlagen ergänzen.

Durch den Einsatz neuer Medien kann die Qualität der Ausbildung erheblich gesteigert werden. Voraussetzung dafür ist die Aufarbeitung der Lehrmaterialien im Hinblick auf den Online-Einsatz und die konsequente Nutzung der Möglichkeiten der neuen Technologien unter didaktischen Gesichtspunkten. Unter diesem Aspekt sind folgende Punkte von Bedeutung:

- Hierarchisierung der Wissensdarstellung.

  Die bisherige lineare Darstellung des Wissens (Bücher, Kurse) kann über Verknüpfungen (Links zu erweiterten Darstellungen) hierarchisiert werden. Solche Möglichkeiten waren bisher- etwa über Leseempfehlungen - nur ansatzweise möglich. Sie erlauben es, dem Lernenden seinen Fähigkeiten und Kenntnissen entsprechend in einem Kurs vor- oder nachzuarbeiten.

- Dynamisierung der Wissensdarstellung

  Die Präsentation des Wissens kann nicht nur textuell, sondern auch im Rahmen von Videos und Computerexperimenten (Spielen) erfolgen. Das ist vor allem für solche Helfer wichtig, die nicht nur über den Kopf, sondern auch aus Erfahrung lernen. Filme, Experimente und Spiele waren schon bisher Bestandteile erfolgreicher Ausbildung, mußten in der Regel aber getrennt von der Theorie vermittelt werden.

- Aktualisierung der Wissensdarstellung

  Unser Wissen wächst ständig und muß immer wieder unter neuen Gesichtspunkten (Aufgaben) dargestellt werden. Über Printmedien ist dies aufwendig (z.B. Neuau¤age eines Textes). Elektronische Formen erlauben eine ¤exible, aktuellere und teilnehmerbezogenere Darstellung von Ausbildugnsinhalten. -

- Verteilung der Wissensdarstellung

  Inhalte von Kursen beruhen häu£g auf Erfahrungen verschiedener Lehrender. Daher müssen wesentliche Inhalte oft durch Zitate belegt werden. Ähnlich wie bei der Hierarchisierung lassen sich Links zu Sekundärquellen einfügen, so daß der Lernende bei Bedarf schnell Zugang zu erweiterten Darstellungen hat.

Kurse, die solche Techniken einsetzen, können auf CD-ROM oder über das Internet zur Verfügung gestellt werden.

## 3.2 Umsetzung in den Lernmodulen

### 3.2.1 Leitbild

Leitbild sind Studierende und Alumni primär der Ingenieurwissenschaften, die schon soviel Grundkenntnisse im Umgang mit Rechnern haben, dass Sie in grösseren Softwareprojekten eine wesentliche Rolle übernehmen können.

### 3.2.2 Lernziele

Lernziel ist dann diese Kunden mit Grundkenntnissen des Qualitätsmanagements bei der Softwarentwicklung vertraut zu machen und Sie in die Lage zu versetzen, diese Grundkenntnisse aufzufrischen, bzw. sie in konkreten Projekten umzusetzen.

### 3.2.3 Lernszenario

Ziel ist nach wie vor die Präsenzveranstaltung, die aber in den Dimensionen Breite, Tiefe und Zeit erweitert wird, um lebenslanges Lernen zu ermöglichen

### 3.2.4 Didaktischer Ansatz

Im Rahmen der Lehreinheiten von MuSofT setzen wir muldimediale Technologien eher spärlich ein. Insbesondere verzichten wir weitgehend auf selbsterklärende Animationen. 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 strukturierte Lernmodule und ein Konzept zu ihrer Integration in grössere Lehreinheiten. Die Kapselung der Lernmodule erreichen wir dadurch, dass wir uns im Sinne der MuSofT Typisierung auf den Leveln 0 und 1 bewegen. Die Vertiefung wird dann über links zu Systemen, die sich auf aktuelle Probleme anwenden lassen, erreicht. Dies erfordert eine klare Trennung zwischen Lernmodulen und Anwendungsbeispielen. Beides zusammen ermöglicht es, die Lernmodule in verschiedenen Kontexten zu verwenden und durch Wahl der Beispiele oder gezielte Vertiefungen einzelner Unterthemen - etwa durch Einsatz der über links angeschlossenen Systeme - an die Interessen und Möglichkeiten der Lernenden anzupassen. Die Flexibilität bei der Integration ermöglicht uns das im Kap. 6 vorgestellte Werkzeug AIDA. Zur Unterstützung der Selbstkontrolle des Lernfortschrittes der Lernenden werden multiple choice Fragebögen zur Verfügung gestellt.

## 3.3 Umsetzung in den Beispielen

Aus Sicht der hier diskutierten Lehreinheiten müssen eine Reihe von Anforderungen an die Übungen gestellt werden. Ziel ist es den Nutzen der Anwendung aufwendiger Techniken im Rahmen des Software Entwicklungs Prozess und der Qualitätssicherung einsichtig zu machen. Dazu gehören:

**Komplexität** Das Beispiel muß aufwendig genug sein, um Projektmanagement, Vorgehensmodell, Softwarearchitektur, Datenbankanschluß usw. als notwendig erkennbar zu machen (Motivation).

**Anschaulichkeit** Das Beispiel darf nicht von der eigentlichen Aufgabe, der Vermittlung von Erfahrungen im Software Engineering, ablenken. Inhalt und daraus abgeleitetes P¤ichtenheft müssen leicht vermittelbar und visualisierbar sein.

**Durchgängigkeit** Das Beispiel sollte es erlauben, möglichst viele Aspekte des SE anzusprechen und dazu Übungen abzuleiten.

**Wiederverwendung** Wiederverwendung ist eine der Schlüsselmotivationen für komponentenorientiertes Software Engineering. Die Beispiele müssen daher so gestaltet werden, dass sie dazu anregen bereits existierende oder von einer anderen Gruppe parallel entwickelten Lösungen von Teilproblemen mitzuverwenden.

**Beispiel 1 Simulation des thermischen Verhaltens von Gebäuden**

Unser Anwendungsbeispiel für Entwicklung (PM) und Prüfung von Software (QM) ist die Simulation des thermischen Verhaltens von Gebäuden (Ziel: Wärmeschutznachweis) und die Simulation des Verhaltens von Anlagen der technischen Gebäudeausrüstung im Wechselspiel mit dem Gebäude ( Ziel: Optimierung des Betriebes). Das zugrundeliegende Problem lässt sich wie folgt beschreiben: Die Wärmeschutzverordnung verlangt den Nachweis, daß bei neu zu erstellenden Gebäuden die Energieverbräuche bestimmte Grenzen nicht überschreiten. Der Energieverbrauch wird über Jahresbilanzen bestimmt. Nach ISO EN 832 berechnet sich der Wärmebedarf eines Gebäudes aus den Transmissionsverlusten und den Lüftungsverlusten abzüglich der solaren und internen Wärmegewinne. Es gibt eine Vielzahl von Möglichkeiten, diese Vorgänge zu modellieren. Die Folien unter `http://www.ike.uni-stuttgart.de/~www_wn/lehre` beschreiben das im Rahmen eines Praktikums im WS 1998 verwendete Modell.

**Beispiel 2 Studentische Projekte**

Unsere Beispiele für Anwendungen des Vorlesungstoffes zum V-Modell sind

1. Projekt SSDA (Studienarbeit, Semesterarbeit, Diplomarbeit)
2. Projekt Seminar
3. Projekt Erstellung Lernmodul

Die Beispiele sind unter derselben Seite, die beim Vorlesungsstoff angegeben ist, zu £nden: `http://www-is.ike.uni-stuttgart.de/intra/q-management/index.html`

# 4 Strukturelles Konzept

Das geplante Vorgehen erlaubt es das Strukturelle Konzept erst bei der Zusammenstellung eines konkreten Kurses festzulegen. Das ist im Ingenieurbereich insofern sinnvoll, als dort die Schwerpunkte entsprechend den Interessen der Lernenden gelegt werden müssen. Dem kommt die Aufteilung in Lehrmodulen des Levels 0 und des Levels 1 entgegen. Sie erlaubt es zusammen mit den in der elektronischen Form verfügbaren Werkzeugen, einzelne Lehrmodule in Praktika zu übernehmen oder zum vertiefenden Selbststudium zur Verfügung zu stellen. Dazu hilft das im 6. Kapitel vorgestellten Werkzeug AIDA. AIDA erlaubt es zum einen die Lernmodule zu Lehreinheiten zu verbinden und zum anderen jeden Lernmodul in learning Objects und Medienobjekte herunterzubrechen. Ein möglicher Aufbau einer Vorlesung aus den zu entwickelnden Lernmodulen wurde schon im Kakitel 2 vorgestellt.

Eine Gliederung im Rahmen einer zweistündigen Vorlesung könnte wie folgt aussehen:

1. Einleitung

    Fehler und ihre Kosten

    Prozessqualität und Produktqualität

    Risikomanagement

    ISO-Normen

2. Prozessqualität - Prozeßmodelle

    Prozessmodelle Übersicht

    Das Capability Maturity Modell (CMM)

    Das V-Modell im Überblick

    Der Rational Uni£ed Process im V-Modell

    Prozessbeschreibung SSDA

    V-Modell -Anwendung

3. Produktqualität - Software-Tests

    Prüfung von SW-Komponenten - Einzeltests

    Prüfung von komponentenbasierten Systemen - Integrationstests

    Prüfungvon Simulationsprogrammen - Funktionstests

    Testumgebungen - Abnahmetests

# 5 Anforderungen an die Lehr-/Lernumgebung

Um die im didaktischen Konzept angesprochene zeitliche Dimension realisieren zu können, müssen wir auf Standardwerkzeuge sowohl zur Erstellung des Lehrmaterials als auch zum Zugriff darauf zurückgreifen. Im wesentlichen sind das Web Browser und MS Of£ce Tools. Ziele, die damit nicht erreicht werden können, müssen so isoliert realisiert werden, dass eine spätere Migration zu anderen Werkzeugen seiteneffektfrei möglich ist. In dem im nächsten Kapitel vorgestellten Werkzeug AIDA haben wir eine Umgebung, die diese Minimalanforderungen erfüllt. Wegen der geringen Anforderungen sollte alles Material auch leicht in die MuSofT-Plattform übertragbar sein.

