Prof. Dr. Johannes Siedersleben, Diplom-Mathematiker, Professor für Informatik an der FH Rosenheim und wissenschaftlicher Leiter von sd&m Research; 1979 bis 1983 wiss. Assistent an der TU Karlsruhe, Schwerpunkte: Lineare Optimierung auf Diagraphen, Komplexitätstheorie, Branch-and-Bound; 1984 bis 1995 Designer, Projektleiter, Chefberater, Bereichsleiter und Geschäftsführer; maßgeblicher Einfluss auf die sd&m-Softwaretechnik, vor allem im Bereich der Spezifikation.

Frau Prof. Dr. Debora Weber-Wulff ist seit 2001 Professorin für Medieninformatik an der Fachhochschule für Technik und Wirtschaft in Berlin und forscht im BMBF-Leitprojekt »Virtuelle Fachhochschule«, wo sie mehrere Arbeitspakete leitet. Sie hat an der Universität Kiel in der Theoretischen Informatik promoviert und in der Industrie bei »Norsk Data« in der Softwareentwicklung gearbeitet. Von 1993-2001 war sie Professorin für Softwaretechnik an der TFH Berlin.

Die SEUH-Organisatoren bedanken sich bei den Sponsoren:

BMW

iteratec

Projective Software GmbH

ÜBER JEDEM STANDARD

s d &m software design & management

Johannes Siedersleben, Debora Weber-Wulff (Hrsg.)

Software Engineering im Unterricht der Hochschulen

SEUH 8 – Berlin 2003

dpunkt.verlag

Prof. Dr. Johannes Siedersleben
johannes.siedersleben@sdm-research.de

Prof. Dr. Debora Weber-Wulff
weberwu@fhtw-berlin.de

Herstellung: Birgit Bäuerlein
Umschlaggestaltung: Helmut Kraus, Düsseldorf
Druck und Bindung: Koninklijke Wöhrmann B.V., Zutphen, Niederlande

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

ISBN 3-89864-201-1
1. Auflage 2003
Copyright © 2003 dpunkt.verlag GmbH
Ringstraße 19 b
69115 Heidelberg

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten.
Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Alle Informationen in diesem Buch wurden mit größter Sorgfalt kontrolliert.
Weder Autoren noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

In diesem Buch werden eingetragene Warenzeichen, Handelsnamen und Gebrauchsnamen verwendet. Auch wenn diese nicht als solche gekennzeichnet sind, gelten die entsprechenden Schutzbestimmungen.

Vorwort

Software-Engineering-Unterricht an den Hochschulen – ist das nicht so etwas wie Manöver im Sandkasten? Wie kann man das Vorgehen in realen Projekten im Unterricht theoretisch und – vor allem – praktisch vermitteln? Die immer noch enormen Schwierigkeiten großer Software-Projekte haben ihre Ursache wenigstens zum Teil in der unzureichenden Vorbereitung der Hochschulabsolventen.

Wohl alle einschlägigen Studiengänge enthalten inzwischen Studentenprojekte – aber haben die irgendetwas mit der Realität zu tun? Oft laufen sie noch chaotischer ab als in der Wirklichkeit. Viele Fragen sind noch nicht oder nur unzureichend beantwortet: Welche Vorgaben erhalten die Studenten? In welcher Entwicklungsumgebung arbeiten sie? Wie lässt sich verhindern, dass sie nicht in technischen Details ersticken? Wie überprüft man das Vorgehen im Projekt und letztlich den Projekterfolg? Sind junge Assistenten, die selber noch kein Projekt gemacht haben, als Betreuer überhaupt geeignet?

Die SEUH-Workshops bringen Experten an einen Tisch, um diese und andere Fragen zu diskutieren und einer Lösung näher zu bringen. Der vorliegende Band enthält die Beiträge zum 8. Workshop SEUH 2003, der diesmal in Berlin stattfindet. Aus den eingereichten Beiträgen hat das Programmkomitee elf Beiträge ausgewählt und in drei Gruppen eingeteilt:

- *Neue Konzepte im Software-Engineering-Unterricht*
- *Sinn und Unsinn von E-Learning*
- *SE-Unterricht in der Zusammenarbeit mit der Industrie*

Als Gastredner konnten wir Herrn Dr. Ulrich Weinmann, den Geschäftsführer der BMW Car IT, gewinnen. Er berichtet über Software im Automobil und die enormen Anforderungen an Software-Engineering, die sich daraus ergeben.

Zur Tradition der SEUH-Workshops gehört der Austausch von Meinungen und Erfahrungen in ganz besonderem Maß. Deshalb lässt der Zeitplan viel Puffer für die Diskussion der Beiträge. In jeder Sitzung gibt es einen Diskutanten, dessen Aufgabe es ist, mit manchmal provokativen Fragen das Gespräch anzuregen (was bisher immer gut geglückt ist). Zwei Sitzungen sind reine Diskussionsveranstaltungen: Am Donnerstag befassen wir uns mit der Rolle von Werkzeugen im Unterricht und der Frage nach der sinnvollen Halbwertszeit der im SE-Unterricht gelehrten Themen: ein oder zehn Jahre? Am Freitag behandeln wir das grundsätzliche Thema des Stands der SE-Ausbildung heute. Ohne der Diskussion vorzugreifen behaupten wir: Es gibt noch viel zu tun!

Ebenfalls Tradition der SEUH ist die Rückmeldung der Studenten: Sie haben in einer eigenen Sitzung die Gelegenheit, die vorgetragenen Beiträge aus ihrer Sicht zu beurteilen.

Der achte Workshop SEUH wird gemeinsam von den GI-Fachgruppen 2.1.1 (Softwaretechnik) und dem German Chapter der ACM veranstaltet. Gastgeberin ist die Fachhochschule für Technik und Wirtschaft Berlin (FHTW).

Die Mitglieder des ständigen Organisationskomitees sind Jochen Ludewig (Universität Stuttgart), Günter Riedewald (Universität Rostock) und Andreas Spillner (Hochschule Bremen).

Im Programmkomitee haben mitgearbeitet: Johannes Siedersleben (Vorsitzender, sd&m AG), Debora Weber-Wulff (Tagungsleiterin, FHTW Berlin), Uwe Dumslaff (sd&m AG), Peter Forbrig (Uni Rostock), Martin Glinz (Uni Zürich), Ulrike Jaeger (FH Heilbronn), Horst Lichter (RWTH Aachen), Jörg Raasch (HAW Hamburg), Andreas Spillner (Hochschule Bremen).

Wir bedanken uns bei allen, die bei der Organisation und bei der Durchführung der SEUH 2003 mitgewirkt haben. Besonderer Dank gilt den Sponsoren, ohne die die SEUH 2003 nicht stattfinden könnte. Dies sind: BMW AG, iteratec GmbH (München), projective GmbH (München), sd&m AG.

München und Berlin, im Januar 2003

Johannes Siedersleben *Debora Weber-Wulff*

Inhalt

Eingeladener Vortrag

Neue Konzepte im SE-Unterricht

E-Learning

Zusammenarbeit mit der Industrie

Software im Automobil – Anforderungen und Chancen

Dr. Ulrich Weinmann
BMW Car IT GmbH, München, Ulrich Weinmann@BMW-CarIT.de

Zusammenfassung

In diesem Beitrag werden die Arbeitsinhalte und die Positionierung der BMW Car IT im Rahmen der fahrzeugbezogenen Softwareentwicklung der BMW Group vorgestellt. Ausgangslage ist die steigende Relevanz von Software im Fahrzeug sowie der anstehende Paradigmenwechsel hin zu „Software als Produkt", bei dem Software als eigenständige Entwicklungsdomäne und als abgrenzbarer Wertschöpfungsbereich begriffen wird.

Die BMW Car IT widmet sich dabei primär der seriennahen Entwicklung einer Software-basierten Infrastruktur im Fahrzeug und bearbeitet dazu die Themenfelder „Software Transfer", „Human Vehicle Interaktion" sowie „Integriertes Datenmanagement" aus Sicht der Informatik. Abschließend werden kurz Herausforderungen und Chancen der Zulieferer und deren Rolle bei der Entwicklung Software-basierter Fahrzeugfunktionen skizziert.

Relevanz und Status von Software

Software im Fahrzeug ist keine Vision – in Premium-Fahrzeugen der neuesten Generation ist bereits ein erheblicher Softwareanteil vorzufinden.

Abb. 1: Elektronik und Software im neuen 7er BMW

Im aktuellen 7er BMW (vgl. Abbildung 1) sind Softwareumfänge von über 10 MB auf bis zu 70 Steuergeräte installiert. Typischerweise wird Software für spezifische Steuergeräte entwickelt und zusammen mit diesen ausgeliefert.

Die Ursache für den Einsatz von Software und deren Engineering-Methoden liegen u.a. in der zunehmenden Vernetzung von immer mehr Fahrzeugfunktionen und der resultierenden Gesamtsystemkomplexität: Bei heute über 700 Fahrzeugfunktionen im aktuellen 7er BMW wären ca. 250.000 direkte Wechselwirkungen zwischen diesen Funktionen möglich. Das Problem wird durch die enorme Varianten-, Konfigurations- und Versionsvielfalt der IT im Fahrzeug verschärft.

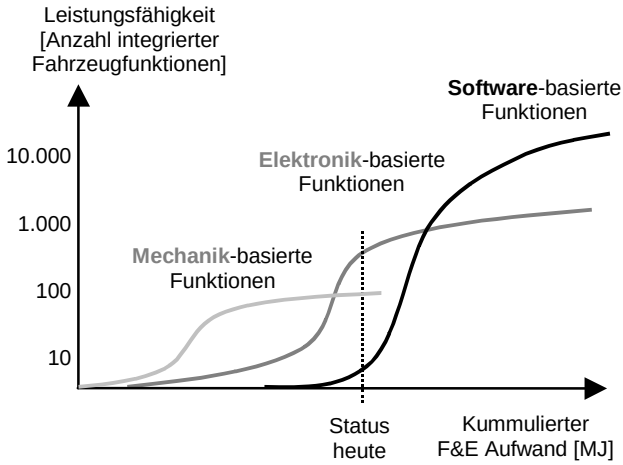


Abb. 2: Potenzial von Softwaretechnologien

Abbildung 2 zeigt das Potenzial von Softwarekonzepten und -technologien zur Beherrschung integrierter Fahrzeugfunktionen über investierte F&E-Aufwände.

Der Umfang und Wertschöpfungsanteil der Software im Fahrzeug wird auch weiterhin deutlich steigen, da fast alle Fahrzeugfunktionen von einem Software-basierten Gestaltungsansatz profitieren können. Software-basierte Infrastrukturkonzepte und Fahrzeugfunktionen werden maßgebliche Innovationstreiber im Fahrzeug sein. Dabei wird die Software besonders im Zusammenspiel mit Elektronik und Mechanik ihr Potenzial entfalten.

Software als eigenständige Domäne

Neben der quantitativen Zunahme von Software im Fahrzeug ist auch ein Paradigmenwechsel bei der Betrachtung der Softwareanteile zu erwarten. Die Software wird zunehmend unabhängiger von einer spezifischen Hardware, Sensorik oder Aktuatorik. Software wird als eigenständiges Produkt konzipiert, entwickelt und ausgeliefert.

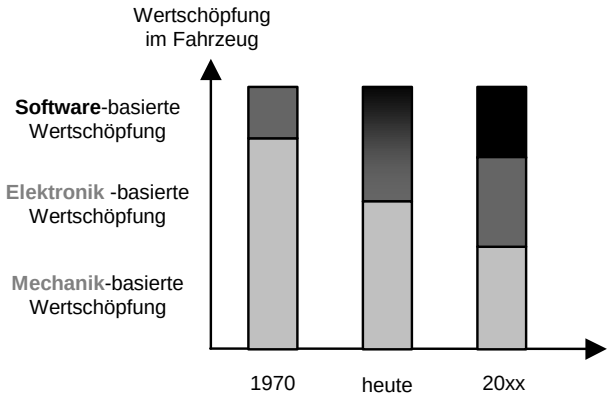


Abb. 3: Wertschöpfung durch Software

Wertschöpfung durch Software

Abbildung 3 skizziert den anwachsenden Wertschöpfungsanteil durch Software, wobei heute oft keine explizite Unterscheidung zwischen Elektronik- und Softwareanteilen vorgenommen wird.

Software stellt erhebliche Herausforderungen für Fahrzeughersteller und Zulieferer in fast allen Leistungsstufen dar, eröffnet aber auch neue Chancen. Insbesondere in der Fahrzeugentwicklung müssen bewährte Konzepte der Informatik auf ihren Einsatz im Fahrzeug hin überprüft und ggf. adaptiert werden.

Für Konzepte, die erfolgreich prototypisch implementiert worden sind, ist die Serientauglichkeit sowie Produktions- und Wartungsfähigkeit der Software durch geeignete Prozesse, Methoden und Werkzeuge sicherzustellen.

Anforderungen an Software im Fahrzeug

Das Software Engineering und die Softwarearchitektur müssen das spezifische Anforderungsprofil von Softwaresystemen im Fahrzeug berücksichtigen.

Abbildung 4 zeigt die Anforderungen, denen sich ein Fahrzeughersteller beim Produkt Automobil gegenübersieht. Hieraus lassen sich auch die Anforderungen an das Onboard Software System ableiten.

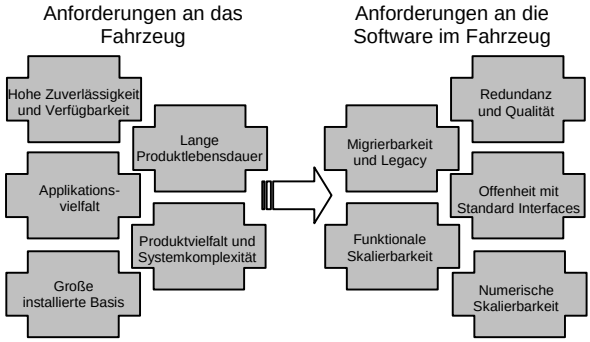


Abb. 4: Anforderungen an Fahrzeuge und Software

Dabei kann sich die Automobilindustrie an Software-Konzepten der Luftfahrtindustrie orientieren, um Sicherheits- und Echtzeitanforderungen sowie die Systemkomplexität zu managen. Andererseits kann bei Systemkosten, Änderungsfrequenzen und Support für eine große installierte Basis eine Orientierung an der Softwareindustrie erfolgen.

Ziele und Aufgaben der BMW Car IT

Die BMW Car IT GmbH wurde Ende 2001 als 100%-ige Tochter der BMW Group gegründet. Zielsetzung ist Stärkung der Kompetenzen zur Softwareentwicklung von fahrzeugbezogenen Systemen.

Der Schwerpunkt der Arbeit liegt auf der seriennahen Entwicklung einer Onbard Software Infrastruktur und assoziierter Entwicklungsmethoden mit einer prototypischen Umsetzung in den 3 Bereichen

- Software Transfer
- Human Vehicle Interaction
- Integriertes Datenmanagement

Die Relevanz für diese Bereiche wird deutlich, wenn man die Konsequenzen des Paradigmenwechsel hin zu „Software als Produkt“ analysiert. Fahrzeuge können künftig neue und verbesserte Fahrzeugfunktionen nachladen.

Dadurch muss eine sichere, kontrollierte Öffnung gewährleistet sein. Der Bereich **Software Transfer** adressiert das Problem der Autorisierung und Authentifizierung beim Zugriff auf das Onboard Software System sowie die Wahrung der Systemintegrität bei Änderungen durch Software Upgrades und Updates aber auch bei Hardware-Änderungen.

Handelt es sich bei einer neuen oder verbesserten Fahrzeugfunktion nicht um eine Service- oder Systemfunktion, sondern um eine Funktion mit einer Schnittstelle zum Kunden, so muss ggf. das MMI angepasst werden. Der Bereich **Human Vehicle Interaktion** untersucht Konzepte zur flexiblen Repräsentation von Fahrzeugfunktionen sowie der Handhabung der Informationstiefe.

Das Thema **Integriertes Datenmanagement** beschäftigt sich mit der Entwicklung einer Informationszentrale zur Bereitstellung von Verwaltungs- und Betriebsdaten im Fahrzeug sowie Mechanismen für deren Interpretation. Künftige Fahrzeugfunktionen benötigen keine Details über Steuergeräte oder Kommunikationsprotokolle, um gewünschte Daten zur weiteren Verarbeitung anzufordern oder bereitzustellen.

Die Themenbereiche lassen sich wie folgt einordnen:

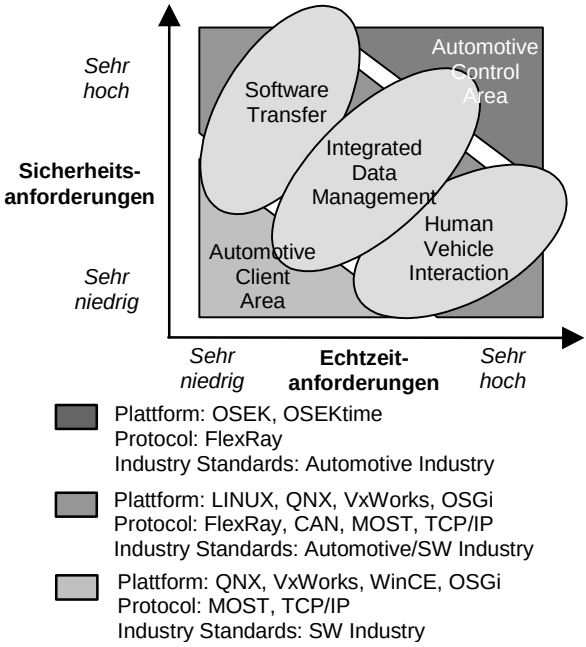


Abb. 5 Einordnung der Themenbereiche

Einordnung der Themenbereiche

In Abbildung 5 ist eine Taxonomie für verschiedenartige Fahrzeugfunktionen skizziert, die nach dem Ausmaß der notwendigen Sicherheits- und Echtzeitanforderungen unterscheidet: Die “Automotive Control Area” umfasst primäre Überwachungs- und Steuerfunktionen für die Fahrt und Fahrsicherheit, wie z.B. X-by-Wire. Die

“Automotive Client Area” beinhaltet Funktionen, die keinen spezifischen Mehrwert durch die Nutzung im Fahrzeug erzeugen, z.B. TV, Börsen News oder Spiele.

Dazwischen sind jene Fahrzeugfunktionen angesiedelt, die entweder nur sehr hohe Sicherheitsanforderungen (z.B. Software-Transfer-Funktionen) oder nur sehr hohe Echtzeitanforderungen (z.B. viele HVI-Funktionen) aufweisen oder den Anforderungen der unterstützen Funktionen unterliegen (z.B. Funktionen zum integrierten Datenmanagement). Gerade in diesen Bereichen werden kurzfristig auch erhebliche Synergien zwischen Automobil- und Software-Industrie erwartet, wie z.B. in der OSGi Vehicle Expert Group oder der Java Micro Edition für Embedded Software Profile.

Die BMW Car IT konzentriert sich auf die Auswahl und Adaption von Informatik-Konzepten aus den genannten Bereichen und deren prototypische Implementierung in Versuchsträgern und -aufbauten. Dabei werden etablierte Softwareentwicklungsmethoden genutzt und ggf. auch erweitert. Wichtig ist die Fokussierung auf die Onboard Software und deren Gestaltung: Was muss im Fahrzeug an Software-Infrastruktur vorhanden sein? Offboard Software wie z.B. die externe Software-Infrastruktur bei Telematik-basierten Fahrzeugfunktionen oder die Entwicklungs- und Testinfrastruktur im Rahmen der Entwicklung einer Fahrzeugfunktion spielen für die BMW Car IT eine untergeordnete Rolle.

Positionierung der BMW Car IT

Die Seriennähe der BMW Car IT wird durch die projektbezogene Einbindung in die BMW Group gewährleistet. Auswahl, Anforderungen und Seriennähe der Softwareprojekte wird durch eine formale Beauftragung der BMW Car IT durch die BMW Fachstellen sichergestellt. Dies unterstützt ferner den Ergebnistransfer, der beispielweise als Funktionsprototyp einer Spezifikation vorliegt oder als abgesicherter Beitrag in ein Lastenheft eingehen kann.

Im Gegensatz zur produktorientierten Serien-Softwareentwicklung bei der BMW Group, realisiert die BMW Car IT einen leichtgewichtigen, projektorientierten Softwareentwicklungsprozess für das Rapid Prototyping von Software-basierten System- und Fahrzeugfunktionen.

Die Innovationsfähigkeit der BMW Car IT wird auch durch die enge Zusammenarbeit mit Forschungs- und Bildungseinrichtungen für Informatik und Software Engineering unterstützt. Hier ist besonders die Zusammenarbeit mit dem Lehrstuhl für Software & Systems Engineering an der Technischen Universität München zu nennen, wie z.B. im Rahmen des Forschungsprojektes „Modellbasierte Entwicklung adaptiver Dienste“.

Rolle und Chancen von Lieferanten

Konsequenzen hat der Paradigmenwechsel hin zu „Software als Produkt“ auch auf die Lieferanten. Fahrzeugfunktionen werden zunehmend als reine Software-Implementierungen realisiert. Die Abstraktion von einer spezifischen Hardware und Elektronik durch General Purpose Computing Plattformen wird die Software austauschbar machen.

Voraussetzung hierfür sind offene Plattformen und Standards, um softwarebasierte Drittanwendungen zu ermöglichen. Hier bieten sich auch Chancen für Softwarefirmen, die bislang nicht als Automobilzulieferer aufgetreten sind, aber mit eigenen Software-Komponenten erfolgreich etabliert sind. Erfahrungen für den Einsatz im Embedded Umfeld sind dabei hilfreich.

Für die Gestaltung einer Onboard Software Infrastruktur kommen Frameworks und Softwarekomponenten aus verschiedenen Bereichen infrage. Das Integrierte Datenmanagement kann Softwarekomponenten z.B. für die Datenhaltung und Dateninterpretation nutzen. Der Software Transfer kann z.B. Verschlüsselungs-, Gateway- oder Firewall-Komponenten verwenden. Für die Human Vehicle Interaktion sind z.B. Komponenten für effiziente Grafik- oder Sprachverarbeitung notwendig.

Die traditionellen Elektronik-Lieferanten von Automobil- und Steuerelektronik stehen vor vergleichbaren Chancen und Herausforderungen wie die Automobilhersteller. Rollen- und Aufgabenverteilung werden sich verändern. Der Mehrwert der zugelieferten Systeme entsteht vor allem durch Software und vernetzte Fahrzeugfunktionen.

Fazit

Die Relevanz von Software im Fahrzeug ist unbestritten. Softwarekonzepte und Softwaretechnologien haben das Potenzial die Komplexität durch die Vernetzung von immer mehr Fahrzeugfunktionen zu beherrschen. Außerdem ermöglicht der Paradigmenwechsel hin zu „Software als Produkt“ die Hardware-unabhängige Konzeption, Entwicklung und Vermarktung von Software.

Die BMW Car IT ist eine strategische Initiative der BMW Group, um Chancen zu nutzen, die sich durch einen Software-basierten Gestaltungsansatz zur Beherrschung hochintegrierter Funktionen im Fahrzeug ergeben. Durch die Positionierung der BMW Car IT sind Innovationskraft und Seriennähe gewährleistet. In den Entwicklungsbereichen „Software Transfer“, „Human Vehicle Interaktion“ und „Integriertes Datenmanagement“ werden Konzepte der Informatik in das Fahrzeug gebracht.

Dabei bieten sich auch Chancen für Softwarehersteller, das Fahrzeug als neues Anwendungsfeld für etablierte Softwarekomponenten zu erschließen.

Integration agiler Prozesse in die Softwaretechnik-Ausbildung im Informatik-Grundstudium

Petra Becker-Pechau, Wolf-Gideon Bleek, Axel Schmolitzky, Heinz Züllighoven

Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik,
{becker, bleek, schmolitzky, zuellighoven}@informatik.uni-hamburg.de

Zusammenfassung

Objektorientierte Programmierung zu lehren erfordert einiges an Didaktik, da die Objektorientierung stark in die Praxis der Softwareentwicklung eingebunden ist. Das Paradigma muss aber nicht nur praktisch, sondern auch konzeptionell in den Software-Entwicklungsprozessen eingeordnet werden. Die Bezüge reichen von der Analyse über den Entwurf und die Implementierung bis hin zur Integration in bestehende Softwarelandschaften. Nach unseren Erfahrungen ist für eine solche Einordnung die klassische Veranstaltungsform mit Vorlesung und begleitenden, kleinen, bindungslosen Übungen nicht mehr angemessen. Wir planen deshalb für das Sommersemester 2003, die in die Objektorientierung einführende Grundstudiumsveranstaltung neu zu strukturieren. Dabei sollen Prinzipien agiler Entwicklungsprozesse in den Übungsteil eingehen. Agile Methoden sind durch ihre kurzen Projektzyklen, die sich iterativ entwickelnden Architekturen und die vielschichtigen Rückkopplungsmechanismen aus unserer Sicht besonders geeignet. Die Studierenden lernen dadurch relevante Fragenstellungen der objektorientierten Softwareentwicklung in kurzer Zeit aus eigener Erfahrung kennen und können darüber in Rücksprachen diskutieren. So können dann Konzepte in der Vorlesung auf wissenschaftlichem Niveau reflektiert werden.

1 Einführung

Objektorientierte Programmierung (OOP) hat in der Informatik-Lehre der Universität Hamburg eine lange Tradition. Bereits Anfang der 90er Jahre wurde eine zweisemestrige Vorlesung im Hauptstudium mit begleitenden Übungen angeboten. Daran nahmen pro Jahr 50 bis 60 Studierende teil. Sie lösten in den Übungen kleine

abgeschlossene Aufgaben mit geringem Bezug zu realen Problemen. Deshalb änderten wir die Lehrveranstaltung so, dass die Studierenden nach einer Phase der Stoffvermittlung eine größere Projektaufgabe bearbeiten sollten [5]. Diese Projektarbeit wurde wiederholt durch Stoffvermittlung unterbrochen. Die neue Veranstaltungsform wurde jeweils von ca. 30 Studierenden besucht, die in Gruppen von fünf bis sieben an einer selbstgewählten Aufgabe bis zu einem Jahr gearbeitet haben.

Inzwischen ist OOP ein Teil des Grundstudiums geworden. Dieses Hineinwachsen in das Grundstudium hat in der neuen Informatik-Studienordnung dazu geführt, dass imperative und objektorientierte Konzepte in einem Semester vermittelt werden müssen. Die entsprechende Veranstaltung *P2* findet im Sommersemester statt und hat um die 350 Teilnehmende. Als Programmierkenntnisse der Studierenden kann lediglich der Stoff von *P1* vorausgesetzt werden, der die funktionale und logische Programmierung umfasst.

In diesem Artikel reflektieren wir die bisherigen Erfahrungen aus *P2*, benennen Probleme und schlagen ein neues Konzept für diese einsemestrige Veranstaltung vor, das erstmals im Sommersemester 2003 an der Universität Hamburg umgesetzt werden wird. Abschließend werfen wir die Frage auf, ob die grundlegende Organisation der Lehrveranstaltung mit den gewünschten Lernzielen zu vereinbaren ist.

Der folgende zweite Abschnitt beschreibt zuerst die Ziele der Veranstaltung. Dann erläutern wir die bisherige Struktur der Lehrveranstaltung in Abschnitt 3 und zeigen in Abschnitt 4 die hierbei beobachteten Probleme auf. Im Abschnitt 5 legen wir das neue Konzept für die Lehrveranstaltung dar. Abschließend diskutieren wir eine grundsätzliche Umgestaltung der Lehre objektorientierter Konzepte.

2 Ziele der Veranstaltung „Objektorientierte Programmierung“

Um OOP angemessen zu lehren, müssen zum einen etliche inhaltliche Themen berücksichtigt werden:

- Grundelemente der imperativen Programmierung
- Ablaufsteuerung, Prozedurbegriff, Datentypen
- Wert und Objekt
- Objekt und Klasse
- Schnittstelle, Kapselung und Sichtbarkeit
- Vererbung
- Objektorientierte Typsysteme
- Modularisierung, Zugriffsrechte
- Referenzen und dynamische Datenstrukturen
- Abstrakte Datentypen und ihre Umsetzung
- Vertragsmodell
- Systematisches Testen

OOP als Tätigkeit kann mit Blick auf elementare softwaretechnische Anforderungen aber nicht ohne die folgenden Aspekte vermittelt werden:

- *Objektorientierte Konstruktion ist Konzeptarbeit gepaart mit „handwerklicher Tätigkeit“*
  Programmieren ist eine Tätigkeit, die eingeübt und praktisch erfahren werden muss. Erst auf dieser handwerklichen Grundlage lassen sich nachhaltig die dahinter stehenden Konzepte vermitteln und gewinnbringend einsetzen.
- *Zusammenhang von Analyse, Entwurf und Konstruktion*
  In einer Grundstudiumsveranstaltung sollte wenigstens in Ansätzen klar werden, dass (objektorientierte) Programmierung die konstruktive Lösung eines wenig strukturierten Anwendungsproblems liefern kann. Programmierung sollte als eine Form der Modellierung verstanden werden.
- *Professionelle objektorientierte Programmierung ist Teamarbeit*
  In Projekten arbeiten Informatiker fast immer in einem Team. Dadurch unterscheidet sich professionelle Programmierung von individueller Freizeitprogrammierung. Arbeitsteilung und Kommunikation gehören deshalb ebenso zur Teamarbeit [6] wie Pünktlichkeit, Verlässlichkeit, Planung und Kontrolle, soziales Verhalten und Sensibilität gegenüber anderen.

Objektorientierte Programmierung umfasst sehr viele, teilweise recht anspruchsvolle programmiersprachliche Konzepte. Diese Konzepte können nach unserer Sicht nur auf der Basis eigener praktischer Programmiererfahrungen eingeordnet und genutzt werden.

3 Die bisherige Veranstaltung

Zunächst halten wir die formalen Randbedingungen für die Lehrveranstaltung im Grundstudium der Hamburger Informatik fest:

- Die Veranstaltung umfasst ein Semester.
- Den Studierenden wurden zuvor noch keinerlei Kenntnisse über imperative oder objektorientierte Konzepte vermittelt.
- Das Format ist mit 2 SWS Vorlesung und 2 SWS Übung vorgegeben. Diese Aufteilung war nicht unumstritten. Von einigen Lehrenden wird auch die Ansicht vertreten, dass eine 3-stündige Vorlesung mit einer 1-stündigen Großübung aufgrund der Kapazitätsengpässe sinnvoll sei.
- Es stehen für die ca. 350 Teilnehmer nur sehr begrenzte Betreuungskapazitäten zur Verfügung. Zurzeit werden 13 Übungsgruppen mit einer Gruppenstärke von 15 bis 30 Personen angeboten. 4 bis 5 dieser Übungsgruppen werden von Studierenden betreut, die anderen von wissenschaftlichen Mitarbeitern.
- In den Übungsgruppen muss ein Leistungsnachweis erbracht werden. Dazu gibt es ein Punktesystem für die wöchentlich ausgegebenen Übungsblätter.

- Die gesamte Lehrveranstaltung soll im Semester in einem festen wöchentlichen Zeitraster stattfinden.

Die Lehrveranstaltung wird somit klassisch in Vorlesungen und Übungen aufgeteilt. Während die wöchentliche Vorlesung die bereits aufgeführten Themen und Aspekte vermittelt, dienen die Übungen der Wiederholung, Vertiefung und praktischen Erfahrung. Wöchentlichen Übungsblätter enthalten kleine Programmieraufgaben, mit denen elementare Programmierung erlernt und Themen der Vorlesung vertieft werden sollen. Daneben wird immer eine Textaufgabe gestellt, an der die Studierenden den vermittelten Stoff reflektieren sollen.

Die zweistündigen Übungen finden in rechnerlosen Seminarräumen statt. Lösungen können über einen Overheadprojektor oder einen Beamer mit Laptop präsentiert werden. Die Termine sind so strukturiert:

- Inhalte der Vorlesung wiederholen und zusammenfassen.
- Notwendige Vorkenntnisse für das nächste Übungsblatt vermitteln / wiederholen.
- Präsentation und Diskussion einer oder mehrerer studentischer Lösungen des zwei Wochen zuvor verteilten Übungsblattes. Dieses Übungsblatt wurde durch den Übungsgruppenleiter bereits korrigiert und bewertet; die Ergebnisse werden am Ende des Übungstermins verteilt. Die Präsentation der Lösungen ist ein Teil der Kriterien für den Leistungsnachweis.