# 6 Technische Realisierung

Vorgesehen ist der Einsatz des Werkzeugs AIDA (Active Information Development Assistant) des IAS der UNI Stuttgart, das die Erstellung frei de£nierbarer Informations-Präsentationen ausgehend von der Spezi£kation der Eingangsinformationsstruktur und der Spezi£kation der Ausgabeinformationsstruktur erlaubt.

AIDA wurde basierend auf der am IAS entwickelten Methode Aktive Informationspräsentation (AIP) entwickelt. Es wird im BMBF-Projekt ITO - Information Technology Online `http://www.ias.uni-stuttgart.de/de/start_forschung.html`) weiterenteickelt und eingesetzt. AIDA ermöglicht die Erstellung von Multimedia-Vorlesungen, indem aus verschiedenen multimedialen Einzelelementen eine integrierte Vorlesungspräsentation erstellt wird. Über Hypermedia-Elemente kann der Lehrende auf alle multimedialen Lehrmaterialien einfach und effektiv zugreifen, wobei er außer dem Web-Browser kein weiteres Werkzeug benötigt. Als Eingangsinformationen können u.a. mit Of£ce-Werkzeugen erstellte Dokumente, Gra£ken verschiedener Formate sowie Videos im AVI-Format verwendet werden. Bei der Erstellung der Informationsapplikation werden die Eingangsinformationen ins HTML-Format (Hypertext Markup Language) transformiert und miteinander verknüpft. Eine standardisierte Benutzungsober¤äche erlaubt den Zugang zu Einzelinformationen. Für die Realisierung von API-Applikationen wurden verschiedene Werkzeuge entwickelt, welche es ermöglichen, ohne HTML-Kenntnisse beliebig komplexe Applikationen aufzubauen. AIDA erlaubt die Erstellung frei de£nierbarer AIP-Applikationen ausgehend von der Spezi£kation der Eingangsinformationsstruktur und der Spezi£kation der Ausgabeinformation. Eine Erstellung einer Multimedia-Vorlesung mit dem Werkzeug AIDA kann wie folgt aussehen.

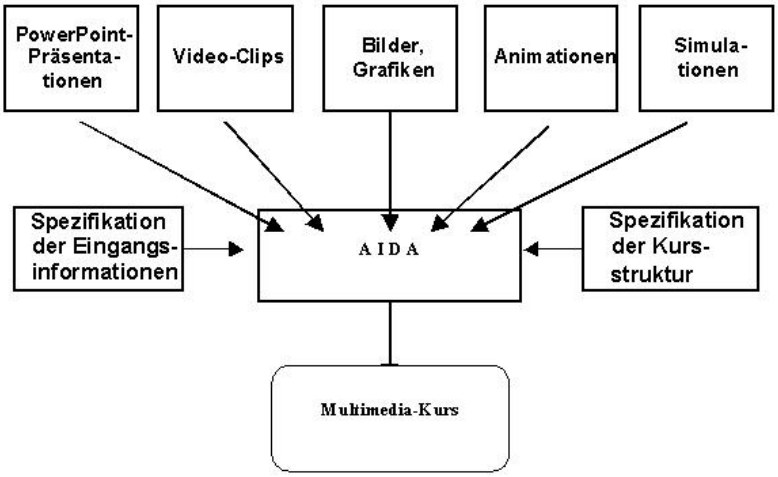



Abbildung 1: Erstellung eines Mulitmedia-Kurses mit Hilfe von AIDA

## 7 Evaluation

Im schon erwähnten Papier zu den didaktischen Grundlagen £ndet sich auch ein längerer Abschnitt zu Evaluation. Er zeigt vor allem, dass der Evaluationsprozess zeitaufwendig ist. Nach unserer Erfahrung sind die Inhalte der hier geplanten Lernmodule nicht statisch genug, um solch eine langwierige Evaluation als sinnvoll erscheinen zu lassen. Geplant ist deswegen

die Lernmodule in aktuellen Vorlesungen einzusetzen und vor allem darauf hin abzuprüfen, wie sie das Erreichen konkreter Lernziele unterstützen und wie sie entsprechend der aktuellen Situation einer konkreten Lehrveranstaltung dazu modi£ziert werden können.

# 8 Zusammenfassung

Die Lehreinheiten 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 Tayloring auf aktuelle Projekte angepasst werden können. Dadurch können die Lernmodule in den verschiedensten Kontexten wiederverwendet werden. Die Lernmodule werden ergänzt durch Beispielanwendungen, die mit den Systemen erzeugt wurden.

Am Ende des Jahres 2001 haben wir eine erste Fassung der Lerneinheit 7 realisiert und sind in der Phase der Erprobung sowohl in eigenen Vorlesungen als auch im Rahmen der Lehreinheit 3.3.

# 9 Literatur

**Didaktische Grundlagen**
Osmankovic, Alma: Möglichkeiten computerunterstützten Lernens im Bereich der beru¤ichen Weiterbildung - Untersuchung eines Lehrgangs des Technischen Hilfswerkes Diplomarbeit 2001 Die Arbeit ist unter `http://www.ike.uni-stuttgart.de/~www_wn/lehre` verfügbar.

**Software-Technik**
Balzert, Helmut: Lehrbuch der Software-Technik : Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung / Helmut Balzert. - Heidelberg ; Berlin : Spektrum, Akad. Verl., 1998

**Vorgehensmodelle**
Oestereich, Bernd : Erfolgreich mit Objektorientierung : Vorgehensmodelle und Managementpraktiken für objektorientierte Softwareentwicklung / Bernd Oestereich. Peter Hruska ... - 2.Au¤age. München, Wien : Oldenbourg, 2001

**V-Modell**
Versteegen, Gerhard (Hrsg.) : Das V-Modell in der Praxis: Grundlagen, Erfahrungen, Werkzeuge. Heidelberg: dpunkt-Verl. 2000

**Rational Uni£ed Process**
Kruchten, Philippe : Der Rational Uni£ed Process : Eine Einführung : Addison-Wesley 1999

Versteegen, Gerhard (Hrsg.) Projektmanagement mit dem Rational Uni£ed Process Versteegen, Gerhard (Hrsg.)Xpert.press Springer 2000

**Software-Tests**
Robbins, John: Debugging Applications, w.CD-ROM. The Bugslayer's guide to £nding and £xing coding errors in Microsoft Windows-based applications. Microsoft Press Corp., 2000

**ISO-Normen**
Thaller , Georg Erwin : ISO 9001 : Software-Entwicklung in der Praxis / Georg Erwin Thaller - 2. Au¤age - Hannover : Heise, 2000

**V-Modell im Uni£ed Process**
Reinhold, Markus : "Rational Uni£ed Process 2000" und "V-Modell, 97": Synergie oder Widerspruch? Aus OBJEKTspektrum 3/2000, S.74-81

**AIDA**
Göhner, Peter: Neue Medien in der Aus- und Weiterbildung. Folien zu Vortraegen.,
`http://www.ias.uni-stuttgart.de/vortraege/rv-zukunft/html/index.htm`

**Methode *Aktive Informationspräsentation*** (AIP)
`http://www.ias.uni-stuttgart.de/de/start_vortrag.html`

# MuSofT-Ergebnisbericht des Teilprojekts 3.3

Corina Kopka

## Inhaltsverzeichnis



## 1 Einleitung

Klassische Softwareentwicklungsprozesse, die z.B. dem Wasserfallmodell folgen, haben das Problem, dass die Risiken in die Zukunft verschoben werden und Fehler früher Phasen nur mit viel Aufwand rückgängig gemacht werden können. In der Industrie hat sich daher eine iterative Softwareentwicklung durchgesetzt. Für eine praxisnahe Ausbildung wurde der Uni£ed Process als iterativer Softwareentwicklungsprozess gewählt.

Die Lerneinheit 3.3 *Durchführung von Softwareprojekten mit dem Uni£ed Process* soll im Rahmen des Projekts *MuSofT - Multimedia in der SoftwareTechnik* entwickelt und für die Lehre der Softwaretechnik eingesetzt werden. Sie soll Unterstützung bei der Durchführung von Softwarepraktika bieten. 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

Uni£ed 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.

# 2 Inhalt

In diesem Abschnitt wird der Lerngegenstand der Lerneinheit 3.3, der Uni£ed Process, zusammenfassend geschildert.

## 2.1 Beschreibung

Im Uni£ed Software Process [JBR98] wird die Lebensdauer eines Softwaresystems zeitlich in mehrere Zyklen unterteilt. Ein Durchlauf des Uni£ed Software Process wird Zyklus (cycle) genannt. Am Ende des Zyklus wird ein neues Produkt-Release freigegeben. Jeder Zyklus ist zeitlich durch Phasen weiter unterteilt. Der Uni£ed Process unterscheidet die Phasen *Konzeptualisierung (inception)*, *Entwurf (elaboration)*, *Konstruktion (construction)*, *Übergang (transition)*. Zu jeder Phase gehört ein Meilenstein, nach dessen Erreichen über die Fortsetzung des Projekts entschieden wird. Phasen bestehen aus mehreren Iterationen.

Jeder Zyklus wird orthogonal zur zeitlichen Unterteilung in Phasen inhaltlich nach *Work¤ows* unterteilt. Im Uni£ed Process werden die folgenden fünf wichtigsten, aufeinanderfolgenden Work¤ows (*core work¤ows*) unterschieden: *Anforderungen*, *Analyse*, *Design*, *Implementierung* und *Test*. In jeder Phase wird diese Folge von Haupt-Work¤ows durchlaufen. Die Gewichtung und die Gestaltung eines Work¤ows sind von der jeweiligen Phase abhängig (vgl. Abb. 1).

In der Phase *Konzeptualisierung* ist der Work¤ow *Anforderungen* wesentlich, die Work¤ows *Design*, *Implementierung* und *Test* haben einen sehr geringen Anteil. Der bedeutendste Work¤ow der Phase *Konstruktion* ist der Work¤ow *Implementierung*, während die Work¤ows *Analyse* und *Anforderungen* nur eine Nebenrolle spielen. Der Anteil des Work¤ows *Test* ist am Ende der Phase *Konstruktion* und zu Beginn der Phase *Übergang* am höchsten und ist ansonsten nach der Phase *Konzeptualisierung* weitgehend konstant. Das Ergebnis eines Work¤ows ist die Erzeugung oder Erweiterung von Modellen. Im Uni£ed Process werden folgende Modelle unterschieden: Anwendungsfallmodell (*use case model*), Analysemodell (*analysis model*), Designmodell (*design model*), Verteilungsmodell (*deployment model*), (*implementation model*) und Testmodell (*test model*). Diese Modelle bestehen aus Teilmengen der UML. Jedes Modell wird in einem bestimmten Work¤ow bearbeitet. Das Anwendungsfallmodell wird im Work¤ow *Anforderungen* bearbeitet. Im Work¤ow *Analyse* wird daraufhin das Analysemodell bearbeitet. Im Work¤ow *Design* werden das Designmodell und das Verteilungsmodell bearbeitet. Schließlich wird im Work¤ow *Implementierung* das Implementierungsmodell bearbeitet und im Work¤ow *Test* das Testmodell.

## 2.2 Charakteristische Aspekte des Uni£ed Process

Charakteristisch für den Uni£ed Process sind das explizite Planen von Iterationen in den Phasen. Das Planen und der Durchlauf der Iterationen werden durch einen Iterations-Work¤ow

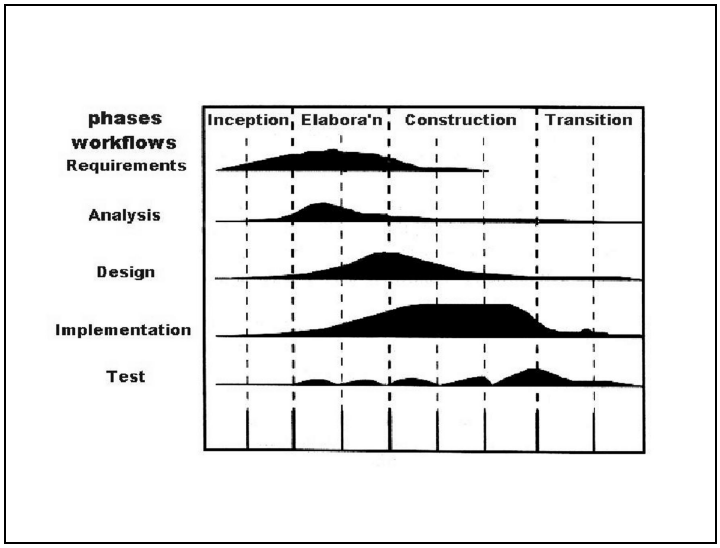



Abbildung 1: Phasen und Work¤ows im Uni£ed Software Process (nach [JBR98])

beschrieben. Dieser wird im folgenden vorgestellt.

**Der Iterations-Work¤ow** In einer Phase £nden im allgemeinen mehrere Durchläufe durch die Folge der Haupt-Work¤ows *Anforderungen, Analyse, Design, Implementierung* und *Test* statt. Ein solcher Durchlauf mit einer vorangestellten Planungsphase und einer abschließenden Beurteilungsphase heißt Iteration und ist eine Art Miniwasserfall.

Der Iterations-Work¤ow folgt einem generischen Muster, der die Basis für die konkreten Iterationen bildet. Der Iterations-Work¤ow beinhaltet die Planung jeder Iteration, den Durchlauf durch die fünf Haupt-Work¤ows und die Evaluation der Iteration nach vorher festgelegten Kriterien. Diese Evaluation dient auch als Grundlage für die mögliche Neuplanung der folgenden Iteration oder für die Änderung des Prozesses.

Da eine Iteration alle Haupt-Work¤ows, inbesondere Implementierung und Test, durchläuft, führt sie zu einer neuen Version des Systems und aktualisiert alle entsprechenden Modelle. Den Unterschied zwischen zwei Versionen nennt man Inkrement. Das bedeutet, dass die Iterationen die resultierenden Modelle inkrementweise erstellen. Jede Iteration ergänzt jedes Modell, wenn sie die Haupt-Work¤ows durchläuft. Z.B. wird an dem Anwendungsfallmodell in frühen Phasen mehr gearbeitet, während das Implementierungsmodell in der Phase *Konstruktion* mehr Beachtung £ndet.

Zwei Iterationen können sich zeitlich überlappen, solange der Test der früheren Iteration abgeschlossen ist, bevor das Design der späteren Iteration begonnen wird. Eine stärkere Überlappung ist nicht möglich, da sonst eventuell auf einer fehlerbehafteten Iteration aufgebaut wird (vgl. Abb. 2). Im wesentlichen bleibt die Folge der Iterationen daher seriell.

In Abb. 3 wird ein Wasserfallmodell und der iterative Uni£ed Process entlang der Zeitach-

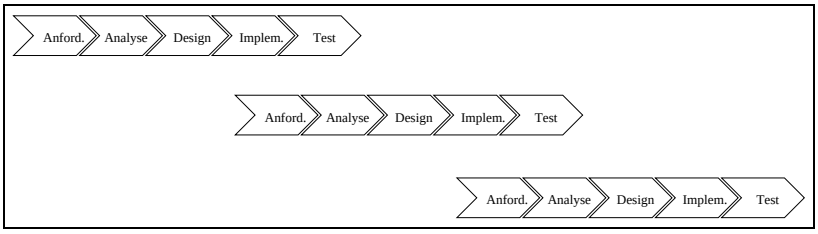



Abbildung 2: Überlappung der Iterationen (nach [JBR98])

se dargestellt. Der Uni£ed Process wird verstanden als eine Folge von Iterationen von der

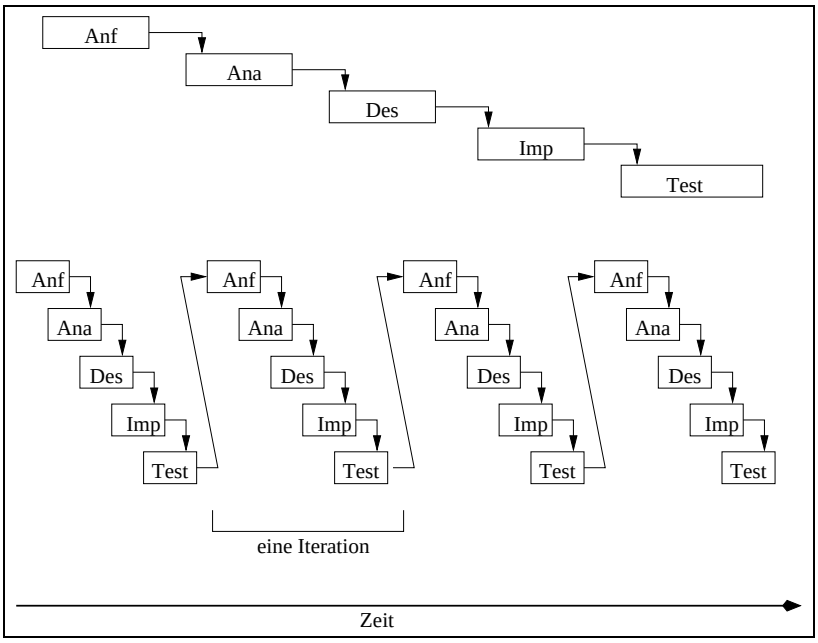



Abbildung 3: Vergleich mit dem Wasserfallmodell (nach [Kru99])

ursprünglichen Idee bis zum fertigen System, während im Wasserfallmodell keine Iterationen vorgesehen sind. Der Mehrwert des Uni£ed Process im Vergleich zu anderen iterativen Vorgehensmodellen ist die explizite Planung der Iterationen von Anfang an mit dem Ziel die verschiedenen erstellten Modelle (Anwendungsfallmodell, Analysemodell, Designmodell, Verteilungsmodell, Implementierungsmodell, Testmodell) zu ergänzen oder zu ändern.

# 3 Didaktisches Konzept

Die Lerneinheit 3.3 soll Lernmodule zur Unterstützung der Durchführung von Softwarepraktika im Bereich der Softwaretechnik bieten. Sie besteht aus drei Lernmodulen: *Uni£ed Process, Projekte maßschneidern (engl. Tailoring), Adaptiver Projekttutor*. Die Strukturierung der Lerneinheit in Lernmodule und die Beschreibung der Lernmodule wird in Abschnitt 4 vorgenommen.