Die Studierenden sollen die Übungen in Arbeitsgruppen von zwei bis vier Personen bearbeiten. So sollen sie Teamarbeit lernen. Gleichzeitig reduziert diese Gruppenarbeit den Korrekturaufwand der Übungsgruppenleiter auf ein realistisches Maß. Jede Übungsgruppe hat fünf bis zehn Arbeitsgruppen.

Jedes Jahr werden in der Vorbereitung die Aufgabenblätter samt Musterlösungen (bestehend aus kommentierten Programmen und Erklärungstexten) neu gestaltet und in einem Autor-Kritiker-Zyklus überarbeitet. Jedes Aufgabenblatt kostet mit seiner Musterlösung etwa eine Woche Arbeit, die sich durch den Überarbeitungsprozess über einen längeren Zeitraum verteilt.

Im letzten Semester wurde die Lehrumgebung BlueJ (siehe www.bluej.org) zum erstenmal eingesetzt. BlueJ ist eine einfach zu bedienende Programmierumgebung für objektorientierte Programmierung mit Java, die speziell für die Ausbildung von Programmieranfängern entwickelt wurde. BlueJ ermöglicht, den "Objects first"-Ansatz 1 zu lehren, d.h. die Studierenden können schon in den allerersten Aufgaben vorgegebene Klassen und Objekte „anfassen". Wenn diese Kapsel schrittweise geöffnet wird, lernen die Studierenden das Inenleben von Objekten (Exemplarvariablen, Methoden mit Variablen, Ausdrücken und Anweisungen) und damit die imperativen Grundkonzepte kennen. Dieser Ansatz vermeidet unter anderem die didaktischen Klimmzüge bei der erstmaligen Erklärung des Java-Konstrukts "public static void main(String[] args)", das mehr aus Ausnahmen als aus Basiskonzepten besteht.

4 Beobachtete Probleme

Vergleichen wir die Ziele der Veranstaltung mit ihrer Realisierung, so zeigen sich deutliche Diskrepanzen. Insbesondere die in Abschnitt 2 aufgeführten Lernziele, die über technische Inhalte hinausgehen, lassen sich in dieser Form nur schwer vermitteln.

- *Zusammenhang von Analyse, Entwurf und Konstruktion*
  Übungsaufgaben für diese drei Aspekte lassen sich nicht in einer Woche lösen. „Folgefehler" wären nicht zu vermeiden, so würde z.B. eine schlechte Analyse oder ein unangemessener Entwurf zu einer mangelhaften Konstruktion führen. Unmittelbares persönliches Feedback zu Zwischenlösungen ist hier essentiell. Die Verteilung über mehrere Übungsblätter löst dieses Problem auch nicht, da vor Ausgabe der Folgeaufgabe die vorangegangene persönlich besprochen werden müsste. Dazu müsste die abgegebene Lösung aber auch schon korrigiert worden sein.
- *Objektorientierte Konstruktion ist Konzeptarbeit gepaart mit „handwerklicher Tätigkeit"*
  Erst auf Basis von handwerklicher Programmiererfahrung lassen sich dahinter stehende Konzepte vermitteln und in ihrer Tragweite begreifen. Studierende benötigen meist mehrere Wochen, um die ersten programmiersprachlichen Fähigkeiten zu lernen. Durch parallele Übungen und Vorlesungen werden aber abstrakte Konzepte vermittelt, deren Erfahrung in den Übungen noch aussteht. Da in den Übungsstunden nicht programmiert wird, kann der Übungsgruppenleiter keine handwerklichen Fähigkeiten vermitteln. Viele Probleme oder Fortschritte lassen sich aber nach unserer Erfahrung nur direkt bei der Programmierung erkennen.
- *Professionelle objektorientierte Programmierung ist Teamarbeit*
  Obgleich die Studierenden ihre Aufgaben in Arbeitsgruppen von zwei bis vier Personen bearbeiten sollen, ist dies nicht notwendigerweise Teamarbeit. Oft werden die Aufgaben unter den beteiligten Personen aufgeteilt, um den Aufwand für den Leistungsnachweis zu optimieren. Die Übungsgruppenleiter kontrollieren die Leistung und geben Rückkopplung durch die Korrektur der Übungsaufgaben und anhand der Präsentation der Übungslösungen durch die Studierenden während der Übungstermine. Die Teamarbeit bleibt den Studierenden überlassen. Einwöchige Übungsaufgaben sind aber im Umfang zu gering, um die Koordination und Planung von Teamarbeit einzuüben.

Während der Veranstaltungen der letzten Jahren sind diese Konflikte zwischen Zielen und Durchführung immer deutlicher geworden. Es zeigten sich verschiedene Probleme:

- *Hoher Vorbereitungsaufwand*
  Jedes Jahr neue Übungsblättern zu erstellen ist aufwändig und teilweise frust-

rierend. Es ist schwierig möglichst kontext-unabhängige Aufgaben zu finden, weil die Themen aufeinander aufbauen und Trennlinien oft künstlich gezogen werden müssen. Für eine interessante und nachprüfbare Veranstaltung müssen aber jedes Jahr überarbeitete Übungen gefunden werden.

- *Hoher Durchführungsaufwand*
  Das schriftliche Korrigieren der Aufgabenlösungen ist sehr zeitaufwändig (ca. 4 bis 6 Zeitstunden pro Woche).
- *Geringer Praxisbezug*
  Aufgrund des (notwendigerweise) begrenzten Umfangs der Aufgaben ist ihr Praxisbezug minimal. Dies gilt insbesondere für Aufgaben zu fortgeschrittenen Konzepten wie Modularisierung oder Vererbung. Die Studierenden bekommen leicht den Eindruck, dass es bei der Software-Entwicklung eher um kleine, zusammenhangslose Aufgaben geht als um die langfristige Arbeit an einem größeren Projekt.
- *Keine Berücksichtigung von Vorkenntnissen*
  Die Aufgaben sind für einige Studierende zu einfach, für andere zu schwierig. Dies langweilt Studierende mit Vorkenntnissen in der ersten Hälfte der Veranstaltung, während schwächere Studierende in der zweiten Hälfte überfordert sind. In beiden Fällen besteht die Gefahr, dass die Teilnahme wegen Misserfolg abgebrochen wird, denn Studierende mit vielen Vorkenntnissen verpassen häufig den Moment, an dem ihre Vorkenntnisse nicht mehr ausreichen.
- *Zu kurze Übungstermine*
  Die zweistündigen Übungstermine sind zu kurz für den mit ihnen verbundenen Anspruch. Allein die Vertiefung des Vorlesungsstoffs würde die Zeit ausfüllen; zusätzlich müssen jedoch alte Aufgaben besprochen und neue Aufgaben vorgestellt werden.
- *Zu wenig individuelles Feedback*
  Die individuelle Rückkopplung kommt für die Studierenden zu kurz. Gute Lösungen können nicht ausreichend gewürdigt werden, bei schlechten Lösungen wären persönliche Gespräche hilfreich. So lassen sich z.B. Programmierrichtlinien am besten direkt am Bildschirm diskutieren.
- *Arbeitsteilung statt Gruppenarbeit*
  Studierende können sich für eine Gruppenlösung eintragen, ohne an der Lösung mitgearbeitet zu haben. Die Tendenz steigt, die Veranstaltung lediglich als „Punktesammeltreffen“ aufzufassen, um den notwendigen Leistungsnachweis zu bekommen. Die Gruppenarbeit wird durch die Studierenden oft lediglich zur Effizienzsteigerung benutzt.
- *Zu viel Raum für Täuschungsversuche*
  Die aktuelle Gestaltung der Übungen erleichtert den Austausch von Lösungen unter den Studierenden sehr, da dies über Übungsgruppen hinweg nur schwer festzustellen ist. Täuschungsversuche frustrieren sowohl die ehrlichen Studierenden als auch die Übungsgruppenleitungen.

Neben den organisatorischen Problemen und den in einem prüfungsorientierten Lehrbetrieb üblichen Reibungsverlusten fällt die Abstimmung zwischen Lehrinhalt und den didaktisch-organisatorischen Formen besonders ins Gewicht. Die klassische Trennung zwischen Vorlesung und Übung passt offensichtlich schlecht zu den vermittelten Inhalten. Durch die strengen Rahmenbedingungen (Punktevergabe, Anwesenheit, Leistungsnachweis) verkehren sich die begrenzten didaktischen Mittel (Gruppenarbeit, Versuch von Praxisbezug) ins Negative.

5 Der neue Ansatz: Agile Prozesse in der Softwaretechnik-Ausbildung

Der neue Ansatz soll im Grundstudium einen realistischen Eindruck von objektorientierter Softwareentwicklung vermitteln und dabei die vorgenannten Probleme vermeiden. Als Nebeneffekt soll die Arbeitsbelastung der Mitarbeiter in einem guten Verhältnis zum Lernerfolg der Studierenden stehen.

In einem projektartigen Übungsbetrieb [2, 8] werden die Studierenden durch eigene Erfahrung an die typischen Aufgaben und Fragestellungen der objektorientierten Softwareentwicklung herangeführt. Die Übungsgruppenleiter können bei der Projektarbeit viel unmittelbarer und effektiver Rückkopplung geben und den Lernerfolg verfolgen, als dies mit Korrekturen von Übungszetteln der Fall ist.

Projekte nach agiler Vorgehensweise werden in kurzen Iterationszyklen durchgeführt, in denen alle typischen Entwicklungsaufgaben von der Analyse bis zur Konstruktion enthalten sind. Ein solcher Übungsbetrieb vermittelt den Studierenden innerhalb kürzester Zeit relevante Erfahrungen. Vielfältige Rückkopplungsmechanismen, von der Arbeit in Paaren [10] bis zur Abnahme einer Softwareversion am Ende einer Iteration, können den Lernerfolg beschleunigen.

Die ursprüngliche Idee

Das neue Konzept sollte ursprünglich eine größere Projektaufgabe umfassen, an der die Studierenden während der gesamten 14 Wochen des Semesters nach Aspekten des Extreme Programming [4] arbeiten. Diese Aufgabe sollte genügend Raum bieten, um die unterschiedlichen Aspekte von OOP zu erlernen. Eine absichtlich unscharfe Aufgabenstellung sollte als roter Faden durch den Stoff der Veranstaltung führen. Diese Aufgabenstellung sollte in kleinen Gruppen projektartig bearbeitet werden.

Es waren regelmäßige Rückkopplungstreffen zwischen dem Übungsgruppenleiter, nun besser als Tutor bezeichnet, und den ihm zugeordneten Gruppen geplant. Dabei sollten Entwurfsentscheidungen und Implementierungen diskutiert und bewertet werden.

Der Tutor sollte über die gesamte Zeit einen kontinuierlichen Entwicklungs- und Rückkopplungsprozess anleiten, der wiederholt durch neue Teilaufgaben angerei-

chert wird. Einige wenige Projektthemen sollten zur Verfügung stehen (z.B. „Pizza Lieferservice“, „Autovermietung“, „Taxizentrale“, „Kinokartenverkauf“, „Bibliotheksausleihe“) [3].

Dieser Ansatz erfordert allerdings, dass die Studierenden bereits elementare imperative und objektorientierte Programmierkenntnisse besitzen. Da diese im Rahmen der Veranstaltung erst vermittelt werden, haben wir uns für einen Kompromiss entschieden.

Der geplante pragmatische Ansatz

Die zur Verfügung stehende Zeit wird aufgeteilt in eine *Laborphase* und eine *Projektphase*. Die Laborphase soll das „Programmieren im Kleinen“ abdecken, während in der Projektphase fortgeschrittene Konzepte (ADTs, Modularisierung, Sichtbarkeitsregeln etc.) anhand einer größeren Projektaufgabe vermittelt werden. Die Laborphase soll acht Wochen, die Projektphase sechs umfassen. Die Vorlesung soll weiterhin die Konzepte der objektorientierten Programmierung vermitteln. Idealerweise werden besonders in der Projektphase die Themen der Vorlesung den unmittelbaren Anforderungen aus den anstehenden Projektaufgaben angepasst.

Laborphase in Zweiergruppen

In der Laborphase werden wie bisher Aufgabenblätter verteilt und wöchentlich bearbeitet. Statt einer schriftlichen Lösung werden die Aufgaben in betreuten Laborsituationen bearbeitet und testiert. Auf diese Weise entfallen der 4- bis 6-stündige Korrekturaufwand für die Übungsgruppenleiter und das Warten der Studierenden auf Rückkopplung. Diese Zeit verbringen sie stattdessen in den Labors, während sich die Studierenden jeweils zu Paaren zusammenfinden und gemeinsam am Rechner die Aufgaben bearbeiten. Die persönliche Betreuung ermöglicht dem Tutor, die individuellen Vorkenntnisse sowie Probleme der einzelnen Studierenden während der Programmierung wahrzunehmen. So kann der Tutor sehr viel unmittelbarer und persönlicher Feedback geben und erhalten. Wir planen Laborblöcke von jeweils drei Zeitstunden. Da innerhalb dieser Blöcke auch die Anteile der Übungen stattfinden sollen, die bisher in den zweistündigen Übungen positioniert waren, kann ein Tutor bei gleichem Zeitaufwand zwei dieser Blöcke im Labor betreuen (siehe die Aufstellung am Ende dieses Abschnitts).

Projektphase mit Gruppen von 6-8 Studierenden

In dieser Phase werden wir die bisherige Veranstaltungsform am deutlichsten verändern. Lediglich Studierende, die an der Laborphase erfolgreich teilgenommen haben, werden für die Projektphase zugelassen, da elementare Programmierkenntnisse eine notwendige Voraussetzung für die Programmierarbeit im Projekt darstellen.

Die Laborzeiten werden in der Projektphase fortgeführt, jedoch mit anderem Schwerpunkt. Die Studierenden sollen sich erneut zu Gruppen zusammenfinden,

diesmal zu Gruppen von sechs oder acht (drei oder vier Paare). Jede dieser Gruppen bildet ein Projektteam. Für die Projektorganisation werden Konzepte agiler Entwicklungsmethoden genutzt [7]. Dies umfasst u.a. die Arbeit in Paaren, die Organisation in Iterationen sowie das Prinzip des einfachen Entwurfs und Anpassungen der Konstruktion mit Hilfe von Refactorings [9].

Die Organisation als Projekt ermöglicht, die in Abschnitt 2 aufgeführten Aspekte in der Lehre direkt zu adressieren:

- Der Zusammenhang von Analyse, Entwurf und Konstruktion wird im Projekt erfahren. Zwar werden Analyse und Entwurf durch den Übungsgruppenleiter angeleitet und teilweise vorgegeben, da im Grundstudium die Konstruktion im Vordergrund steht. Dennoch muss im Projekt analysiert und entworfen werden.
- Relevante Fragestellungen ergeben sich auch aus der Größe der Aufgabenstellung und der projektartigen Organisation und werden von den Tutoren aufgegriffen und in Zusammenhang mit der Vorlesung gestellt. So kann das „handwerkliche Begreifen“ konzeptionelle Überlegungen anregen.
- Die Arbeit in Teams an einer gemeinsamen, längeren Aufgabe ermöglicht, typische Herausforderungen der Teamarbeit in objektorientierten Projekten zu erfahren. Die Studierenden werden sensibilisiert für Fragen der Aufgabenplanung und -teilung. Übungsblätter mit kleinen, genau spezifizierten Aufgaben legen fälschlicherweise nahe, dass es für Aufgaben korrekte, eindeutige Lösungen gibt, die direkt zu Beginn der Konstruktion geplant werden können. Diese Gefahr ist in der Projektarbeit geringer. Hier soll vorrangig über mehr oder weniger angemessene Entwurfs- und Konstruktionsalternativen vor dem Hintergrund der unscharfen Anforderungen diskutiert werden. Es soll deutlich werden, dass „richtig“ und „falsch“ keine objektiven Kriterien für Softwareentwürfe sind. Neue Anforderungen im Verlauf des Projekts regen zu Umstrukturierungen (Refactorings [9]) an und unterstreichen den vorherigen Punkt.

Zu Beginn der Projektphase wird den Studierenden in einer konstituierenden Sitzung das Projektthema vorgestellt (etwa "eine Verwaltungssoftware für ein Kino erstellen"). Sie erhalten ein Kernsystem, das minimale Anforderungen bereits erfüllt. Diese Kernsystem gibt eine Grundarchitektur der zu erstellenden Anwendung vor; es wird im Laufe des Projekts von den Studierenden erweitert und überarbeitet.