## 3.1 Leitbild

Im Fachbereich Informatik werden im Bereich der Softwaretechnik Praktika durchgeführt, die vermehrt im Grundstudium oder auch im Hauptstudium für verschiedene Studiengänge angeboten werden. Es betrifft die Studiengänge der Kerninformatik und Angewandten Informatik, die natur- und ingenieurwissenschaftlichen Studiengänge mit Nebenfach Informatik und die sogenannten Bindestrich-Informatiken. Die Lerneinheit 3.3 *Durchführung von Softwareprojekten mit dem Uni£ed Process* soll im Rahmen eines Softwarepraktikums in der Hochschule vorzugsweise im Grundstudium eingesetzt werden. Sie kann ebenso in den berufsbildenden IT-Ausbildungen eingesetzt werden, die Softwareentwicklung lehren. Institutionen wie Universitäten, Fachhochschulen, IT-Zentren oder Weiterbildungszentren der Unternehmen können die LE 3.3 einsetzen. Gründliche Kenntnisse des objektorientierten Paradigmas und damit einhergehender Vorgehensweisen in der Entwicklung sind notwendig.

## 3.2 Lernziele

Die Lernziele sind für die Lernmodule unterschiedlich.

**Uni£ed Process** Hier sind die Ziele im kognitiven Bereich des Faktenwissens und des Wissens um komplexe Zusammenhänge anzusiedeln. Da davon ausgegangen wird, dass die Lerner selbständiges Lernen gewöhnt sind, ist ein offeneres (kein straffes lineares) Vorgehen im Lernmodul erforderlich. Das selbstgesteuerte hypermediale Explorieren soll ermöglicht werden, um differenziert auf verschiedene Lernertypen eingehen zu können.

*Das konkrete Lernziel:* Nach der Bearbeitung dieses Lernmoduls kennt der Lernende die zentralen Begriffe eines Entwicklungsprozesses (Tätigkeiten, Dokumente, Rollen, Methoden) und deren Zusammenhang und hat eine grobe Vorstellung über einen möglichen Projektablauf basierend auf dem Uni£ed Process. Der Projektablauf wird dadurch expliziert und das Beschäftigen des Lerners damit schafft ein Bewusstsein für den Entwicklungsprozess.

**Projekte maßschneidern** Hier sind die Ziele im kognitiven Bereich von Fähigkeiten anzusiedeln. Ausgehend von der Problemstellung, einen konkreten Projektverlauf für das aktuelle Anwendungsprojekt zu bestimmen, wird zusätzliche Information und Hilfestellung (Fallbeispiele: Vorstellung von Projektabläufen für Musterprojekte und deren Besonderheiten und Beschreibung der Ableitung aus generischen projekttypspezi£schen Projektabläufen) gegeben. Fähigkeiten wie interpretieren, ordnen, zuordnen, ableiten, analytisch und synthetisch anwenden, bauen auf vorhandenem Vorwissen (z.B. Lernmodul *Uni£ed Process*, Fallbeispiele in diesem Lernmodul) auf.

*Das konkrete Lernziel:* Nach der Bearbeitung dieses Lernmoduls kann der Lernende bezogen auf ein konkretes Projekt einen Projektablauf basierend auf dem Uni£ed Process vorschlagen oder bestimmen. Das Re¤ektieren über alternative konkrete Abläufe ist ein gutes Training für die spätere Anwendung dieses Wissens, beim Befolgen des geplanten Projektablaufs.

**Adaptiver Projekttutor** In diesem Lernmodul sind die Ziele im Bereich kognitiver und sozial-kommunikativer Fähigkeiten zu sehen. Wesentlich bei den kognitiven Fähigkeiten ist etwas tun können weil es verstanden wird. Dies bezieht sich auf die Anwendung eines konkreten Entwicklungsprozesses im aktuellen Projekt durch Unterstützung eines virtuellen Projekttutors. Sozial-kommunikative Fähigkeiten werden durch das Arbeiten in einem Team durch die Visualisierung der Abhängigkeit der Arbeitsschritte und des Arbeitsfortschritts gefördert.

*Das konkrete Lernziel:* Durch die Benutzung dieses Lernmoduls kann der Lerner einen konkreten Entwicklungsprozess innerhalb eines Teams befolgen. Er hat durch Anwendung eines geplanten Projektablaufs beispielhaft gelernt, welches die Schnittstellen zu den anderen Teammitgliedern sind, welche Abhängigkeiten zwischen Arbeitsschritten bestehen und wie sich Verzögerungen bei der Durchführung dieser Arbeitsschritte auf das gesamte Projekt auswirken.

## 3.3 Lernszenario

**Uni£ed Process** Der Ablauf eines Lernvorganges in diesem Lernmodul wird vom Lerner im Selbststudium durch hypermediales Explorieren gesteuert. Das Lernmodul informiert schwerpunktmäßig und der Lerner konsumiert. Es ist ein multimediales interaktives Nachschlagewerk über den möglichen Projektablauf eines Musterprojektes. Dem Lerner wird über die visuelle gra£sche Darstellung des Projektablaufs (z.B. als Petri-Netz) ein gra£sches Navigieren zu den einzelnen Tätigkeiten (Aktivitäten) ermöglicht, die dann in einzelnen Szenen näher beschrieben werden. In jeder Szene sollen zu jeder Aktivität eine Kurzbeschreibung, die vorliegenden und bereitzustellenden Dokumente (Artefakte), dafür benötigte Templates, die zugehörigen Methoden und die ausführende Rolle (Worker) angegeben werden. Die Schnittstellen zu anderen Aktivitäten und der zugehörigen Rollen sollen ebenso visualisiert werden. Ein weiterer Zugang für das Navigieren des Lerners wird über ein zusätzliches Anbieten von Funktionen (z.B. Verzeichnis für Aktivitäten, Rollen, Methoden, Artefakte; Glossar; Literaturverzeichnis) gewährleistet.

Der Lehrende kann dieses Lernmodul als mediales Element der Präsenzveranstaltung (Vorlesung im Praktikum) zum einen so benutzen wie der Lerner (hypermedial erkundend) und einen Lehrweg beschreiten, den er sich zum Lehren festgelegt hat (Lehrersicht). Hiermit zeigt er einen Ausschnitt des Lernmoduls in der Vorlesung. Zum anderen ist vorgesehen, geführte Touren (Guided Tours) für den Lehrenden schon in der Entwicklung vorzubereiten, die sequentiell eine Folge von Szenen starr verknüpfen. Eventuell müssen dafür Szenen anders didaktisch aufbereitet werden als in der Lernersicht.

Das Lernmodul wird vorzugsweise in der Präsenzlehre eingesetzt, die Lernersicht des Lernmoduls bietet zudem die Möglichkeit des Selbststudiums ohne Betreuung am Ausbildungsort oder zu Hause.

Nach der Bearbeitung dieses Lernmoduls kennt der Lernende die zentralen Begriffe eines

Entwicklungsprozesses (Tätigkeiten, Dokumente, Rollen, Methoden) und deren Zusammenhang und hat eine grobe Vorstellung über einen möglichen Projektablauf basierend auf dem Uni£ed Process. Der Projektablauf wird dadurch expliziert und das Beschäftigen des Lerners damit schafft ein Bewusstsein für den Entwicklungsprozess.

**Projekte maßschneidern** Der Ablauf eines Lernvorganges in diesem Lernmodul beginnt idealerweise mit dem Studium von Fallbeispielen für konkrete Projektabläufe. Ein mögliches Ableitungsszenario wäre, ausgehend von einem projekttypspezi£schen generischen Ablauf das Ableiten projektspezi£scher Abläufe und deren Rahmenbedingungen und Besonderheiten zu beschreiben. Innerhalb dieser Rahmenbedingungen gehören auch die Kennzeichnung von Aktivitäten und Dokumenten, die auf jeden Fall Teil eines konkreten Projektablaufs sein müssen. Dazu gehören auch Dokumente, die im konkreten Projekt an die lehrende Person abzugeben sind. In einem zweiten Schritt sollen die Lerner (evtl. mit Betreuung) einen Projektablauf für ihr eigenes Projekt herleiten. Dabei werden sie von einem Werkzeug unterstützt. Dieses Lernmodul kann als betreute Übung im Praktikum statt£nden.

Die Planung des Projektablaufs soll optional auch um weitere Aufgaben eines Projektmanagements erweitert werden können, wie z.B. Ressourcenzuordnung im Entwicklungsteam: Die Zuordnung von Teammitgliedern zu Rollen oder/und Aktivitäten könnte dann das Tailoring ergänzen.

Dieses Lernmodul wird nicht direkt durch die lehrende Person eingesetzt, sondern sie betreut die Lerner bei der Nutzung in der Präsenzlehre (Übung). Die Lerner können es zum Teil auch im Selbststudium in Selbstlernphasen nutzen.

Nach der Bearbeitung dieses Lernmoduls kann der Lernende bezogen auf ein konkretes Projekt ein Projektablauf basierend auf dem Uni£ed Process vorschlagen oder bestimmen. Das Re¤ektieren über alternative konkrete Abläufe ist ein gutes Training für die spätere Anwendung dieses Wissens, beim Befolgen des geplanten Projektablaufs.

**Adaptiver Projekttutor** Dieses Lernmodul ist ein adaptives projektbegleitendes Hilfesystem. Ein virtueller Projekttutor kann in einem Anwendungsprojekt den Arbeitsfortschritt einer Arbeitsgruppe abschätzen und Hilfestellung für den weiteren Projektverlauf anbieten. Dafür wird eine Übersicht über die erledigten und verbleibenden Arbeitsschritte für die Arbeitsgruppe gegeben. Für die verbleibenden Arbeitsschritte können Mitglieder der Arbeitsgruppe an die Präsentation des Musterprojekts anknüpfen (Lernmodul *Uni£ed Process*), um eine geeignete Anleitung für die weitere Arbeit zu bekommen.

Dieses Lernmodul wird im Praktikum durch die Arbeitsgruppen eingesetzt. Eine Betreuung durch einen menschlichen Tutor ist hier nicht vorgesehen. Durch die Benutzung dieses Lernmoduls kann der Lerner einen konkreten Entwicklungsprozess innerhalb eines Teams befolgen. Er hat durch Anwendung eines geplanten Projektablaufs beispielhaft gelernt, welches die Schnittstellen zu den anderen Teammitgliedern sind, welche Abhängigkeiten zwischen Arbeitsschritten bestehen und wie sich Verzögerungen bei der Durchführung dieser Arbeitsschritte auf das gesamte Projekt auswirken.

### 3.4 Didaktischer Ansatz

Der didaktische Ansatz im Lernmodul *Uni£ed Process* erlaubt entdeckendes Lernen. Es werden Fakten und Zusammenhänge im Projektablauf eines Musterprojekts präsentiert, die aber durch selbständiges Navigieren entdeckt werden. Zusätzliche Anleitungen durch Verzeichnisse, Glossar, usw. unterstützen den Benutzer im Beschaffen von Informationen. Die Verzeichnisse für Aktivitäten, Rollen, Methoden, Artefakte und das gra£sche Navigieren durch den Projektablauf vernetzen die Inhalte nach Zusammenhängen.

Das didaktische Konzept im Lernmodul *Projekte maßschneidern* ist eine Kombination von fallbasiert, produktorientiert (das Produkt ist der konkrete Projektablauf) und dekonstruktiv analytisch. Die Ableitung eines konkreten Projektablaufs baut auf das Studium der Fallbeispiele auf.

Der didaktische Ansatz ist im Lernmodul *Adaptiver Projekttutor* hier prozessorientiert. Um den Entwicklungsprozess zu verinnerlichen, wird der Prozess Schritt für Schritt erfahren.

## 4 Strukturkonzept

Die Lerneinheit 3.3 soll Lernmodule zur Unterstützung der Durchführung von Softwarepraktika im Bereich der Softwaretechnik bieten. Im Softwarepraktikum entwickeln die Studierenden in Arbeitsgruppen aufgrund einer konkreten Aufgabenstellung in verschiedenen Projekten Software. Dabei wird ein Entwicklungsprozess durchlaufen, der mit Hilfe dieser Lerneinheit zu erlernen ist und expliziert wird. Die multimediale Visualisierung des Ablaufs eines Projektes basierend auf dem Uni£ed Process soll dem Lernenden projektbegleitend mehr Unterstützung geben. Eines der Lernmodule kann auch für die Präsenzlehre im Hörsaal durch Dozenten eingesetzt werden, um eine Einführung in den Uni£ed Process innerhalb einer Vorlesung zu geben.

Die Lerneinheit besteht aus drei Lernmodulen:

**Lernmodul *Uni£ed Process*** Dieses Lernmodul präsentiert den Projektablauf eines Musterprojekts. Der Lerner kann im Selbststudium selbstgesteuert den Ablauf eines Musterprojekts betrachten. Die Darstellung des Projektablaufs soll Tätigkeiten, Dokumente und Rollen berücksichtigen. Dieses Lernmodul kann wahlweise auch von einem Dozenten zum Lehren innerhalb einer Vorlesung eingesetzt werden. Für den Einsatz als Lehrmodul ist vorgesehen, eine lineare feste Folge von Szenen (in Form einer *Guided Tour*) aufzubereiten oder einen anderen Zugriff auf Szenen des Lernmoduls zu ermöglichen. Dieses Lernmodul unterscheidet jeweils eine Sicht für den Lerner und den Lehrer.

**Lernmodul *Projekte maßschneidern (engl. Tailoring)*** Dieses Lernmodul stellt ein Werkzeug zur Verfügung, womit die zentralen Elemente eines Entwicklungsprozesses (Aktivitäten, Dokumente, Rollen) auf die Erfordernisse des Anwendungsprojekts und auf die jeweilige Anwendungsdomäne anpaßbar sind. Mit dem Werkzeug soll ein konkreter Prozess für das Anwendungsprojekt modelliert werden. Eventuell sollen die Anzahl der Iterationen in den Phasen des Uni£ed Process eingeschränkt werden, da die Projekte eines Praktikums in der Ausbildung im Vergleich zur industriellen Entwicklung sehr kurzlebig sind. Beispiele für konkrete Projektabläufe aus Musterprojekten sollen mit ihren Besonderheiten zur Hilfestellung vorgestellt werden.

**Lernmodul *Adaptiver Projekttutor*** In diesem Lernmodul soll ein virtueller Projekttutor auf Basis eines konkreten Projektablaufs den Arbeitsfortschritt einer studentischen Arbeitsgruppe anhand der erstellten Dokumente in einem Projekt erkennen. In Abhängigkeit des erzielten Arbeitsfortschritts wird Hilfestellung für den weiteren Projektverlauf gegeben. Dies kann durch eine Übersicht über die verbleibenden Arbeitsschritte und durch das Präsentieren von Lernmodulen mit genauen Beschreibungen, Anleitungen und Beispielen zu diesen Schritten geschehen. Der konkrete Projektablauf ist entweder ein vorgegebener Standardprozess oder einer der von der Arbeitsgruppe erarbeitet wurde.

# 5 Lehr/Lernumgebung

Die Lernmodule können unabhängig voneinander eingesetzt werden. Es wäre aber sinnvoll, sie im Verbund einzusetzen. Die zu entwickelnden Lernmodule unterstützen unterschiedliche Lernphasen. Während das Lernmodul *Uni£ed Process* eine Einführung bietet, werden die Lernmodule *Projekte maßschneidern* und *Adaptiver Projekttutor* zur Vertiefung und Anwendung des bisherigen Wissens eingesetzt. Es existieren somit Querbezüge zwischen den Lernmodulen. Zum Beispiel möchte der Lernende beim Einsatz der Lernmodule *Projekte maßschneidern* und *Adaptiver Projekttutor* auf den konkreten Projektablauf eines Musterprojekts im Lernmodul *Uni£ed Process* zugreifen. Diese Art von Querbezügen sollte technisch unterstützt werden.

Optional könnte die Kommunikation in einem Projektteam und zwischen Team und Tutor zusätzlich unterstützt werden. Hierzu könnte die Übungsplattform der “Virtuellen Welt“ (Projekt Dortmund/Paderborn) eingesetzt werden, wenn sie sich bzgl. eines Einsatzes in einem Praktikum als geeignet erweist.

# 6 Technische Realisierung

**Formate und Werkzeuge** Bei der technischen Umsetzung der Lerneinheit 3.3 ist aufgrund der Vereinbarungen im MuSofT-Projekt Java vorgesehen. Für die Gestaltung des Lernmoduls *Uni£ed Process* als hypermediales System kommen z.B. HTML u.ä. Formate in Frage. Des weiteren ist für das Tailoring ein Werkzeug bereitzustellen, das für Prozessmodellierung geeignet ist und auch zur Visualisierung des Uni£ed Process in den anderen Lernmodulen verwendet werden kann. Hierfür ist zwischen den Alternativen Eigenentwicklung und Einsatz bereits existierender Werkzeuge (eventuell auch deren Weiterentwicklung) sorgfältig auszuwählen. Für die Entscheidungs£ndung werden geeignete existierende Prozessmodellierungswerkzeuge und Frameworks für gra£sche Editoren in die engere Wahl genommen und getestet.

**Vorgehensweise in der Entwicklung** Das Teilprojekt 3.3 vollzieht sich in den groben Phasen Vorplanung, Konzeptionierung, Realisierung und Test. In die Vorplanung gehören inhaltliche und didaktische Vorüberlegungen, die Festlegung eines Vorgehensmodells und die Projektplanung. Die Konzeptionierungsphase beinhaltet im wesentlichen die Erstellung eines Grobkonzepts, eines Feinkonzepts und eines Drehbuchs. Letzteres bildet die Vorlage für die

Realisierungsphase, in der die softwaretechnische Entwicklung der Lerneinheit und die Produktion von Medienobjekten statt£nden. Beim Testen wird zwischen technischem Test und Evaluation als inhaltlich-didaktischer Test der Lerneinheit unterschieden. Eine Probesequenz (Szenenfolge) soll beispielhaft entwickelt und getestet werden, 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) sollen mehrfach iterativ durchlaufen werden (Abb. 4). Eine Iteration wird für eine Sequenz von Szenen (bei dokumentbasierten Lernmodulen, wie z.B die multimediale Präsentation im Lernmodul *Uni£ed Process*) oder für eine Menge von Anwendungsfällen (bei softwarebasierten Lermodulen, wie z.B. *Tailoring*) durchgeführt.

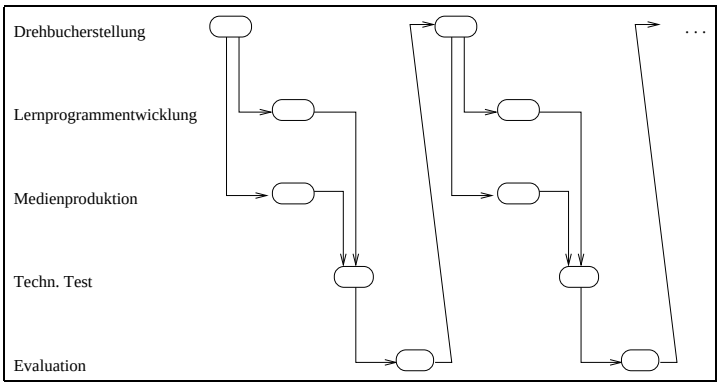



Abbildung 4: Iterationen in der Entwicklung

# 7 Evaluierung

Für den Einsatz und die Evaluation der Lerneinheit sind verschiedene Veranstaltungen der Universität Dortmund geeignet.