Während des 6-wöchigen Projekts werden den Studierenden drei Impulse für 2-wöchige Iterationen gegeben, indem sie weitere Anforderungen erhalten. Dies kann z.B. sein „Implementieren Sie eine Kundenverwaltung!“ Während sie versuchen, die domänenspezifischen Inhalte zu erarbeiten, müssen sie gleichzeitig die korrespondierenden objektorientierten Konzepte finden. Wöchentliche Treffen mit dem Tutor begleiten diesen Prozess. Der Tutor erläutert die Aufgaben, gibt Hilfestellung beim Entwurf und der Aufgabenverteilung und kommentiert die Ergebnisse. Er leitet den

Prozess an und sorgt für eine tragfähige Gesamtarchitektur, die er implizit in den Design-Diskussionen motiviert. Unterstützt wird er hierbei durch das vorgegebene Kernsystem. Auf diese Weise erhalten die Studierenden eine Modellarchitektur, ohne dass dieses Thema explizit aufgegriffen werden muss. Modellarchitekturen sind Thema weiterführender Veranstaltungen im Hauptstudium. In den wöchentlichen Treffen präsentiert die Projektgruppe ihre Arbeit der vergangenen Woche. Der Tutor stellt Fragen. Die Vorstellung wird vom Tutor protokolliert; Fragen und offene Probleme werden für den nächsten Termin festgehalten. Der Tutor wird der Gruppe spezifische Anforderungen stellen, die dann umgesetzt werden sollen. Da die neuen Anforderungen auf der Basis bereits geleisteter Arbeit umgesetzt werden, wird das bereits Gelernte wiederholt.

Der Lernerfolg wird in der Projektphase nicht mehr durch gelöste Übungsaufgaben definiert, sondern über die Fähigkeit, neu angeeignete Konzepte für eine reale Aufgabenstellung anzuwenden. Dies ist kein formales Kriterium, sondern muss den Umständen entsprechend bewertet werden. Auch in der Projektphase betreut der Tutor die Studierenden während der Programmierung. Dadurch kann er die Fortschritte, das Verständnis und die individuellen Probleme der einzelnen Projektmitglieder weiterhin verfolgen und ummittelbares Feedback geben.

Randbedingungen des neuen Ansatzes

Betreuungsaufwand: An der Veranstaltung werden voraussichtlich 320 Studierende teilnehmen. Die Studierenden werden in Gruppen von 20 Personen jeweils 3 Zeitstunden in der Universität an ihren Aufgaben arbeiten sowie in dieser Zeit Rücksprache mit dem Tutor halten. Es werden immer zwei Gruppen parallel in benachbarten Räumen stattfinden; diesen 40 Studierenden werden zwei Tutoren und eine studentische Hilfskraft für technische Fragen zur Verfügung stehen. Ein wissenschaftlicher Mitarbeiter mit einer Lehrverpflichtung von 2 SWS wird zwei solcher Gruppen betreuen, also insgesamt 6 Stunden anwesend sein. Dies entspricht dem Zeitaufwand der klassischen, übungsorientierten Organisation (siehe Abschnitt 4), da in der neuen Organisationsform die zusätzlichen Korrekturen der Übungsaufgaben wegfallen. Diese Tätigkeit wird in Anwesenheit der Studierenden ausgeführt.

Die folgende Gegenüberstellung der Aufwände in Zeitstunden/Woche für eine Lehrverpflichtung von 2 SWS verdeutlicht, dass sich die Aufwände in der Betreuung nicht erhöhen.

Bisheriger übungsorientierter Lehrbetrieb:

Koordinationstreffen der Übungsgruppenleiter	1
Übungstermin (mit 26 Personen):	2
Korrekturen und Vorbereitung des Übungstermins:	5
Gesamtsumme	8

Neues Konzept (in beiden Phasen):

Koordinationstreffen der Übungsgruppenleiter	1
Betreuung von zwei Gruppen á 20 Personen in Anwesenheit	6
Vor- und Nachbereitung	1
Gesamtsumme	8

Bisher wurden durchschnittlich 26 Studierende von einem Tutor betreut, bei ca. 320 Studierenden wurden also 13 Betreuer benötigt. Jetzt werden zwei Gruppen mit jeweils 20 Studierenden von einem Tutor und zusätzlich in Halbzeit von einer studentischen Hilfskraft betreut. Dies ergibt einen Bedarf von 9 Tutoren und 4 studentischen Hilfskräften.

Wir gehen also davon aus, dass der zeitliche Aufwand sich nicht erhöhen wird. Hinzu kommt, dass während der Projektphase die Erstellung von detailliert ausgearbeiteten Aufgabenzetteln und Musterlösungen wegfällt. Lediglich die Zeiteinteilung der Tutoren wird unflexibler.

Das neue Konzept verlangt deutlich höhere Kompetenzen der Tutoren durch die Bewertung und Hilfestellung in Anwesenheit sowie die implizite Anleitung der Gesamtarchitektur eines wachsenden Systems bei unscharfer Aufgabenstellung. Als Tutoren werden deshalb ausschließlich wissenschaftliche Mitarbeiter eingesetzt. Studentische Hilfskräfte bewerten die Lösungen nicht, wie es in der vorherigen Organisationsform der Fall war, sondern unterstützen die Tutoren bei der Gruppenbetreuung.

6 Kritische Überlegungen

Das neue Modell für die Grundstudiums-Lehrveranstaltung ist für uns ein Kompromiss, der unsere Wunschvorstellungen mit den aktuellen Möglichkeiten im Hamburger Grundstudium verbindet. Allerdings fragen wir uns, ob nicht eine radikalere Revision des Lehrekonzepts notwendig ist.

Universitäre Informatik umfasst zahlreiche Themen – die anwendungs-, d.h. praxisorientierte, Softwareentwicklung nach dem objektorientierten Paradigma ist nur eines. Doch scheint uns ein Informatikstudium ohne solide handwerkliche Fähigkeiten in der Softwarekonstruktion verbunden mit den dahinter stehenden Konzepten schwer vorstellbar. Dies weist einer Lehrveranstaltung „objektorientierte Softwareentwicklung“ im Grundstudium ihren Platz zu. Hier sollen, aus unserer Sicht, dieses Handwerkszeug und die dazu passenden Konzepte elementar vermittelt werden.

Wenn wir uns von den aktuellen organisatorischen und vor allem strukturellen Randbedingungen befreien könnten, würden wir eine andere Vorgehensweise wählen. Denn wir haben umfangreiche Erfahrungen in der beruflichen Aus- und Weiterbildung von konventionell arbeitenden Programmierern zu objektorientierten Anwendungsentwicklern. Dort gehen wir, kurz gefasst, so vor:

- Wir vermitteln zunächst ganz elementare Begriffe und Konzepte der Programmierung (z.B. Variable, Algorithmus, Ablaufsteuerung) im Seminarstil. Dabei wechseln kurze Vortragseinheiten mit kleinen „Fingerübungen“ in betreuter Gruppenarbeit ab.
- Nach ca. einer Woche lernen die Kursteilnehmer intensiv (d.h. ganztägig) die Basiskonstrukte einer objektorientierten Programmiersprache. Dabei werden vom Dozenten jeweils einzelne Konstrukte und Sprachmerkmale (teils interaktiv am Rechner) vorgestellt, die dann unmittelbar in kleinen Übungen ausprobiert werden. Dabei arbeiten die Teilnehmer in Programmierpaaren (nach den Prinzipien des Extreme Programming [4]) unter Anleitung eines erfahrenen Trainers.
- Nach zwei bis drei Wochen beherrschen die Teilnehmer die Basiskonstrukte. Dann werden weitergehende Techniken und Konstruktionen vorgestellt und in „Miniprojekten“ umgesetzt. Neben den Konstruktionsaufgaben steht die Projektarbeit im Team an erster Stelle. Konzepte werden dann vom Projekt-Coach vorgestellt, wenn sie ein aufkommendes Problem im Projekt lösen helfen.

Wir haben diese Ausbildungsform in mehreren innerbetrieblichen Schulungszyklen erprobt. Dass sie deutlich erfolgreicher ist, lässt sich durch zwei Feststellungen belegen:

- Die Resonanz ist sehr positiv und das Engagement der Teilnehmer ist sehr viel höher als bei unseren früheren Ausbildungsseminaren, die sich eher an konventionellen Schulungsformen orientierten.
- Die Erfolgsquote ist sehr hoch. Durch Rücksprachen können wir die fachlichen Kenntnisse der Teilnehmer gut einschätzen. Da wir oft auch als Coaches in nachfolgenden Projekten des Unternehmens arbeiten, können wir auch die Eignung der ehemaligen Trainees für die Projektarbeit einschätzen. Zwischen 80 und 90 % der Teilnehmer erreichen das gesteckte Kursziel.

Was bedeutet dies bezogen auf die universitäre Ausbildung?

- Das Verhältnis von Vorlesung, d.h. Vortrag, und Übung muss umgekehrt werden. Die Übungen dürfen nicht mehr nur Erläuterungen dessen sein, was in der Vorlesung an Konzept und Theorie vorgetragen wird. Die praktische Auseinandersetzung mit Aufgaben und Projektthemen wird zum treibenden Motor. Wenn dabei Fragen aufkommen, können konzeptionelle Lösungsansätze direkt aufgegriffen und verarbeitet werden. Damit werden Vortragseinheiten ereignisgetrieben oder situativ in den Lernprozess integriert. Wenn genügend Erfahrungswissen vorliegt, lassen sich in der Reflexion weitergehende Konzepte bruchlos in die Projektarbeit einbringen.
- Wenige elementare Grundkenntnisse der Programmierung reichen aus, um mit der problem- und aufgabenorientierte Projektarbeit zu beginnen. Soweit sie bei

Studierenden nicht ohnehin aus der Schulzeit vorhanden sind, können sie leicht in einem einführenden Schnellkurs vermittelt werden.

- Die Orientierungen an wöchentlichen Aufgaben mit „objektiven" Punktesystemen drängen die Übungsgruppen in die Richtung konventioneller Schulformen – Übungsaufgaben werden gestellt und Ergebnisse werden mit Blick auf eine Musterlösung bewertet und besprochen. Dies verstellt den Blick für die Probleme und Lösungsstrategien in der praktischen Softwareentwicklung. Statt der schulischen Bewertung von „richtig" und „falsch" sollte eine Diskussion über „angemessen" und „unangemessen" treten, in der dann die unterschiedlichen Sichtweisen und Wertesysteme von Auftraggebern und Benutzern sowie den Entwicklern deutlich werden.
- Erfahrungen im Programmieren und in der Projektarbeit lassen sich am besten intensiv, d.h. ganztägig, vermitteln. Diese Einsicht verträgt sich nicht mit den konventionellen Lehreschemata deutscher Universitäten. Die Zwänge, die sich aus den festgelegten Veranstaltungsformen und -rastern, der Raumvergabe und der Verknüpfung von Hauptstudium und Nebenfach ergeben, verhindern länger laufende Projektveranstaltungen. Einzelne deutsche und vor allem zahlreiche skandinavische Universitäten haben allerdings erfolgreich diese Umstrukturierung realisiert.
- Die Studierendenzahlen und die personelle Ausstattung der universitären Informatik passen nicht zusammen. Länger laufende Studienprojekte mit intensiver Softwareentwicklung erfordern ein hohes Maß an qualifizierter Betreuung. Die unmittelbare Diskussion von Programmentwürfen und rasche Hilfen bei Konstruktionsproblemen erfordern eine höhere Qualifikation als die Korrektur von schriftlichen Übungsaufgaben. Diese Qualifikation gibt es in der universitären Informatik derzeit nicht in ausreichendem Maße.
- Der Aufwand für Lehre und Betreuung im Studium steht heute in keinem gesunden Verhältnis zu den Lernerfolgen. Unsere eigenen Erfahrungen aus der Übungsbetreuung, den Grundstudiumsprüfungen und der Zusammenarbeit mit Studierenden im Hauptstudium sagen uns, dass nicht annähernd die Erfolgsquoten oder das Teilnehmerengagement unserer Ausbildungsseminare erreicht werden. Wobei wir, in aller Bescheidenheit, trotzdem der Meinung sind, dass wir unseren Studierenden eine vergleichsweise hochwertige Ausbildung mitgeben.

Vielleicht zeigt sich an der Problematik der Grundstudiumslehre im Bereich objektorientierter Programmierung das Dilemma der universitären Informatik in Deutschland: Wie lassen sich die Ansprüche nach einer Ausbildung zum Wissenschaftler mit den Anforderungen der gesellschaftlichen Praxis verbinden? Wir glauben nicht, dass darin ein unüberbrückbarer Widerspruch liegt; vielmehr sollte in einer wesentlich von Informationstechnologie geprägten Gesellschaft eine praxisbezogene Informatik ebenso selbstverständlich sein wie eine forschungsnahe Praxis.

Literatur

1. Barnes, D.J., and Kölling, M.: Objects First with Java – A Practical Introduction using BlueJ, Prentice Hall / Pearson Education, New York, 2003.
2. Bastian, J., and Gudjons, H.: Das Projektbuch, Bd. 1, Theorie, Praxis, Erfahrung. Bergmann/Helbig, Hamburg, Germany, 4. Auflage, 1994.
3. Bastian, J., and Gudjons, H.: Das Projektbuch, Bd. 2, Über die Projektwoche hinaus, Projektlernen im Fachunterricht. Bergmann/Helbig, Hamburg, 3. Auflage, 1998.
4. Beck, K.: eXtreme Programming explained – embrace change, Addison-Wesley, Boston, 2000.
5. Bleek, W.-G., Gryczan, G., Lilienthal, C., Lippert, M., Roock, S., Wolf, H., Züllighoven, H.: Von anwendungsorientierter Softwareentwicklung zu anwendungsorientierten Lehrveranstaltungen - der Werkzeug & Material-Ansatz in der Lehre, Software Engineering im Unterricht der Hochschulen (SEUH) 99, Berichte 52, B. Dreher/Ch. Schulz/D. Weber-Wulff (Hrsg.), Workshop des German Chapter of the ACM und der Gesellschaft für Information (GI) am 25. und 26. Februar 1999 in Wiesbaden, S. 9-20, 1999.
6. Clark, H. H., Brennan, S. E.: Grounding in Communication. In Resnick, L., Levine, J. M., Teasley, S. D. (eds.): Perspectives on Socially Shared Cognition. Washington, USA, 1991.
7. Cockburn, A.: Agile Software Development, Addison-Wesley, Boston, 2002.
8. Dewey, J.: How We Think: A Restatement of the Relation of Reflective Thinking to the Educative Process. Chicago, USA, 1933.
9. Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison-Wesley, Boston, 1999.
10. Williams, L., Kessler, R.: All I Really Need to Know about Pair Programming I Learned in Kindergarten, *Communicationsof the ACM*, Vol. 43, No. 5, 2000, S. 108

“Our educational institutions have failed …”

Andreas Spillner
Hochschule Bremen, Zentrum für Informatik und Medientechnologien
spillner@hs-bremen.de

Zusammenfassung

Neben den Hochschulen, als klassische Ausbildungsstätten für Informatiker[1] *mit jeweils unterschiedlicher Ausrichtung, sind eine große Zahl von kommerziellen Trainingsanbietern vorhanden. Meist werden in Weiterbildungskursen spezielle Themen aus dem IT-Bereich angeboten. Eine Tendenz ist derzeit zu erkennen, auch Grundlagenwissen in Kursen zu vermitteln.*

Ein dreistufiges Qualifizierungsprogramm im Bereich Softwaretest wird vorgestellt, das in England konzipiert und von vielen Ländern, unter anderem auch von Deutschland, aufgegriffen wurde. Der Beitrag ist als Diskussionsanregung gedacht und soll unterschiedliche Positionen verdeutlichen.