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 zu evaluieren, ist im Rahmen dieses Vorhabens eine Zusammenarbeit mit den Veranstaltern des Softwarepraktikums des Fachbereichs Informatik der Universität Dortmund angestrebt, die zur Zeit ein grobes Prozessmodell [Sch01] einsetzen, das an das ISP-Modell (*Irrational Separated Model*) von Hitz und Kappel [HK99] angelehnt ist. Der Prozess selbst ist aber nicht das Lernobjekt. Durch den Einsatz der Lerneinheit 3.3 des MuSofT-Projekts soll in Softwarepraktika auch der Entwicklungsprozess zum Lerngegenstand werden und beispielhaft anhand des Uni£ed Process gelehrt werden können.

Für die Evaluation wird im Softwarepraktikum ein Projekt nach dem bisherigen groben Prozessmodell durchgeführt. Ein zweites Projekt wird mit Unterstützung der Lerneinheit 3.3

nach dem Uni£ed Process durchgeführt. Es wird ein Fragebogen entwickelt, der nach Durchführung der beiden Softwareprojekte an die Teilnehmer des Praktikums verteilt wird.

Neben der Evaluation im Softwarepraktikum ist das Lernmodul *Uni£ed Process* auch geeignet, in der Vorlesung "Softwaretechnik" eingesetzt zu werden. Beim Einsatz als Lehrmodul steht die Evaluation durch den vortragenden Dozenten im Vordergrund. Beim Einsatz aller Lernmodule im Softwarepraktikum spielt neben der Evaluation durch den Lehrenden auch die Evaluation durch Lernende eine große Rolle.

# 8 Zusammenfassung

Die Lerneinheit 3.3 ist ein Lehr-und Lernsystem, das über die Gestaltung als Uni£ed-Process-Informationssystem hinaus geht. Die Einbettung der Lerneinheit in ein didaktisches Konzept führt zu Anforderungen wie die didaktischen Aufbereitung von Lehrmaterial, der Einsatz mediendidaktischer Elemente und die Betreuung der Lernenden im Lernprozess:

- Dozenten sollen anhand von Lehrmodulen den Uni£ed Process lehren können. Die Einbeziehung eines didaktischen Konzepts in der Entwicklung der Lerneinheit 3.3 soll die Präsentation reinen Faktenwissens in einem Informationssystem ergänzen.

- Lerner sollen unter Betreuung in einem konkreten Projekt den Entwicklungsprozess auf Basis des Uni£ed Process planen lernen. Lernen bedeutet an dieser Stelle Anwenden des Faktenwissens über den Uni£ed Process, insbesondere auch das Planen von Iterationen des Iterations-Work¤ows in einem konkreten Projekt.

- Lerner sollen den Uni£ed Process explizit erfahren. Lernen bedeutet an dieser Stelle das Befolgen eines vorgegebenen oder des selbst geplanten Entwicklungsprozesses innerhalb des Softwarepraktikums. Hiermit werden z.B. geplante Prozesse evaluiert. Diese Evaluation ist ebenfalls ein Lernprozess mit dem Ziel, Projekte besser planen zu können.

Dies unterscheidet die Lerneinheit 3.3 als multimediales Lehr- und Lernsystem von einem reinen Informationssystem, wie z.B. das Online-System für den Rational Uni£ed Process [RUP]. Gewiss sind die visuelle gra£sche Darstellung eines Projektablaufs im Softwarepraktikum und ein gra£sches Navigieren durch diese Darstellung geeignete Mittel, um ein multimediales Nachschlagewerk zu konzipieren. Dies ist auch für die Lerneinheit 3.3 vorgesehen. In Abgrenzung zum RUP-Hilfesystem sollen die Zusammenhänge zwischen Rollen, Aktivitäten und Dokumenten zur besseren Übersicht auch gra£sch besser verdeutlicht werden, wie etwa die Schnittstellen zwischen Rollen oder die Eintritts- und Austrittskriterien für Aktivitäten (in Form von Ein-, Ausgabedokumenten).

Darüber hinaus soll eine stärkere Betonung auf die Konzeption der Lerneinheit als Lehr- und Lernsystem gelegt werden. Dies bedeutet z.B. eine stärkere Betonung auf den Übungsanteil (z.B. in einer bestimmten Rolle die Schnittstellen zu anderen Rollen erkunden oder Projektabläufe selbst planen). Des weiteren soll der Projektfortschritt erfasst werden und davon abhängig das weitere Vorgehen in Form von Übersichten über den restlichen Projektablauf und passende inhaltliche Informationen/Lernmodule präsentiert werden.

## Literatur


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

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

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

[RUP] *Rational Uni£ed Process*. `www.rational.com/rup_info/`.

[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 des MuSofT-Teilprojekts 3.4 über das erste Projektjahr

Udo Kelter

## Inhaltsverzeichnis



## 1 Einleitung

Dieses Papier beschreibt die Lehrinhalte, die durch das Teilprojekt 3.4 im Detail abgedeckt werden sollen, und begründet die Auswahl. Weiter werden die Materialien, die im ersten Pro-

jektjahr entwickelt worden sind, und die beim Einsatz der Materialien gewonnenen Erfahrungen dargestellt.

# 2 Thema

Die geplanten Lehrmaterialien decken mehrere wesentliche Aspekte des Managements der Software-Entwicklung ab. Da Software meist in Projekten entwickelt wird, bezeichnen wir diesen Bereich als Projektmanagement.

Die im MuSofT-Teilprojekt 3.4 erstellten Materialien können nur einen Teil des Themengebiets Projektmanagement abdecken; die Auswahl £el auf folgende Teilgebiete:

1. Versions- und Kon£gurationsmanagement
2. Projektplanung
3. Fehler- und Problemmanagement

## 2.1 Kriterien für die Themenauswahl

Für diese Auswahl gab es mehrere Arten von Gründen, die in den folgenden Abschnitten diskutiert werden: (a) Merkmale des Themenbereichs Projektmanagement, (b) thematische (Nicht-) Überlappung mit anderen MuSofT-Teilprojekten und (c) didaktische Aspekte.

### 2.1.1 Merkmale des Themenbereichs Projektmanagement

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 Systemanalyse und Programmentwurf hat der Themenbereich Projektmanagement in der grundständigen Lehre 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.

[1] Bei anderen softwaretechnischen Themengebieten, z.B. Modellierungssprachen, sind die Querbezüge und Zusammenhänge viel ausgeprägter, so daß einzelne Teilbereiche nicht ohne weiteres ausgetauscht oder weggelassen werden können.

- 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 ergeben 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 "häu£g" gelehrt werden, häu£g in dem Sinne, daß sie in entsprechendem Umfang in Vorlesungen an vielen Standorten gelehrt bzw. in vielen Softwaretechnik-Lehrbüchern vertreten sind.

- Die Materialien sollen in möglichst kleinen Teilmengen isoliert einsetzbar sein.

- 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 bzw. nur am Rande behandelt werden:

- Vorgehensmodelle und Prozeßmodellierung

- ISO 9000

### 2.1.3 Didaktische 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 beru¤ichen 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 £nden 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 werden in den Materialien von Teilprojekt 3.4 nur am Rande berücksichtigt; im Vergleich zu anderen Lehrbüchern sind sie zugunsten anderer Themen unterrepräsentiert.

## 2.2 Geplante Themengebiete

Die vorstehend diskutierten Kriterien für die Themenauswahl haben zu folgender Liste von Themengebieten geführt, die durch die in Teilprojekt 3.4 zu erstellenden Materialien zumindest einführend, punktuell auch vertiefend abgedeckt werden sollen:

1. Versions- und Kon£gurationsmanagement (VKM)
2. Projektplanung
3. Fehler- und Problemmanagement

### 2.2.1 Versions- und Kon£gurationsmanagement

Einfache Kon£gurationsmanagementprobleme 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 Kon£gurationsmanagement 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.
- Das Thema VKM ist verbunden mit mehreren praxisrelevanten benachbarten Bereichen (Werkzeuge, Problemmanagement, Qualitätssicherung, Build-Systeme, ...)

- 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:

**Grundstudium / Programmierpraktikum:** Im Programmierpraktikum treten die folgenden praktischen Probleme auf, bei denen der Einsatz von VKM motiviert ist und geübt werden kann:

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

- zeitliche Revisionen von Dokumenten (Kon£gurationen 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 hier Materialien das System CVS ausgewählt.

**Hauptstudium:** Mit VKM-Problemen ist hier insb. in einer Projektgruppe und bei der Bachelor- bzw. Diplomarbeit zu rechnen. Die im Grundstudium erworbenen (und hier vorausgesetzten) Kenntnisse sind in folgenden Punkten auszubauen:

- Bzgl. der verteilten Gruppenarbeit ergeben sich bei Projektgruppen nicht generell neue Anforderungen. Allerdings müßten die Teilnehmer an einer Projektgruppe selbständig in der Lage sein, ein Repository aufzubauen, zu kon£gurieren und ggf. über verteilte Clients zugreifbar zu machen (beim Programmierpraktikum ist dies von den Betreuern zu übernehmen).

- Der Umfang der erstellten Systeme kann hier signi£kant größer als im Programmierpraktikum sein, so daß hier die automatische Systemgenerierung (make o.ä.) wichtig wird und mit Kon£gurationen und Varianten zu rechnen ist.

Im Einzelfall können noch weitgehendere Anforderungen je nach dem Thema der Projektgruppe oder der Diplomarbeit auftreten, die aber zu speziell sind.

Vermittelt werden können die notwendigen theoretischen und praktischen Kenntnisse in Softwaretechnik-Vorlesungen, aber auch in Kompaktkursen innerhalb einer Projektgruppe oder in Spezialvorlesungen. Die zu erstellenden Materialien sollen möglichst modular gestaltet sein, um in unterschiedlichen Kontexten benutzt werden zu können.

**Gewählte Einzelthemen:** Aufgrund der vorstehenden Überlegungen wurden folgende Einzelthemen aus dem Themengebiet Versions- und Kon£gurationsmanagement ausgewählt: Motivation für Versions- und Kon£gurationsmanagement, paralleles Arbeiten in Entwicklergruppen, lange Transaktionen und Work Spaces, Revisionen, das Mischproblem, das Concurrent Versions System (CVS), Aufbau, Benutzung und Administration eines Repositories, Übersicht über VM/CM-Systeme und Integration mit build-Systemen.

Nach dem derzeitigen Planungsstand sollen diese Themen durch folgenden Lehrmodule abgedeckt werden:

- Modul KM1: Einführung in das Kon£gurationsmanagement
- Modul CVS1: Einführung in CVS
- Modul KM2: Differenzen zwischen Dokumenten
- Modul CVS2: Weitere CVS-Funktionen
- Modul KM3: Kon£gurationen und Varianten
- Modul KM4: Automatisierte Dokumentgenerierung
- Modul KM5: KM und Projektmanagement

### 2.2.2 Projektplanung und -Verfolgung

Zentral für das Themengebiet Projektplanung und -Verfolgung sind Methoden der Aufwandsschätzung und die Netzplantechnik. Diese Themen werden auch relativ häu£g in Lehrbüchern behandelt, da sie nicht sehr kompliziert sind und in der beru¤ichen Praxis durchaus 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 praktischen Arbeiten z.B. im Rahmen des Programmierpraktikums sind zu wenig umfangreich, um z.B. Netzpläne sinnvoll einsetzen zu können. Für die gängigen Aufwandsschätzungsmethoden fehlen Vergleichsdaten aus früheren Projekten. Am ehesten können diese Techniken im Hauptstudium, insb. bei Projektgruppen, Studienarbeiten und Diplomarbeiten eingesetzt werden.

**Gewählte Einzelthemen:** Folgende Einzelthemen aus dem Themengebiet Projektplanung und -Verfolgung wurden ausgewählt: Methoden der Aufwandsschätzung, Netzplantechnik, hier insb. elementare und geschachtelte Netzpläne, Netzplanauswertung, Kapazitätsplanung und -Ausgleich, Aufwandserfassung und Kostenzuordnung und inkrementelle Neuplanung von Projekten.

Nach dem derzeitigen Planungsstand sollen diese Themen durch folgenden Lehrmodule abgedeckt werden:

- Modul PPL: Projektplanung – Grundbegriffe
- Modul AWS1: Aufwandsschätzung – Grundbegriffe

- Modul AWS2: Die Function-Point-Methode
- Modul NPT1: Netzplantechnik
- Modul NPT2: Einsatzplanung und Kapazitätsausgleich

### 2.2.3 Fehler- und Problemmanagement

Das Themengebiet Versions- und Kon£gurationsmanagement behandelt Methoden und Werkzeuge zum Verwalten von Versionen, wobei aber offen bleibt, warum überhaupt neue Versionen, Varianten und Releases erstellt worden sind und welche Mängel wo behoben worden sind. Letzteres ist das Kernthema des Fehler- und Problemmanagements. Das Fehler- und Problemmanagement ist somit die naheliegendste Ergänzung des Versions- und Kon£gurationsmanagements.

Das Themengebiet Fehler- und Problemmanagement ist in bisherigen Lehrbüchern weniger stark repräsentiert, zumindest explizit. Dies liegt u.a. daran, daß die meisten Lehrbücher bisher stark auf die Erstentwicklung von Software fokussiert sind, während die Weiterentwicklung nur am Rande betrachtet wird[2]. Dieser Zustand ändert sich aber derzeit insofern, als im Zusammenhang mit objektorientierten Methoden zunehmend inkrementelle Entwicklungsprozesse eingesetzt und gelehrt werden (mit dem Extremfall XP). Hierdurch rückt auch das Fehler- und Problemmanagement stärker in den Blick.

Da objektorientierte Entwicklungsprozesse und insb. der RUP bereits in einem anderen Teilprojekt behandelt werden, konzentriert sich das Teilprojekt 3.4 auf die Werkzeugunterstützung beim Fehler- und Problemmanagement. Ein wesentliches Lernziel ist dabei wiederum, ein konkretes Fehler- und Problemmanagementsystem benutzen und kon£gurieren zu können. An der Auswahl dieses Systems wird derzeit gearbeitet.

**Gewählte Einzelthemen:** Nach dem derzeitigen Planungsstand sollen folgende Einzelthemen aus dem Themengebiet Versions- und Kon£gurationsmanagement abgedeckt werden: Erfassung von Fehler- und Problemmeldungen, Analyse und Klassi£kation, Arbeitsplanung und Entscheidungsunterstützung, Reporting, Statistiken und Metriken, Integration mit Versions- und Kon£gurationsmanagement sowie Testwerkzeugen,

# 3 Didaktisches Konzept

## 3.1 Generelles Konzept

Die Materialien sollen sowohl die Präsenzlehre als das Selbststudium abdecken. Die unterschiedlichen Medien sollen dabei wie folgt eingesetzt werden:

Selbststudium:

[2] Immerhin wird meist erwähnt, daß die Weiterentwicklung meist wesentlich aufwendiger als die Erstentwicklung ist.

1. Volltext in druckbarer Form, ggf. ergänzt um eine HTML-Variante[3]

2. in Java geschriebene Animationen, die entweder als Applet in HTML-Text eingebettet werden oder als Applikation selbständig laufen. Bei den Animationen handelt es sich um zweidimensionale Graphiken, die konzeptuelle Strukturen (z.B. Versionsgraphen, Netzpläne) veranschaulichen, Veränderungen darin visualisieren und erläutern. Die Animationen können auch interaktive Elemente enthalten, z.B. zur Eingabe bestimmter Kommandos oder Werte, mit denen gerechnet werden soll.

3. Videoaufzeichnungen von Sitzungen an einem Rechner, in denen die Bedienung bestimmter Werkzeuge und Systeme erklärt wird.

Präsenzlehre:

1. Stichworte und Skizzen zur Vortragsunterstützung (Folien oder HTML)

2. Animationen (s.o.)

3. Videoaufzeichnungen (s.o.)

Die "Masse" des Stoffs wird in sowohl beim Selbststudium als auch bei der Präsenzlehre in Texten und Bildern (Graphiken), also in "konventionellen" Medien, angeboten; "neue" Medien sollen nur punktuell dort eingesetzt werden, wo ein tatsächlicher Mehrnutzen gegenüber konventionellen Medien erreicht werden kann.

Übungen zu dem Lehrmaterial werden angeboten werden, sind aber kein Gegenstand multimedialen Materials. In den meisten Fällen bestehen die einzig sinnvollen Übungen darin, die für das Thema einschlägigen Werkzeuge konkret einzusetzen. Zusätzliche rechnergestützte Übungen in neuen Medien (z.B. Simulationen der Werkzeuge) erscheinen in diesen Fällen nicht sinnvoll.

Wegen der Heterogenität der Teilgebiete ist der Medieneinsatz für jedes Teilgebiet individuell zu betrachten.

Auf die konventionellen Medien (Volltext und Vortragsfolien) gehen wir i.f. nicht ein.

Im Bereich Fehler- und Problemmanagement wurden noch keine Materialien erstellt.

## 3.2 Versions- und Kon£gurationsmanagement

Mit erster Priorität wird hier ein ca. 30 Minuten langes Video realisiert, das die grundlegenden Arbeitsabläufe bei der Benutzung von CVS anhand von Beispielen zeigt. Gezeigt wird primär ein Bildschirm mit den Mausbewegungen darauf sowie den jeweiligen Ausgaben des Systems; dies alles wird durch einen Sprecher zusätzlich erläutert. In späteren Versionen des Videos soll zusätzlich der Effekt der CVS-Operationen durch Veränderungen in Graphen, die

[3] Die Volltexte waren schon weitgehend vor dem Projekt MuSofT vorhanden und sind im WWW frei verfügbar. Obwohl sie nicht im Rahmen des Projekts erstellt wurden, stellen sie eine zentrale Komponente des Medienspektrums dar.

Versionsbäume darstellen, visualisiert werden. Als Vorbereitung zu dem Video ist ein Lehrtext vorgegeben, in dem die grundlegenden Begriffe von CVS erklärt werden.

Das Video ist in mehrere Abschnitte von ca. 1.5 - 6 Minuten Länge gegliedert, zwischen denen jeweils eine Diskussionspause eingelegt werden kann.

Die Erstversion des Videos wurde im Wintersemester 2001/02 testweise im Programmierpraktikum bei 8 Gruppen eingesetzt. Ca. 1 Woche später wurde die Teilnehmer bzgl. des Nutzens des Videos befragt. Wesentliche Beobachtungen waren:

- Das Video wurde weit überwiegend als nützlich und hilfreich angesehen. In der Tat verlief die Benutzung des zentralen CVS-Repositories in dem Praktikum problemlos, obwohl es sich um eine für die Studenten völlig neue Technologie handelt.

- Die Diskussionspausen wurden intensiv genutzt.