1 Einleitung

Der Titel des Beitrags ist ein Zitat aus dem Geleitwort von David Parnas zum Buch “Basiswissen Softwaretest” [1]. David Parnas schreibt unter anderem in seinem Geleitwort Folgendes:

“*… Software is well known for low reliability and lack of trustworthiness. In part this is attributable to the difficulty of dealing with the complexity of today's software systems, but the inadequate knowledge, skills, and professionalism of many of the practitioners also contributes to this problem. Moreover, we can thank the inadequately qualified people who produced today’s software for the unreliability and complexity of the products that serve as support software for new products.*

Our educational institutions have failed the public in this field. They have not recognized that those who study Computer Science require a professional education, one similar in style to the education provided to those who study medicine, law, or engineering. In those fields, the curriculum is designed around a set of professional requirements. Students are told what they must learn, rather than allowed to learn what they feel like learning. In the software field, universities have allowed the contents of courses to depend on the whim of the instructor, and the choice of courses to be largely up to the student. As a result, when an employer or client meets a graduate of a Computer Science programme, only experienced software developers are

[1] Es wird im Beitrag die männliche Form verwendet, Frauen sollen dadurch selbstverständlich nicht ausgeschlossen oder ausgegrenzt werden.

able to judge whether or not a graduate has the knowledge and skills needed for the job. Often, we cannot even find a graduate who has the appropriate body of knowledge and experience. …" [1, S. viii]

Parnas verlangt von einer Ausbildung zum Software-*Ingenieur* ein ähnlich festgelegtes Curriculum und Prüfungswesen, wie es bei den Medizinern, Juristen und Ingenieuren üblich ist. Nur Personen, die eine solche Ausbildung erfolgreich abgeschlossen haben, sollen Software entwickeln bzw. ändern dürfen. David Parnas ist der Ansicht, dass die Ausbildung an den Hochschulen den Erfordernissen in der Praxis nicht gerecht wird, da nicht klar ist, welche Fähigkeiten die Absolventen an den Hochschulen erworben haben (s. a. [2]).

Die Hochschulen können und sollen nach meiner Einschätzung Grundlagenwissen vermitteln. Allerdings gehen die Meinungen, was zu den Grundlagen der Informatik dazu gehört, weit auseinander. Es gibt Empfehlungen der Gesellschaft für Informatik für Informatik-Studium, -Ausbildung, -Fortbildung und -Weiterbildung ([4], [5], [6]). Die Empfehlungen belassen aber einen weiten Spielraum für unterschiedliche Ausprägungen und Schwerpunktsetzungen durch den einzelnen Lehrenden. So wird beispielsweise in den ergänzenden Empfehlungen der Gesellschaft für Informatik zu den Lehrinhalten und Veranstaltungsformen im Informatik-Studium aus dem Jahre 1997 (s. [6]) eine stärkere Berücksichtigung der Softwaretechnik im Informatik-Studium gefordert. Eine beispielhafte Liste der Lehrinhalte umfasst 23 Positionen, eine davon ist *Testen*. Es gibt darüber hinaus keine Hinweise, was alles zum Thema Testen gehört und somit zu lehren ist.

Das *Software Engineering Coordinating Committee* der IEEE hat im SWEBOK-Projekt das aktuelle Wissen im Bereich Software Engineering zusammengestellt [7]. Im Kapitel 5 sind alle bekannten Testverfahren strukturiert aufgelistet. Eine Zuordnung der einzelnen Verfahren untereinander und eine Einordnung zu Oberbegriffen wird vorgenommen. Die einzelnen Verfahren werden nur kurz beschrieben und es wird auf die ausführlichere Literatur verwiesen. Die umfangreiche Auflistung der Verfahren ist wenig hilfreich bei der Auswahl der Verfahren, die in der Lehre behandelt werden sollen.

Die an den Hochschulen vermittelten Grundlagen sollen die Studierenden in die Lage versetzen, sich in die aktuellen Themen und Anforderungen im Beruf schnell und umfassend einarbeiten zu können. In der IT-Branche kann keiner ohne lebenslanges Lernen den sich schnell ändernden Anforderungen im Beruf gerecht werden. Sicherlich auch deshalb gibt es seit längerem neben der Ausbildung durch die Hochschulen Weiterbildungsangebote von kommerziellen Ausbildungseinrichtungen im IT-Bereich.

Die Kurse konzentrieren sich meist auf aktuelles Wissen für bestimmte Bereiche, Spezialwissen wird vermittelt. Viele der Kurse können mit einer Prüfung und einem Zertifikat abgeschlossen werden. Derzeit gibt es ein großes Angebot an Kursen mit entsprechenden Abschlüssen und Zertifikaten (s. z. B. [8]).

Darüber hinaus werden auch Kurse angeboten zum Erwerb von Grundlagenwissen. Diese Kurse stehen dann in Konkurrenz zur Ausbildung an den Hochschulen. Eine solche Initiative soll im Beitrag näher vorgestellt werden und zu einer Diskussion anregen, ob die Hochschulen solche Initiativen aufgreifen sollen und das Lehrangebot entsprechend anpassen oder – etwas zugespitzt formuliert – die Freiheit von Forschung und Lehre als hohes Gut beizubehalten ist. Weiterbildungsmaßnahmen, die sich auf ein Produkt oder eine Produktfamilie eines Herstellers beziehen werden nicht diskutiert (z. B. Kurse von Microsoft oder Cisco).

2 Certified Tester[2]

Weiterbildungsmaßnahmen werden von vielen Firmen angeboten. Die Qualität der Kurse läßt sich oft erst nach einer Teilnahme bewerten. Bei einem schlechten Kurs ist es dann allerdings schon zu spät, Zeit und Geld sind relativ nutzlos vergeudet. Um eine bessere Vergleichbarkeit zumindest der Kursinhalte zu gewährleisten, ist in England das *Information Systems Examinations Board* (ISEB [10]) gegründet worden. Es ist der *British Computer Society* [11] angegliedert, die Organisation ist mit der deutschen Gesellschaft für Informatik e.V. vergleichbar.

ISEB ist in unterschiedlichen Bereichen tätig. Im Bereich Softwaretest wurden von einem Gremium von Fachexperten 1997 Lehrinhalte für ein dreistufiges Qualifizierungsprogramm definiert. Die Grundlagen zum Softwaretest sind im Lehrplan zum Erhalt des *Foundation Certificate* [12] beschrieben. Entsprechende Kurse werden ab 1998 angeboten. Darauf aufbauend kann das *Practitioner Certificate* [13] erworben werden, um vertiefte Kenntnisse im Prüfen und Testen während der Softwareentwicklung nachzuweisen. Ein drittes weiterführendes Zertifikat, das *Practitioner Diploma*, ist in Planung.

Kursanbieter müssen sich von ISEB akkreditieren lassen, d. h. die Kursinhalte werden auf Übereinstimmung mit dem laut Lehrplan zu behandelnden Themen geprüft. Jeder Kursteilnehmer kann sich einer Prüfung durch ISEB unterziehen und bei Bestehen ein entsprechendes Zertifikat erhalten. Die Prüfungen werden völlig unabhängig von den Kursanbietern von ISEB durchgeführt. Die Prüfungsfragen stehen den Kursanbietern nicht zur Verfügung. Personen können sich auch ohne vorherige Teilnahme an einem Kurs einer Prüfung unterziehen.

Die ISEB-Aktivitäten wurden von anderen europäischen Ländern aufgegriffen und ähnliche Initiativen wurden gestartet. Es sind landesspezifische *Testing Boards* etabliert worden, die gemeinsam das *International Software Testing Qualification Board* [14] bilden.

Die landesspezifischen *Testing Boards* sind zuständig für die Akkreditierung von Trainingsanbietern und die Durchführung von Prüfungen in ihren jeweiligen Ländern. Das *International Software Testing Qualification Board* koordiniert die natio-

[2] Ausführliche Informationen unter [9]

nalen Initiativen und sorgt für die Einheitlichkeit und Vergleichbarkeit der Lehr- und Prüfungsinhalte unter den beteiligten Ländern.

In Deutschland hat der ASQF (Arbeitskreis Softwarequalität Franken e. V. [15]) und die Fachgruppe TAV (Test, Analyse und Verifikation von Software, FG 2.1.7 [16]) der Gesellschaft für Informatik e. V. die ISEB-Aktivitäten aufgegriffen und ein entsprechendes Qualifizierungs- und Prüfungsschema aufgebaut.

Die fachlichen Inhalte werden vom *German Testing Board* [17] überwacht. In diesem Gremium sind Personen von Trainingsanbietern, Testexperten aus Industrie und Beratungsunternehmen sowie Vertreter der GI-Fachgruppe TAV unter dem Dach des ASQF organisiert, um gemeinsam die Lehrpläne und das Prüfungswesen weiterzuentwickeln. Das *Board* ist ehrenamtlich tätig. Es überprüft als unabhängige Instanz die angebotenen Kurse nach definierten Kriterien und spricht eine Akkreditierung der Kursanbieter aus. Es sorgt für die regelmäßige Durchführung von Prüfungen, bei denen die Kursteilnehmer nach bestandener Prüfung einen anerkannten Qualifikationsnachweis erhalten.

3 Kursinhalte und Teilnehmerzahlen

Um einen Eindruck davon zu vermitteln, welche Inhalte die Kurse behandeln, werden hier der Grund- und der Aufbaukurs kurz vorgestellt. Eine ausführliche Darstellung des Inhalts des Grundkurs ist in [1] zu finden, da das Buch passend zum Inhalt des Lehrplans aufgebaut ist. Ergänzend werden aktuelle Zahlen über den Stand der Akkreditierung und über die Anzahl der bisherigen Kursteilnehmer genannt.

Foundation Level

Der viertägige Kurs zum Erwerb des *Foundation Certificat*e gliedert sich in die folgenden sechs Themen:

- Grundlagen des Software-Testens
- Testen während des Lebenszyklus
- Dynamischer Test
- Statischer Test
- Test-Management
- Testunterstützungswerkzeuge

Für jedes Thema ist der zeitliche Aufwand und die einzelnen Punkte, die zu behandeln sind, im Lehrplan vermerkt [18]. So gliedert sich der Kursteil Test-Management, für den insgesamt 100 Minuten veranschlagt werden, in folgende Teile: Organisation, Konfigurationsmanagement, Testaufwand, Testüberwachung und -steuerung, Fehlermeldungsmanagement und einzusetzende Standards für den Test. Zu bedenken ist, dass es sich um den Grundlagenkurs handelt und keine vertiefte Darstellung der einzelnen Themen erfolgt. Die beiden Teile, die ausführlich

behandelt werden, sind das Testen während des Lebenszyklus (240 Minuten) und die Grundlagen des Software-Testens (200 Minuten).

Practitioner Level

Der insgesamt neuntägige Kurs zum Erwerb des *Practitioner Certificate* enthält 30-70% praktische Übungen und gliedert sich in die folgenden zehn Themen:

- Einführung (1,5 h)[3]
- Test-Prozess (3,5 h)
- Test-Management (6,5 h)
- Test und Risiko (4 h)
- Test-Techniken (20 h)
- Reviews (7 h)
- Incident Management (1,5 h)
- Test-Prozess-Verbesserung (3 h)
- Test-Werkzeuge (6 h)
- Führungsstil (3 h)

Für jedes Kapitel sind auch hier, neben dem zeitlichen Aufwand, die einzelnen Themen im Lehrplan vermerkt [13]. Beispielsweise sind im siebenstündigen Kursteil Reviews folgende Themen zu behandeln: Einführung in die grundlegenden Prinzipien von Reviews, informelle Reviews und Walkthroughs, Technische Reviews und Inspektionen, jeweils mit den entsprechenden Übungsanteilen.

Voraussetzungen für die Zulassung zur Prüfung ist das Zertifikat *Foundation Level* und eine 18-monatige einschlägige Berufserfahrung oder der Besuch der Kurse für den Erwerb des *Practitioner Levels*.

Das Kursprogramm für das *Practitioner Diploma* ist noch in Bearbeitung.

Kursanbieter und Teilnehmerzahlen

Seit 1998 wurden von ISEB insgesamt 23 Ausbildungseinrichtungen für den Kurs zum Erwerb des *Foundation Levels* akkreditiert. Bis Mitte 2002 haben über 8000 Personen an den englischsprachigen Kursen teilgenommen. Derzeit legen jeden Monat ca. 500 Personen die Prüfung zum *Foundation Certificate* ab. „Das Programm hat den Status des Software-Testens und die Anerkennung, die man Software-Testern entgegenbringt, deutlich gehoben. Auf dem britischen Arbeitsmarkt wird man inzwischen ohne das Zertifikat in der Regel gar nicht erst zu einem Vorstellungsgespräch als Software-Tester eingeladen" (aus dem Geleitwort von Dorothy Graham [1, S. xii]).

Deutschsprachige Kurse werden seit Herbst 2001 angeboten. Bis zum Jahresende 2002 haben etwa 200 Personen an den Prüfungen mit Erfolg teilgenommen. Die Erfolgsquote bei den Prüfungen liegt bei ca. 80%. Das beste Prüfungsergebnis er-

[3] In Klammern ist der zu veranschlagende Stundenaufwand angegeben.

zielte bisher eine Person, welche die Prüfung ohne vorherige Teilnahme am Kurs abgelegt hat.

4 Fragen an uns Lehrende an den Hochschulen

Folgende Fragen und Positionen sollten wir überdenken und diskutieren:

- Ist es erforderlich, solche Lehrinhalte aufzugreifen und in den Unterricht an den Hochschulen einfließen zu lassen?
- Soll an allen Hochschulen dasselbe gelehrt werden? Geht die Vielfalt des Lehrangebots dann verloren? Der Hochschullehrer bestimmt die Lehrinhalte, er trägt dafür die Verantwortung.
- Kann bei detailliert definiertem Lehrplan noch schnell genug auf aktuelle Themen reagiert werden und können die Themen in den Unterricht einfließen? Eine Festschreibung auf konkrete Lehrinhalte ist eine zu starke Einschränkung. Ganz abgesehen davon, dass es vermutlich keine Einigung auf konkrete Lehrinhalte geben wird. Es gibt zu viele unterschiedliche Meinungen.
- Sind Kursanbieter wirklich an einer fundierten Ausbildung interessiert oder wollen sie doch nur Geld verdienen?
- Besteht überhaupt eine direkte Konkurrenz zwischen Hochschulen und kommerziellen Weiterbildungseinrichtungen? Ein nebeneinander ist sinnvoll, beide haben ganz unterschiedliche Ausrichtungen und sprechen unterschiedliche Personenkreise an.
- Warum sollen die Lehrenden an den Hochschulen aktiv werden? Es gibt derzeit eine Zertifizierungs-Manie, man sollte gelassen abwarten, bis sie vorüber ist.
- Kann nur durch kommerzielle Weiterbildungsangebote eine Konkurrenzfähigkeit der einzelnen Firmen gewährleistet werden, da die Ausbildung im IT-Bereich so schnell veraltet?
- Müssen sich die Hochschulen viel massiver im Weiterbildungsmarkt positionieren? Nur so kann eine fundierte und fachlich hochwertige Weiterbildung garantiert werden.

Meiner Meinung nach sollten wir in der Informatik-Ausbildung dahin kommen, dass ein Grundwissen von allen Informatikern beherrscht wird. Bei diesem Grundwissen sollten auch nur geringe Unterschiede zwischen den Fachhochschulen und den Universitäten vorhanden sein. Darauf aufbauend sind dann Schwerpunktsetzungen vorzunehmen. Diese können je nach Hochschulart und je nach Ausrichtung der Hochschule bzw. des Fachbereichs unterschiedlich sein. Ein Diplom-Informatiker sollte über ein Grundlagenwissen im Bereich Softwaretest verfügen, unabhängig davon, ob er an einer Universität oder an einer Fachhochschule studiert hat. Auf diesem Wissen aufbauend können dann unterschiedliche Vertiefungen im Hauptstudium vorgenommen werden. Beispielsweise kann dies an den Universitäten im Bereich

der Verifikation und an den Fachhochschulen im Bereich der Testautomatisierung erfolgen.

Schwierig wird die Definition des gemeinsamen Grundwissens, dass jeder Informatiker beherrschen muss. Möglicherweise sind solche Initiativen, wie die beschriebene im Bereich Softwaretest, hilfreich und sollten Berücksichtigung finden, sofern es um Grundlagen geht. Eine direkte Ausrichtung an den aktuellen Bedürfnissen der Praxis halte ich für verkehrt.

David Parnas hat mit seiner Äußerung „*Our educational institutions have failed ...*“ nach meiner Meinung nur bedingt Recht. Die Informatik-Ausbildung an den Hochschulen in Deutschland hat einen guten Ruf und wird von den meisten Firmen als im Wesentlichen adäquat anerkannt. Die Hochschulabsolventen werden nach Aufnahme ihrer Berufstätigkeit durch gezielte Weiterbildungsmaßnahmen in den firmenspezifischen Methoden und Werkzeugen geschult. Das Grundlagenwissen sollten sie bereits mitbringen.

Ein Beispiel für ein Ausbildungsprogramm für Quereinsteiger ist *Sidestep* [19]. Aus Mangel an qualifiziertem Personal hat eine Firma in Kooperation mit einer Universität ein sechsmonatiges Programm zusammengestellt. Es wird das benötigte Grundlagenwissen und die Kenntnisse vermittelt, die ganz gezielt auf die Bedürfnisse der Firma ausgerichtet sind. Nur große Firmen können sich solch eine passend zugeschnittene Ausbildung leisten.

In einer persönlichen Diskussion mit David Parnas äußerte er die Ansicht, dass wir nur das lehren sollten, was die letzten 20 Jahre Gültigkeit hatte und auch noch für die nächsten 20 Jahre bestehen wird. Die Zeitspannen halte ich in der Informatik für zu lang, prinzipiell sollten wir aber unseren Studierenden nur das vermitteln, was über eine Tagesaktualität hinaus Bestand hat.

Ich wünsche mir für die Tagung eine anregende Diskussion und für die Leser des Tagungsbands eine Motivation zum Nachdenken über die eigene Position und vielleicht ja auch ein Anstoß zur Diskussion mit den Kollegen.

Literatur und WWW-Seiten[4]

1. Spillner, A.; Linz, T.: Basiswissen Softwaretest. dpunkt, Heidelberg, 2002
2. Parnas, D.L.: University Programmes in Software Development (Abstract). In [3], 9
3. Lichter, H.; Glinz, M. (Hrsg.): Software Engineering im Unterricht der Hochschulen, SEUH 7 – Zürich 2001. dpunkt, Heidelberg, 2001
4. Liste der Empfehlungen zur Aus- und Weiterbildung der Gesellschaft für Informatik http://www.gi-ev.de/informatik/publikationen/empfehlungen.shtml
5. Ausbildung von Diplom-Informatikern an wissenschaftlichen Hochschulen – Empfehlungen der Gesellschaft für Informatik. Informatik-Spektrum, 8/3, Juni 1985, 164-165

[4] Die Gültigkeit der angegebenen URLs wurde mit Drucklegung des Beitrags überprüft. Eine Garantie für deren Gültigkeit über dieses Datum (Dez. 2002) hinaus kann nicht übernommen werden.

6. Ergänzende Empfehlungen der Gesellschaft für Informatik: Lehrinhalte und Veranstaltungsformen im Informatikstudium. Informatik-Spektrum, 20/5, Oktober 1997, 302-306
7. Guide to the Software Engineering Body of Knowledge, IEEE – Trail Version 1.00. May 2001 http://www.swebok.org/
8. Zertifizierer Comptia sieht sich als Ergänzung zur Berufsausbildung. Computer Zeitung, 34, 19. August 2002, 22
9. Informationen zum *Certified Tester* http://www.certified-tester.de
10. Informationen zum *Information Systems Examinations Board* http://www.iseb.org.uk
11. Informationen zur *British Computer Society* http://www1.bcs.org.uk
12. Lehrplan *Foundation Level* (englische Version) http://www.iseb.org.uk ⇒Qualifikation Areas ⇒Software Testing ⇒ISEB Foundation Syllabus V2.0 – 25 February 1999
13. Lehrplan *Practitioner Level* (englische Version) http://www.iseb.org.uk ⇒Qualifikation Areas ⇒Software Testing ⇒ISEB Practitioner Syllabus V1.1 – 04 September 2001
14. Informationen zum *International Software Testing Qualification Board* http://www.istqb.org
15. Informationen zum Arbeitskreis Softwarequalität Franken e. V. http://www.asqf.de/deu/index.php
16. Informationen zur GI-Fachgruppe TAV (Test, Analyse und Verifikation von Software) http://www.fbe.hs-bremen.de/spillner/gi.htm
17. Informationen zum *German Testing Board* http://www.certified-tester.de/board
18. Lehrplan *Foundation Level* (deutsche Version) http://www.certified-tester.de/lehrplan.php
19. Siedersleben, J.: Sidestep: Die Informatik-Initiative von sd&m. In [3], 39-44

Die Einführung objektorientierter Gestaltungsprinzipien anhand von Rollenspielen

Stefan Dißmann
Universität Dortmund, Lehrstuhl Software-Technologie, 44221 Dortmund
stefan.dissmann@udo.edu

Kurzfassung

Dieser Beitrag beschäftigt sich mit der Vermittlung grundlegender objektorientierter Gestaltungsprinzipien. Mit dem Rollenspiel als Animation eines objektorientierten Systems wird hier eine Lehrform vorgeschlagen, die im Vergleich zu den heute vielfach propagierten Multimedia- und E-Learning-Systemen nur wenig Aufwand in der Vorbereitung erfordert und zugleich ein effektives handlungsorientiertes Lernen ermöglicht.

Der Beitrag motiviert den Einsatz von Rollenspielen in der Ausbildung und geht kurz auf den in der Softwaretechnik üblichen Einsatz von Rollenspielen als Maßnahme der Qualitätssicherung ein. Anschließend werden die Zielsetzungen und Rahmenbedingungen für den Einsatz von Rollenspielen in der Ausbildung formuliert und an einem Beispiel ihre Umsetzung demonstriert. Das Beispiel bildet dann den Ausgangspunkt für eine kritische Analyse des vorgeschlagenen Ausbildungskonzeptes und für einen Ausblick auf weitere Einsatzbereiche.

1 Ausgangsproblematik

Die Grundlage für jede erfolgreiche Wissensvermittlung ist die Einbeziehung des Lernenden in den Lernprozess. Nur wenn es dem Lehrenden gelingt, den Lernenden zum Mitdenken und Mitarbeiten zu motivieren, werden die vermittelten Inhalte aufgenommen, verstanden und als Wissen verfestigt. Den Ausgangspunkt für das Verstehen neuer Inhalte bilden immer schon bekannte Fakten und gemachte Erfahrungen. Die Vermittlung von neuem Wissen bereitet daher immer dann Probleme, wenn der Lernende noch nicht über Wissen und Erfahrungen verfügt, an die im Lernprozess angeknüpft werden kann.

Diese Situation kommt bei der Vermittlung der grundlegenden objektorientierten Gestaltungsprinzipien zu Beginn der Ausbildung in Softwaretechnik häufig vor: Die Berücksichtigung dieser Gestaltungsprinzipien während der Softwareentwicklung führt letztlich zu charakteristischen Abläufen während der Produktausführung, welche für den Lernenden nicht durch die Betrachtung des ausgeführten Produktes

erfahrbar sind. Technische Beschreibungen des Produktes (Quelltexte, visualisierte Systemarchitekturen) beinhalten die Darstellung dieser internen Abläufe, müssen aber für ihr Verstehen vom Leser interpretiert werden. Unerfahrenen Lesern gelingt diese Interpretation nur selten.

Unter den grundlegenden Gestaltungsprinzipien sollen hier die Vorgaben verstanden werden, die den Aufbau eines objektorientierten Systems als Menge von gemeinsam über den Austausch von Nachrichten handelnden Objekten bewirken. Objekte müssen dabei die bekannten Anforderungen an die Kapselung, Lokalität, Geheimhaltung, konzeptionelle Vollständigkeit und autonomes Handeln erfüllen. Nicht beachtet werden hier zunächst solche Prinzipien, die für die Umsetzung von Spezialisierung, Generalisierung und Vererbung Bedeutung besitzen.

Der Lehrende, der in die objektorientierte Softwareentwicklung einführen will, steht damit vor einem Dilemma: Um die Wirkungsweise objektorientierter Gestaltungsprinzipien zu verstehen, benötigt der Lernende Einblick in ein funktionierendes objektorientiertes Produkt, diesen Einblick kann er aber nur durch das Verstehen technischer Beschreibungen erlangen, was wiederum ohne Wissen über die Prinzipien nicht möglich ist. In der Realität der akademischen Ausbildung wird dieses Dilemma meist durch eine vorgeschaltete Beschäftigung mit einer objektorientierten Programmiersprache gelöst. Die Lernenden werden zunächst noch »unwissend« zu objektorientierten Programmstrukturen hingeführt, die dann erst später explorativ als solche herausgearbeitet und expliziert werden. Diese Lösung benötigt viel Ausbildungszeit und ist offensichtlich nur dann praktikabel, wenn die Programmierausbildung der Ausbildung in der Softwaretechnik vorangeht.

2 Rollenspiele als Mittel der Veranschaulichung

Muss oder will der Lehrende auf eine vorgeschaltete Programmierausbildung verzichten, so kann er nicht auf eine exemplarische Implementierung eines objektorientierten Systems zurückgreifen. Es kommt dann als Beispiel nur der Einsatz einer andersgearteten Veranschaulichung eines solchen Systems infrage, einer Animation, die die systeminternen Abläufe verdeutlicht. Eine solche, speziell für den Zweck der Ausbildung geschaffene Animation kann in ihrem Abstraktionsniveau den Vorkenntnissen des Lernenden angepasst werden und so die Wirkungsweise und Bedeutung der verschiedenen objektorientierten Gestaltungsprinzipien für die technischen Abläufe des Endproduktes angemessen veranschaulichen.

In der Zeit von Multimedia und E-Learning ist der lehrende Informatiker leicht versucht, die Realisierung einer solchen Animation durch eine simulierende und visualisierende Software vorzunehmen. Software-Animationen von Softwaresystemen sind jedoch lediglich künstliche Veranschaulichungen von Programmstrukturen und -verhalten, die wie die statischen technischen Beschreibungen beim Lernenden nicht zwangsläufig zu Assoziationen mit bekannten Erfahrungen führen. Für einen erfolgreichen Einsatz muss eine Software-Animation daher sorgfältig

geplant werden und aufwändig erstellt werden. Der Lernende bleibt trotzdem nur Betrachter des Animationsablaufes, da ihm für steuernde Eingriffe das Verständnis der systeminternen Abläufe fehlt. Wird die Ausbildung in einer Gruppe durchgeführt, kann eine solche Software-Animation durch ihren vorgegebenen Ablauf die Kommunikation innerhalb des Lernprozesses nur wenig anregen.

Allerdings bietet sich neben der Animation durch spezielle Software eine alternative Form der Veranschaulichung dynamischer Vorgänge an, das Rollenspiel, d.h. das von Menschen animierte Systemverhalten als Theaterspiel. Menschen sind im Sinne der Informatik autonom und nicht deterministisch handelnde, sehr gut anpassbare Systemkomponenten, die in der Regel immer etwas fehlerhaft arbeiten. Die menschliche Flexibilität ermöglicht es jedoch, in vielen Fällen diese Fehler durch intelligente »Ausnahmebehandlungen« der anderen Komponenten zu korrigieren.

Wird also bei der Veranschaulichung der internen Abläufe eines objektorientierten Systems auf menschliche Akteure zurückgegriffen, so kann mit vergleichsweise wenig Aufwand eine ansprechende Animation geschaffen werden. Je weniger dabei Wert auf einen vorgegebenen Ablauf gelegt wird, desto einfacher kann ein Erfolg erzielt werden. Gelingt es, die menschliche Animation eines objektorientierten Systems als Rollenspiel so einfach zu gestalten, dass keine Probe notwendig ist und kann daher auf beliebige Akteure zurückgegriffen werden, so bietet es sich an, die Rollen mit den Lernenden selbst zu besetzten. Der Lernende wird dann vom Zuhörer und Betrachter zu einem integralen Bestandteil des Systems, aus einer Präsentation wird ein handlungsorientiertes Lernprojekt. Handlungsorientiertes Lernen [1, 2] stellt eine sehr effektive Lernform dar, die beim Lernenden aktive Mitarbeit im Lernprozess, Aufmerksamkeit über einen langen Zeitraum und eine tiefe Verankerung der gemachten Erfahrungen als Wissen bewirkt. Handlungsorientierte Gruppenarbeit wie ein Rollenspiel mit verschiedenen Mitspielern initiiert unmittelbar gruppendynamische Lernprozesse und kooperatives Lernen.

Die vorangehende Argumentation zeigt, dass die menschliche Animation eines objektorientierten Systems als Rollenspiel Vorteile für den Lernerfolg verspricht und zugleich – bei geschicktem Einsatz der menschlichen Akteure – mit geringem Aufwand gestaltet werden kann. Bevor die Randbedingungen herausgearbeitet werden, die ein erfolgreiches Rollenspiel sicherstellen, soll kurz auf den traditionellen Einsatz des Rollenspiels in der Softwareentwicklung eingegangen werden.

3 Rollenspiele in der Software-Entwicklung

Das »Durchspielen« des internen Ablaufs eines Produktes zählt zu den Standardaufgaben von Softwareentwicklern und ist in nahezu allen strukturierten oder objektorientierten Vorgehensmodellen implizit oder explizit vorgesehen. Ein Beispiel hierfür ist der zu den klassischen Techniken der Softwareentwicklung zählende »walkthrough« im Rahmen der Qualitätssicherung. Im Umfeld der objektorientierten Entwicklung wird insbesondere beim Einsatz von CRC-Karten während

der Anforderungsanalyse die Simulation des durch die Karten spezifizierten Systemmodells als gemeinsames Rollenspiel von Fachexperten und Entwicklern als wesentlicher Entwicklungsschritt [3, 4, 5] hervorgehoben.

Rollenspiele sind also eine etablierte Technik, die in objektorientierten Vorgehensmodellen [6, 7] zur Evaluation der entwickelten objektorientierten Lösungen eingesetzt wird. Mitspieler im Rollenspiel sind dabei erfahrene Entwickler, die die objektorientierten Gestaltungsprinzipien beherrschen und durch das gemeinsame Spiel u.a. deren Einhaltung am gegebenen Produkt prüfen sollen.

Der vorliegende Artikel propagiert demgegenüber das Rollenspiel als Hilfsmittel in einer frühen Phase der Ausbildung, um Lernenden die grundsätzliche Wirkungsweise objektorientierter Gestaltungsprinzipien nahe zu bringen und ihnen den Erwerb von objektorientierten Basiserfahrungen ohne Programmierkenntnisse zu ermöglichen. Als Nebeneffekt bereitet diese Art der Ausbildung die Lernenden natürlicherweise auch auf den traditionellen Einsatzbereich von Rollenspielen in Anforderungsanalyse und Qualitätssicherung vor.

4 Rahmenbedingungen für ein geeignetes Beispielsystem

Das in den ersten Abschnitten entworfene Ausbildungsszenario geht davon aus, dass die betroffenen Lernenden über keinerlei Vorkenntnisse im Bereich objektorientierter Entwicklung verfügen. Trotzdem sollen mehrere Lernende gemeinsam in der Lage sein, als Akteure eines Rollenspiels ein objektorientiertes System so zu animieren, dass sie aus ihrem Handeln neues Wissen und neue Erkenntnisse ableiten können. Dabei sollen sie die Bedeutung grundlegender objektorientierter Gestaltungsprinzipien ohne Rückgriff auf Wissen aus diesem Bereich erkennen können.

Das durch das Rollenspiel animierte Beispielsystem muss daher verschiedene, sehr unterschiedliche und sich zum Teil scheinbar konträr gegenüberstehende Rahmenbedingungen erfüllen. Es wird ein objektorientiertes System benötigt,

- das eine leicht verständliche Aufgabe erfüllt, um zusätzliche Probleme der Lernenden beim Erfassen des Problembereichs zu vermeiden,
- dessen Struktur und Arbeitsweise ohne die Begriffe der Objektorientierung beschrieben werden können, um den Lernenden Handlungsanweisungen für das zugehörige Rollenspiel geben zu können,
- dessen Details von den Lernenden erkannt werden können, um die Bedeutung einzelner Aspekte untersuchen zu können,
- dessen Struktur und Arbeitsweise einfach gezielten Modifikationen unterzogen werden können, um daraus Rückschlüsse auf die Bedeutung und Wirksamkeit der Gestaltungsprinzipien ableiten zu können, und
- dessen Struktur und Arbeitsweise leicht geändert werden können, um Ideen aus der Diskussion der Lernenden schnell animieren zu können.

Wesentliches Hilfsmittel für die Umsetzung dieser Rahmenbedingungen ist der gezielte Einsatz der natürlichen Fähigkeiten und der Intelligenz der Lernenden.

- Die Lebenserfahrung des Lernenden ermöglicht es, eine allgemein bekannte Aufgabenstellung für das Beispielsystem auszuwählen.
- Die Fähigkeit zum eigenständigen Schlussfolgern ermöglicht es dem Lernenden, unscharfe Handlungsanweisungen aufgrund des Kontextes der Aufgabenstellung zu vervollständigen und selbstständig zu handeln.
- Der Lernende kann sein Handeln zusätzlichen Anweisungen des Lehrenden anpassen und damit spontan ändern.