- Rund die Hälfte der Studenten hatte - entgegen der Vorgabe - den Lehrtext (Modul CVS1) nicht vorab gelesen. Auch diese Gruppe beurteilte das Video positiv, hat aber vermutlich den Inhalt weniger präzise verstanden, worauf auch bestimmte Umfrageergebnisse hindeuten. Das mit dem Video verfolge Ziel, die Benutzung eines KMS zu motivieren und "Berührungsängste" abzubauen scheint auch bei einem reduzierten Verständnis der fachlichen Inhalte erreicht zu werden.

- Es wurden diverse Möglichkeiten erkannt, das Video im Detail didaktisch noch zu verbessern. Diese Verbesserungen sollen in eine für Anfang 2002 geplante neue Version ein¤ießen.

## 3.3 Projektplanung und -Verfolgung

Der Einsatz neuer Medien erscheint in diesem Stoffgebiet vor allem bei der Visualisierung von Netzplanberechnungen (Vorwärts- / Rückwärtsrechnung) sinnvoll. Anzuzeigen sind die in einem konkreten Beispiel jeweils auszuführenden Rechenschritte.

Besonderer Wert wurde darauf gelegt, daß nicht nur ein fest vorgegebener Netzplan berechnet werden kann, sondern z.B. während eines Vortrags vom Vortragenden ad hoc neue Beispiele erstellt und vorhandene modi£ziert werden können.

Es wurde ein erster Prototyp eines Editors und Simulators für Netzpläne als Java-Applikation erstellt. Der Prototyp weist folgende Eigenschaften auf:

- Es können vorbereitete Beispiel-Netzpläne, die in einem XML-Format gespeichert sind, geladen werden, aber auch völlig neue Netzpläne erstellt bzw. vorhandene modi£ziert werden. Die Bedienelemente zum Editieren der Netzpläne wurden bewußt einfach gestaltet.

- Der Algorithmus zur Vorwärts- / Rückwärtsrechnung des Netzplans kann in mehreren Varianten simuliert werden (feingranulare / gröbere Rechenschritte, explizit auszulösende Einzelschritte / automatischer Ablauf in verschiedenen Geschwindigkeitsstufen). Der gerade erfolgte Rechenschritt wird jeweils durch einen kurzen Text erläutert.

Der Prototyp wurde im Wintersemester 2001/02 erstmals in der Vorlesung Softwaretechnik I und den zugehörigen Übungsgruppen eingesetzt. Der Einsatz in der Vorlesung dauerte ca. 10 Minuten; hiervon konnten nur ca. 2 Minuten auf Erklärungen zu den Anzeigen und der Bedienung des Systems verwendet werden. Die Einsatzerfahrungen können wie folgt zusammengefaßt werden:

- Das System kann sich im Editier- oder im Simulationsmodus be£nden; zwischen beiden Zuständen kann rasch umgeschaltet werden. Das Erkennen des aktuellen Modus scheint manchen Zuschauern Probleme zu bereiten. Die Darstellung des aktuellen Modus muß daher noch expliziter werden, ebenso ggf. die Darstellung des Umschaltvorgangs.

- Die Kommandos zur Simulationssteuerung sind bisher nur graphisch gekennzeichnet. Die graphische Darstellung wird anscheinend nicht schnell genug verstanden und muß dahingehend verbessert. Die vorhandenen Tool Tips nützen nur bedingt, weil sie nicht immer sichtbar sind und ggf. bei einer Projektion nicht gut lesbar sind.

Daneben wurden noch eine Reihe kleinerer Details erkannt, in denen das System optimiert werden muß.

## 4 Strukturelles Konzept

Wie schon in Abschnitt 3.1 erwähnt werden unterschiedliche konventionelle und neue Medien im Verbund eingesetzt, die in geeigneter Kombination sowohl die Präsenzlehre als auch das Selbststudium unterstützen.

In den Fällen, wo ein Lernziel darin besteht, Algorithmen zu verstehen, die auf Netzplänen, Dateibäumen oder ähnlichen Datenmengen arbeiten, sollen die Algorithmen visualisiert werden, indem einzelne Fallbeispiele durch Animationen dargestellt werden. Die Ausgestaltung dieser Animationen und ggf. zusätzlichen Editierfunktionen hängt sehr von Einzelfall ab und kann hier nicht pauschal diskutiert werden.

In den Fällen, wo die Benutzung und Bedienung von Werkzeugen erlernt werden soll, werden die wichtigsten Arbeitsabläufe (“*use cases*”) durch Ver£lmungen erklärt.

## 5 Anforderungen an die Lehr-/Lernumgebung

I.w. werden Anzeige- bzw. Abspielwerkzeuge für die eingesetzten Medienformate benötigt. Für die Java-Simulationen wird ein JDK 1.2 benötigt.

Für die Distribution der Materialien mit Ausnahme des Videos reicht ein einfacher Dokumentenserver (FTP oder HTTP) aus. Ein entsprechender WWW-Server (`pi.informatik.uni-siegen.de`) be£ndet sich bereits seit längerem in Betrieb und wird intensiv genutzt.

# 6 Technische Realisierung

Die technische Realisierung der einzelnen Medienobjekte und Programme wird in separaten Dokumenten beschrieben werden.

# 7 Evaluation

Das oben erwähnte Video wurde mittels einer Umfrage anhand eines Fragebogens evaluiert.

Die Art, wie die weiteren Materialien evaluiert werden, liegt zur Zeit noch nicht fest.

# 8 Zusammenfassung

Das Teilprojekt 3.4 behandelt aus dem Themenkomplex Projektmanagement die Themenbereiche

1. Versions- und Kon£gurationsmanagement
2. Projektplanung
3. Fehler- und Problemmanagement

Im ersten Projektjahr wurden diverse Vorarbeiten geleistet und Erstversionen von zwei neuen Objekten erstellt, einem Video, das in die Bedienung eines graphischen Front-Ends für das Kon£gurationsmanagementsystem CVS einführt, und einem Java-Programm, durch das Netzpläne editiert werden können und die Vorwärts-/Rückwärtsrechnung darin simuliert werden kann.

/99/ T. Bühren, M. Cakir, E. Can, A. Dombrowski, G. Geist, V. Gruhn, M. Gürgrn, S. Handschumacher, M. Heller, C. Lüer, D. Peters, G. Vollmer, U. Wellen, J. von Werne
Endbericht der Projektgruppe eCCo (PG 315)
Electronic Commerce in der Versicherungsbranche
Beispielhafte Unterstützung verteilter Geschäftsprozesse
Februar 1999

/100/ A. Fronk, J. Pleumann,
Der DoDL-Compiler
August 1999

/101/ K. Alfert, E.-E. Doberkat, C. Kopka
Towards Constructing a Flexible Multimedia Environment for Teaching the History of Art
September 1999

/102/ E.-E. Doberkat
An Note on a Categorial Semantics for ER-Models
November 1999

/103/ Christoph Begall, Matthias Dorka, Adil Kassabi, Wilhelm Leibel, Sebastian Linz, Sascha Lüdecke, Andreas Schröder, Jens Schröder, Sebastian Schütte, Thomas Sparenberg, Christian Stücke, Martin Uebing, Klaus Alfert, Alexander Fronk, Ernst-Erich Doberkat
Abschlußbericht der Projektgruppe PG-HEU (326)
Oktober 1999

/104/ Corina Kopka
Ein Vorgehensmodell für die Entwicklung multimedialer Lernsysteme
März 2000

/105/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Zwischenbericht der Projektgruppe IPSI
April 2000

/106/ Ernst-Erich Doberkat
Die Hofzwerge — Ein kurzes Tutorium zur objektorientierten Modellierung
September 2000

/107/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
November 2000

/108/ Stefan Austen, Wahid Bashirazad, Matthais Book, Traugott Dittmann, Bernhard Flechtker, Hassan Ghane, Stefan Göbel, Chris Haase, Christian Leifkes, Martin Mocker, Stefan Puls, Carsten Seidel, Volker Gruhn, Lothar Schöpe, Ursula Wellen
Endbericht der Projektgruppe IPSI
Februar 2001

/109/ Leonid Abelev, Carsten Brockmann, Pedro Calado, Michael Damatow, Michael Heinrichs, Oliver Kowalke, Daniel Link, Holger Lümkemann, Thorsten Niedzwetzki, Martin Otten, Michael Rittinghaus, Gerrit Rothmaier
Volker Gruhn, Ursula Wellen
Zwischenbericht der Projektgruppe Palermo
Februar 2001

/110/ Eugenio G. Omodeo, Ernst-Erich Doberkat
Algebraic semantics of ER-models from the standpoint of map calculus.
Part I: Static view
März 2001

/111/ Ernst-Erich Doberkat
An Architecture for a System of Mobile Agents
März 2001

/112/ Corina Kopka, Ursula Wellen
Development of a Software Production Process Model for Multimedia CAL Systems by Applying Process Landscaping
April 2001

/113/ Ernst-Erich Doberkat
The Converse of a Probabilistic Relation
June 2001

/114/ Ernst-Erich Doberkat, Eugenio G. Omodeo
Algebraic semantics of ER-models in the context of the calculus of relations.
Part II: Dynamic view
Juli 2001

/115/ Volker Gruhn, Lothar Schöpe (Eds.)
Unterstützung von verteilten Softwareentwicklungsprozessen durch integrierte Planungs-, Workflow- und Groupware-Ansätze
September 2001

/116/ Ernst-Erich Doberkat
The Demonic Product of Probabilistic Relations
September 2001

/117/ Klaus Alfert, Alexander Fronk, Frank Engelen
Experiences in 3-Dimensional Visualization of Java Class Relations
September 2001

/118/ Ernst-Erich Doberkat
The Hierarchical Refinement of Probabilistic Relations
November 2001

/119/ 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 Intermediate Report
November 2001

/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