Die Animation eines Beispielsystems durch ein Rollenspiel im Zuge des handlungsorientierten Lernens erfolgt also im Wesentlichen dadurch, dass der Lernende sich im Sinne einer objektorientierten Denkweise selbst spielt: Der Lernende ist ein intelligentes, hochflexibles und autonomes Objekt, das genau ein Objekt des Beispielsystems simuliert. Einfache Anweisungen reichen aus, um ein solches menschliches Objekt autonom als Systemobjekt arbeiten zu lassen. Das gesamte System wird aus allen Lernenden gebildet, die Nachrichten austauschen. Sie sind dadurch unmittelbar in die Abläufe integriert und lernen die objektorientierten Arbeitsweisen zugleich in ihrer Gesamtwirkung und aus der Sicht eines einzelnen Objekts heraus kennen.

Begriffe der Objektorientierung werden für das Drehbuch des Rollenspiels zunächst nicht benötigt, es reicht aus, stattdessen ein System aus menschlichen Spezialisten zu beschreiben, die nach wenigen vorgegebenen Regeln zusammenarbeiten. Für jeden Rollenspieler, also Spezialisten des Systems, gelten die folgenden vier Grundregeln, die auf natürliche Art das Einhalten der Prinzipien Kapselung, Lokalität und Geheimhaltung erzwingen.

- Er hält sich an die ihm vorgegebenen Handlungsanweisungen.
- Er fragt andere Spezialisten, sobald er an die Grenzen seiner Handlungsmöglichkeiten stößt.
- Er hilft anderen Spezialisten, sofern ihn diese um Hilfe bitten.
- Er sichert seinen Status als Spezialist dadurch, dass er die ihm zugeordneten Daten und Handlungsanweisungen streng vertraulich behandelt.

Die Bühne für ein solches Rollenspiel ist einfach gestaltet. Jeder Lernende übernimmt die Rolle eines Spezialisten und erhält einen globalen Namen, der auf seinen Aufgabenbereich hinweist. Zudem erhält er einen entsprechenden Satz vertraulicher Daten – seine Attributwerte – und einen Satz vertraulicher Handlungsanweisungen, anhand derer er agieren darf. Diese Anweisungen grenzen grob die Menge der für den Spezialisten möglichen Methoden ein, ohne diese jedoch präzise aufzuführen.

Das Rollenspiel weist also zwei grobe Vereinfachungen gegenüber technisch spezifizierten objektorientierten Systemen auf, da auf die Angabe expliziter Benutzungsbeziehungen zwischen den Spezialisten und auf die exakte Definition der Leistungsfähigkeit der einzelnen Spezialisten verzichtet wird. Beide Vereinfachun-

gen müssen vorgenommen werden, da das Verstehen exakter Spezifikationen dieser Aspekte bereits objektorientierte Grundkenntnisse erfordern würde. Das Rollenspiel bleibt trotzdem spielbar, da einerseits die Fähigkeit der Rollenspieler zu verbaler Kommunikation gerade ohne die Vorgabe von Methoden und Benutzungsbeziehungen beliebig ausgeschöpft werden kann und andererseits mit dem Lehrenden als Spielleiter eine übergeordnete Instanz bereitsteht, die die Rollenspieler bei ihrem Handeln unterstützen kann.

Der Nachrichtenfluss innerhalb des Systems erfolgt durch den Austausch schriftlicher Kurznachrichten von einem Spezialisten zum anderen, die jeweils für die Frage und die zugehörige Antwort verwendet werden. Dadurch ist die Abfolge der Nachrichten deutlich sichtbar, sie kann leicht gesteuert, festgehalten und in einer sich anschließenden Diskussion nachvollzogen werden. Gleichzeitig lassen sich mit einfachen Hilfsmitteln verschiedene Ausführungsszenarien realisieren: Wird für das gesamte System genau ein Stift bereitgestellt, der dann mit der aktuellen Kurznachricht von einem Spezialisten zum nächsten weitergereicht wird, so kann immer nur einer der Spezialisten handeln: Es wird ein sequentiell arbeitendes objektorientiertes System realisiert. Werden mehrere Stifte eingesetzt, so ist das nebenläufige Arbeiten von Spezialisten – mit allen dabei auftretenden Schwierigkeiten – möglich.

Als vorbereitende Maßnahmen für das Rollenspiel müssen die vier Grundregeln erklärt und verbindlich gemacht werden, die Aufgabe des zu spielenden Systems muss knapp umrissen werden und die Lernenden müssen ihre Rollen zugeteilt bekommen. Dann muss hinreichend lange gespielt werden, um jedem Lernenden die Gelegenheit zu geben, eigene Erfahrungen zugleich als aktiver Mitspieler und als passiver Zuschauer zu sammeln. Abschließend müssen die Aufarbeitung der Erfahrungen und das Explizieren der objektorientierten Gestaltungsprinzipien erfolgen.

5 Ein Rollenspiel: Ausschnitt aus einem Banksystem

Um die Konzentration der Spieler auf die Abläufe des Systems zu lenken, bietet es sich an, ein objektorientiertes System nachzuspielen, dessen Leistung allen beteiligten Lernenden unmittelbar und intuitiv verständlich ist. Der Autor setzt hierzu ein Beispiel aus dem auch in Lehrbüchern [5, 6] beliebten Bankbereich ein, einen Ausschnitt aus dem Kontext der Benutzung eines Bankautomaten. Das sich ergebende Szenario wird hier kurz geschildert, um daran anknüpfend die Schulungstechnik diskutieren zu können.

Ein Bankautomat stellt zwei Funktionen bereit, das Abfragen des Kontostands und das Abheben von Bargeld. Für die Ausführung dieser beiden Funktionen wird ein Ausschnitt aus dem zugehörigen Banksystem benötigt, der die Überprüfung der Eingaben, das Ermitteln eines Kontos und dessen Kontostands und die Berechnung von Zinsen umfasst. Der Bankautomat muss gegebenenfalls Banknoten in Höhe des angeforderten Betrags ausgeben können.

Im Rollenspiel sind entsprechend Rollen für die Spezialisten *Kundenkontakt*, *Kontenverwaltung*, *Sollzinsberechnung*, *Banknotenausgabe* und für mehrere Spezialisten *Konto* vorgesehen. Hinzu kommen mehrere Kunden, die den Automaten nutzen. Durch die Zahl der Kunden, die direkt auch die Zahl der notwendigen Konten bestimmt, kann eine flexible Zahl von Rollen für die Mitspieler geschaffen werden, so dass ihre Zahl der der teilnehmenden Lernenden angepasst werden kann.

Die folgenden Abbildungen zeigen die Anweisungen und Daten für die Spezialisten *Kontenverwaltung* und *Sollzinsberechnung*. Die Vorgaben für die anderen Spezialisten sind ähnlich kurz gehalten.

Spezialist für Kontenverwaltung

Aufgaben:

Verwalten der Zuordnung von Kontonummer, Kartennummer, PIN und Kontobezeichnung.
Diese Zuordnung darf nicht verändert werden.
Die Zuordnung ist sicherheitskritisch und darf nicht herausgegeben werden.

Kontonummer	Kartennummer	PIN	Kontobezeichnung
2341	1432	4567	6
5109	9015	3128	4
1344	4413	6891	3
...	...	...	...

Spezialist für Sollzinsberechnung

Aufgaben:

Durchführen der Zinsberechnung!
Die Formel ist das exklusive Wissen dieses Spezialisten und darf nicht weitergegeben werden!

Kreditzinsen berechnen sich nach folgender Formel:
Anzahl der Tage * Zinssatz aus Tabelle * Betrag / 360

Grenzbetrag	Zinssatz
Fehlbetrag kleiner als Euro 10000	0.10
Fehlbetrag größer als Euro 10000	zusätzlich 0.02 für den Fehlbetrag über DM 10000

Zur Vorbereitung erhält jeder Mitspieler die vertraulichen Handlungsanweisungen seiner Rolle mit den zugehörigen Daten. Das Spiel beginnt mit einem Arbeitsauftrag eines Kunden an das System, indem dieser dem Spezialisten für den *Kundenkontakt* die erforderlichen Angaben liefert. Da dieser mit den ihm zur Verfügung stehenden Anweisungen lediglich die Vollständigkeit der Angaben prüfen kann, muss er sich

zwangsläufig hilfesuchend mit einer Nachricht an einen anderen Spezialisten wenden und der Nachrichtenaustausch und damit die Ausführung des Systems beginnen. Da die Leistungen der Spezialisten nicht formal definiert sind, treten gelegentlich Anfragen auf, die nicht beantwortet werden können. Da auch die Antworten auf Nachrichten nicht vordefiniert sind, kann der angesprochene Spezialist hierauf entsprechend flexibel reagieren.

Der so entstehende Nachrichtenverkehr zwischen den Spezialisten des Systems wird den teilnehmenden Lernenden grafisch visualisiert. Werden die beteiligten Spezialisten als Knoten dargestellt und wird für jede ausgetauschte Nachricht eine verbindende Kante eingetragen, so ergibt sich schrittweise ein Kollaborationsdiagramm [8]. Diese Darstellungsform ist in der erlebten Spielsituation unmittelbar verständlich und benötigt keine Erläuterungen. Alternativ lässt sich aber auch eine Darstellung als Sequenzdiagramm [8] einsetzen, wenn dieses während des Spiels schrittweise in Übereinstimmung mit dem erlebten Nachrichtenfluss aufgebaut wird. Die Bedeutung der Darstellung ergibt sich aus dem Erstellungsprozess.

6 Aus dem Rollenspiel ableitbare Erkenntnisse

Das Rollenspiel zeigt bei seiner Durchführung sehr schnell, dass für das Zusammenwirken der Spezialisten keine detaillierten Kenntnisse fremder Handlungsanweisungen notwendig sind. Die Prinzipien der Lokalität, der Kapselung und der Geheimhaltung der Interna von Spezialisten werden durch die Regeln erzwungen und können anschließend leicht durch den Lehrenden expliziert werden.

Für das vertiefte Verständnis der Wirkungsweise der objektorientierten Gestaltungsprinzipien ist es wichtig, dass die Lernenden nach einem ersten spielerischen Umgang mit dem von ihnen selbst gestalteten System zunehmend dessen Einzelaspekte erkennen und so die Vorzüge der objektorientierten Struktur herausarbeiten.

Das Rollenspiel beginnt daher immer mit einfachen Aufgabenstellungen an das System, dem Abfragen des Kontostands für ein existierendes Konto und dem Abheben eines zulässigen Barbetrags. Sind den Lernenden hierdurch die Bearbeitungsprinzipien vertraut geworden, sollten sie Ausnahmesituationen erfahren, zum Beispiel den Versuch eines Zugriffs mit falscher Geheimzahl, den Versuch des Abhebens eines nicht gedeckten oder eines nicht mit den verfügbaren Banknoten erfüllbaren Betrags. Diese Ausnahmesituationen erfordern einen Nachrichtenfluss, der den Lernenden verdeutlicht, dass eine geeignete Kommunikation in objektorientierten Systemen sorgfältig geplant werden muss. Beispielsweise ergibt sich häufig als »natürliche« Reihenfolge der Nachrichten im Banksystem zunächst die Kontrolle des verfügbaren Kredits, dann wird die Transaktion gebucht und die Ausgabe der Banknoten angestoßen. Ist der Betrag nun nicht auszahlbar, weil die Banknoten nicht passend gestückelt werden können, so bleibt die Aufgabe zwar lösbar, erfordert dann aber ein umständliches Zurücksetzen der Buchung.

Durch die verschiedenen Spielsituationen wird das System den Lernenden zunehmend vertraut, sie akzeptieren die Funktionsfähigkeit eines nach objektorientierten Prinzipien agierenden Systems. In einem folgenden Schritt müssen ihnen nun die besonderen Vorzüge eines solchen Systems nahe gebracht werden. Hierzu wird das nun schon vertraute System gezielten Änderungen unterzogen. In dem bekannten System können beispielsweise der Aufbau der Tabelle der *Kontoverwaltung* oder die Berechnungsformel der *Sollzinsberechnung* geändert werden. Die Lernenden erkennen, dass lokale Änderungen an der Datenhaltung oder an den Handlungsanweisungen keine Auswirkungen auf die anderen Spezialisten und insbesondere auch nicht auf die in den ersten Experimenten etablierten Kommunikationsabläufe haben. Durch das Einführen weiterer Spezialisten, beispielsweise einen *Stückelungsberechner* als Ergänzung zur *Banknotenausgabe* oder eine zusätzliche *Habenzinsberechnung*, kann den Lernenden auch gezeigt werden, dass selbst das Hinzunehmen neuer Spezialisten nur überschaubare Änderungen des Nachrichtenflusses bewirkt und die Realisierung der meisten Spezialisten unverändert lässt. Abschließend können dann das nebenläufige Erstellen und Bearbeiten von Nachrichten und die dabei auftretenden Probleme demonstriert werden.

Insgesamt wird den Lernenden durch das Rollenspiel deutlich, dass ein System auf der Basis von Kommunikation über Nachrichten tragfähig ist. Gleichzeitig zeigt das Spiel auch die Schwächen nur informal und grob definierter Spezialisten: Die Unbestimmtheit der von einem Spezialisten erbrachten Leistungen erschwert das Festlegen von Abläufen innerhalb des Systems erheblich. Hier wird den Lernenden deutlich, welche Bedeutung die (semi-)formale Definition klarer Schnittstellen besitzt. Insgesamt sollten die Lernenden im Anschluss an das Rollenspiel erkannt haben, dass Objekte sowohl eine Abstraktion realer Gegebenheiten darstellen als auch eine solide Basis für ein flexibles technisches System bilden können.

Das Banksystem mit der Verwaltung von Konten bietet auch Ansatzpunkte, um weiterführende objektorientierte Konzepte zu motivieren. So führt die Untersuchung der verschiedenen im Beispielsystem auftretenden *Konto*-Objekte recht unmittelbar zum Begriff der Klasse. Betrachtet man verschiedene Kontenarten, also neben dem Girokonto auch noch das Sparkonto, so lässt sich auch das Konzept der Vererbung herleiten. Allerdings beschreiben die Rollen des Rollenspiels immer konkret handelnde Akteure, sind also im Sinne der Objektorientierung Objekte. Auch das durch das Rollenspiel gegebene Spiel ist immer nur das Ausführen eines Systems anhand einer konkreten Objektstruktur und veranschaulicht keine Typaspekte.

Das Rollenspiel könnte sicher um Rollen ergänzt werden, die Aufgaben der Klassen und des Laufzeitsystems übernehmen: Wird die Klasse selbst als Rolle aufgenommen, dann kann sie neue Instanzen durch die Herausgabe von Handlungsanweisungen an weitere Mitspieler erzeugen. Auch Vererbung im Sinne der Erweiterung einer Klasse könnte durch das »Zusammenkleben« der Anweisungen von Ober- und Unterklasse durch den die Unterklasse repräsentierenden Spieler verdeutlicht werden. Konzepte wie die Redefinition von Handlungsanweisungen in Unter-

klassen und die sich aus der Polymorphie ergebenden Probleme würden allerdings das Einführen von präzise abgegrenzten Methoden erfordern, da sonst die Redefinition nicht erkennbar wäre. Unter solchen Erweiterungen des Rollenspiels würde aber die Einfachheit der Spielidee leiden, das Spiel müsste um zahlreiche Regeln erweitert werden. Da der Autor befürchtet, dass dies die Lernenden doch stark überfordern und damit den gesamten Lernerfolg gefährden würde, hat er bisher auf entsprechende Versuche im Unterricht verzichtet.

7 Weitere Einsatzbereiche von Rollenspielen

Ausgehend von den positiven Erfahrungen mit den Rollenspielen bei der Einführung in die objektorientierte Denkweise hat der Autor menschliche Animationen durch die Lernenden auch in der Programmierausbildung erprobt. Einfache Datenstrukturen wie Liste und Baum lassen sich – wenn objektorientiert programmiert – recht gut durch Lernende veranschaulichen, die Elemente oder Knoten spielen. Auch die Arbeitsweise von Rekursion lässt sich mit Rollenspielen leicht deutlich machen: Der erste Rollenspieler erhält einen Stapel gleicher Arbeitsanweisungen, bearbeitet die erste und gibt im Falle des rekursiven Aufrufs alle übrigen Anweisungen an einen zweiten Rollenspieler weiter. Im Gegensatz zu der festen Zahl von Objekten im oben vorgestellten Banksystem lebt diese Animation gerade von der Hinzunahme weiterer Mitspieler während des Spiels.

Ein anderer denkbarer Einsatzbereich von Rollenspielen in der Softwaretechnik-Ausbildung ist das Erarbeiten von komplexen Systemarchitekturen. Zum Beispiel verbirgt sich das teilweise komplexe Verhalten von Entwurfsmustern [9] häufig hinter einer recht schlichten Klassenstruktur. Hier könnte es für fortgeschrittene Lernende hilfreich sein, das Verhalten durch ein Rollenspiel zu erarbeiten.

Die Weiterbildung ist ein zusätzlicher Bereich [10], in dem Rollenspiele erfolgreich eingesetzt werden können, um kompetente Anwender strukturierter Techniken und Programmiersprachen zunächst einmal von der technischen Umsetzbarkeit und Tragfähigkeit der objektorientierten Ideen zu überzeugen. Für diesen programmiererfahrenen Kreis von Lernenden können dann auch technische Beschreibungen der einzelnen Spezialisten ausgegeben werden.

8 Zusammenfassung

Rollenspiele bieten eine Möglichkeit, im Rahmen der Ausbildung technische Systeme durch die Lernenden selbst simulieren und analysieren zu lassen. Gerade die Funktionsweise objektorientierter Systeme lässt sich leicht durch ein Rollenspiel nachbilden und so auch Lernenden ohne jegliche Programmierkenntnisse demonstrieren. Dabei können die grundlegenden Prinzipien der objektorientierten Gestaltung deutlich gemacht werden. Ein ausschnittsweise vorgestelltes Beispielsystem, das der

Autor praktisch in Lehrveranstaltungen eingesetzt hat, zeigt die Möglichkeiten und auch die Grenzen des Lernens anhand von Rollenspielen.

Die Vorbereitung eines Rollenspiels erfordert im Vergleich zu technischen Animationen sehr wenig Aufwand. Allerdings stellt sich die Frage, ob die zu erwartende Verbesserung des Lernerfolgs den Einsatz eines Rollenspiels überhaupt rechtfertigt. Mit der Motivation des Spiels und der Aufarbeitung des gespielten Systems müssen für das Rollenspiel ein bis zwei zweistündige Übungseinheiten angesetzt werden. Auf den ersten Blick scheint dies ein hoher Aufwand für eine Einführung zu sein, die lediglich einige grundlegende Prinzipien der objektorientierten Denkweise nahe bringt. Die Erfahrungen des Autors mit diesem Einstieg zeigen jedoch, dass sich der Aufwand im Verlauf der weiteren Ausbildung rentiert, und zwar insbesondere in den folgenden drei Bereichen:

- Eine ungefähre Vorstellung von dem beabsichtigten »Ergebnis« objektorientierter Entwicklungsarbeit weckt das Interesse des Lernenden stärker als das – formal präzise – Einführen von Konzepten, deren Wirkungsweisen jedoch nur langsam deutlich werden.
- Ist der Lernende bereits am Anfang der Ausbildung mit dem Begriff des Objekts und der Funktionsweise des objektorientierten Systems vertraut, so wird damit ein Ziel definiert, auf das während der gesamten weiteren Ausbildung hingearbeitet werden kann.
- Das Rollenspiel verschafft Lernenden und Lehrendem gemeinsames Wissen über ein funktionstüchtiges, objektorientiertes System, das insbesondere immer dann zur Konstruktion weiterer Beispiele herangezogen werden kann, wenn Objektstrukturen oder Kommunikationsverbindungen diskutiert werden. Jeder Lehrende wird bei Verständnisschwierigkeiten und Fragen gerne an ein verstandenes Beispiel anknüpfen. Nach Einschätzung des Autors kann allein hierdurch der Aufwand für das Rollenspiel zurückgewonnen werden.

Literatur

1. Gudjons, H.: Handlungsorientiert lernen. Klinkhardt, 2001
2. Wöll, G.: Handeln: Lernen durch Erfahrung. Schneider, 1998
3. Wilkinson, N.M.: Using CRC Cards. SIGS Books, 1995
4. Ambler, S.W.: The Object Primer. SIGS Books, 1995
5. Wirfs-Brock, R. et al.: Designing Object-Oriented Software. Prentice Hall, 1990
6. Rumbaugh, J. et.al.: Object-Oriented Modeling and Design. Prentice Hall, 1991
7. Jacobson, I. et al.: The Unified Software Developement Process. Addison-Wesley 1999
8. Rumbaugh, J. et al.: Unified Modelling Language Reference Manual. Addison-Wesley 1999
9. Gamma, E. et al.: Design Pattern. Addison Wesley 1995
11. Dissmann, S.: Erfahrungen mit dem Einsatz von Rollenspielen zur Einführung in die objektorientierte Denkweise. Proceedings Net.ObjectDays 2001, 355-360

Konzeption und Analyse eines Softwarepraktikums im Grundstudium

Andreas Metzger
Fachbereich Informatik, Technische Universität Kaiserslautern
Postfach 3049, 67653 Kaiserslautern
metzger@informatik.uni-kl.de

Zusammenfassung

In diesem Artikel wird ein Softwarepraktikum im Grundstudium der Informatik an der Universität Kaiserslautern diskutiert, in welchem die systematische Erstellung einer komplexen Ampelsteuerung gefordert war.

Neben der Beschreibung der Aufgabenstellung werden die Ergebnisse des Softwarepraktikums und die gemachten Erfahrungen analysiert, wobei erläutert wird, wie diese Resultate in bestehende Verbesserungen des Praktikums einflossen und welche zukünftigen Veränderungen sinnvoll erscheinen.

1 Einleitung

Im Wintersemester 1998/99 fand an der Universität Kaiserslautern eine Modernisierung des Curriculum der Informatikstudiengänge [1] statt, bei welcher die Inhalte der Grundvorlesungen der Praktischen Informatik stärker in Richtung der ingenieurmäßigen Softwareentwicklung verlagert wurden. In drei jeweils vierstündigen Vorlesungen zu dieser Thematik werden Kenntnisse über Prinzipien, Techniken, Methoden und Werkzeuge für eine systematische Erstellung komplexer Software vermittelt, welche in einem abschließenden Softwarepraktikum mittels einer großen Aufgabe vertieft werden. Für dieses Praktikum werden neben einer Aufgabe im Feld der betrieblichen Informationssysteme und einer Aufgabe zu einem allgemeinen Gebiet, auch eine Aufgabe in der Domäne der eingebetteten Systeme (ES) angeboten. Dieser Artikel beschreibt die Konzeption der ES-Aufgabe [2] und analysiert die gemachten Erfahrungen.

Als Thema für die ES-Aufgabe wurde die Realisierung einer Ampelsteuerung gewählt, da dies eine leicht verständliche und alltägliche Anwendungsdomäne darstellt. Bei der Durchführung dieser Aufgabe werden alle Phasen eines typischen Softwareentwicklungsprozesses durchlaufen – auch wenn einige davon nur angedeutet werden. Am Ende eines kompletten Prozessdurchlaufs schließt sich eine Iteration mit modifizierter Aufgabenstellung an.

In Abschnitt 2 werden zunächst die Inhalte der ES-Aufgabe vorgestellt und von den, bezüglich der Inhalte, gemachten Erfahrungen berichtet. Anschließend wird in Abschnitt 3 die Durchführung des Praktikums beschrieben und einige Ergebnisse und Konsequenzen dargestellt.

2 Die Praktikumsaufgabe

Aufgabe ist die Entwicklung einer Ampelsteuerung für einen sechs Ampelkreuzungen umfassenden Ausschnitt aus dem Straßennetz von Kaiserslautern. Das zu entwickelnde System soll sowohl die lokale Steuerung der Ampeln *einer* Kreuzung, als auch die koordinierte Steuerung *aller* Ampeln in einer „Grünen Welle“ ermöglichen. Desweiteren soll das System über eine Benutzerkonsole verfügen.

Den Studierenden wurden zu Beginn die Anforderungen an das System aus Kundensicht und die Straßenpläne der Innenstadt ausgehändigt. Desweiteren erhielten sie Auszüge aus einem domänenspezifischen Handbuch. Die Entscheidung, die Aufgabe für einen konkreten Straßenzug aus der Stadt der Hochschule durchführen zu lassen, hat sich äußerst positiv auf die Motivation der Praktikumsteilnehmenden ausgewirkt.

Zusätzlich zu obigen Dokumenten wird für jede Phase der Systementwicklung zusätzlich eine Einführung in die zu verwendende Modellierungs-, bzw. Implementierungstechnik gegeben und jeweils erprobte Methoden für die Anwendung dieser Techniken vorgestellt. Die dabei präsentierten Inhalte werden in den folgenden Unterabschnitten skizziert.

2.1 Analyse und Entwurf

Während der Analysephase findet zunächst eine Modellierung der Struktur und des Verhaltens des Systems mit Hilfe von UML [3] statt. Neben einem Ausgangspunkt für die anschließende Implementierung dienen diese Modelle zusätzlich der Dokumentation. Deshalb wird von den Studierenden verlangt, diese Modelle stets auf dem aktuellen Stand zu halten.

Die Struktur des Systems soll durch ein Klassen- und ein Instanzenmodell beschrieben werden. Als Methode zur Aufstellung solcher Modelle wird vorgeschlagen, die physikalischen Objekte (Sensoren, Aktuatoren, Kontroller, etc.) als Ausgangspunkt zu nehmen, da sich dieser Ansatz in einer in unserer Arbeitsgruppe entwickelten Anforderungsanalysemethode bereits bewährte [4]. Zusätzlich erlaubt diese Modellierung von Klassen in der Analysephase eine frühe Festlegung von Entwicklerverantwortlichkeiten.

Um Testfälle für den späteren Test des Systems zu erhalten, sollen während der Analyse Szenarien aufgestellt werden, die ausgewählte Aspekte der gewünschten Systemfunktionalität beschreiben. Diese Szenarien dienen zudem als Einstieg für die anschließende Verhaltensmodellierung. Die Modellierung solcher Szenarien erfolgt mit Hilfe von UML-Sequenzdiagrammen, wobei zur Kommunikation asynchrone Nachrichten, welche typisch für eingebettete Systeme sind, Verwendung finden. Nach der Aufstellung dieser Sequenzdiagramme soll das Verhalten dann vollständig mit Hilfe endlicher Automaten in UML-Zustandsdiagrammen spezifiziert werden.

Ein Entwurf (Design) im eigentlichen Sinne kann in diesem Praktikum aus Zeitgründen nicht durchgeführt werden, da dazu z. B. die endgültige Zielhardware und nicht-funktionale Anforderungen wie z. B. Verlässlichkeit berücksichtigt werden müssten. Daher entwickeln die Studierenden einen Prototyp der Ampelsteuerung.

Als Modellierungswerkzeug wird die kommerzielle Telelogic Tau SDL Suite [5] zur Verfügung gestellt. Dieses Tool eignet sich insofern zur UML-Modellierung, dass auch Editoren für die wichtigsten UML-Diagrammtypen vorhanden sind.

Ergebnisse

Die Modellierung des Systems wurde von den Studierenden ohne schwerwiegende Probleme durchgeführt, da die UML-Notation aus den vorausgegangenen Vorlesungen gut verstanden war. Die abgegebenen Klassendiagramme hatten im Schnitt eine Zahl von 11 Klassen und 15 Relationen mit nur geringen Abweichungen zwischen den Gruppen. Diese relativ einheitliche Komplexität (im Vergleich zu Praktika wie z. B. [6]) liegt sicherlich an der Vorgabe, sich an den physikalischen Objekten zu orientieren. Die Zustandsautomaten besaßen eine mittlere maximale Komplexität von 54 Zuständen und 88 Transitionen.

Die Entscheidung, ein kommerzielles Werkzeug an Stelle freier Software einzusetzen, hat sich in der hohen Qualität der abgegebenen Modelle bezahlt gemacht. Viele der freien Tools, die wir in den vorausgegangenen Jahren eingesetzt hatten, verfügten oft über keinerlei Syntax-Prüfungen oder ermöglichten keine Ausgabe der Modelle in einem vernünftigen Format. Der Umgang mit dem Werkzeug konnte von den Studierenden schnell erlernt werden, vermutlich auch weil nur die Editor-Funktionalität Verwendung fand und keine komplexeren Funktionen benutzt wurden.

2.2 Implementierung und Test

Die Implementierung erfolgt in der Programmiersprache Java [8], wobei sehr großen Wert auf eine *sinnvolle* Dokumentation des Codes gelegt wird [9].

Am Anfang der Implementierung wird eine Abbildung der Modellklassen auf Klassen der Programmiersprache vorgenommen und die Relationen zwischen diesen Klassen umgesetzt. Daran schließt sich die Implementierung der Zustandsdiagramme an. Dabei wird von einer prozeduralen Realisierung mittels bedingter Anweisung (Case- oder If-Then-Else-Konstrukte) abgeraten, weil diese Lösung zu einem sehr unübersichtlichem und schlecht erweiterbarem Code führt. Als objektorientierter Realisierungsansatz wird deshalb das State-Pattern [10] propagiert.

Bei der Implementierung der grafischen Benutzersschnittstelle hatten die Studierenden außer der Vorgabe, das Model-View-Controller-Pattern [10] anzuwenden, weitgehend freie Hand.

Die Verbindung der Ampelsteuerung mit der Umgebung wird mittels der Java-RMI-Technik [11] aufgebaut. Da die Ampelsteuerung verständlicherweise nicht mit der echten Umgebung getestet werden konnte, wurde den Studierenden eine grafische Simulation der Umgebung zur Verfügung gestellt, welche neben dem Verhalten der Ampeln auch eine dynamische Anzahl an Fahrzeugen simuliert.

Ergebnisse

Bei der Beurteilung der Abgaben hat sich gezeigt, dass viele der OO-Konzepte noch nicht vollständig von den Studierenden beherrscht wurden. So hatten z. B. einige Gruppen erhebliche Mengen von Code produziert, was durch den Einsatz von Vererbung nicht nötig gewesen wäre. Dies dürfte wohl stark mit der Art der Übungen der vorausgegangenen Vorlesungen zusammenhängen, bei welchen solche Überlegungen auf Grund der geringen Größe der Aufgaben nicht nötig waren oder sich nicht so deutlich in einem erhöhten Aufwand niederschlugen. Diese Erfahrung zeigt deutlich, wie wichtig die Vertiefung der Kenntnisse an einem großen Beispiel ist. Auch in [12] kommt man zu diesem Fazit.

Desweiteren gab es bei der Umsetzung der Pattern Probleme. Diese wurden zu stark im Sinne von Templates verstanden, weshalb die Übertragung der eigentlichen Lösungsidee auf die konkrete Aufgabenstellung zum Teil fehlschlug. Mögliche Ursachen für diese Problematik können Schwierigkeiten bei der Vermittlung objektorientierter Programmierung (siehe [12]) oder ein mangelnder Unterricht in Mustern sein (siehe [14]). Man kann dies sicherlich verbessern, indem in den dem Praktikum vorausgehenden Vorlesungen der Charakter der Pattern besser herausgearbeitet wird.

Die Größe des von den Gruppen jeweils abgegebenen Codes lag zwischen 4400 und 15000 Zeilen (im Schnitt bei 8000 Zeilen) inklusive der Kommentare. Obwohl die Anforderungen an die GUI sehr moderat formuliert waren, investierten dennoch viele Gruppen großen Aufwand in diesen Teil. Die Komplexität der GUI erreichte in einigen Fällen die Komplexität der eigentlichen Steuerung. Im Durschnitt betrug der Anteil des GUI-Codes am Gesamt-Code 30%. Hier sollte über eine genauere Vorgabe – besonders auch dessen was *nicht* gefordert ist – nachgedacht werden, um die Verteilung des Arbeitsaufwandes auf die gewünschten Schwerpunkte zu lenken.

2.3 Iteration

Im Softwarepraktikum sollen die Studierenden nicht nur eine Entwicklung von Null durchführen, sondern auch die Probleme bei einer Erweiterung oder Änderung eines bestehenden Systems erkennen. Daher wurde am Ende des Praktikums eine Iterationsphase angeschlossen, für welche die Umsetzung einer geänderten Problembeschreibung gefordert war. Typische Änderungen waren das Hinzufügen einer zusätzlichen Ampel und die Veränderung der Hauptverkehrsrichtung der „Grünen Welle“.

Ergebnisse

Da die Iteration schon zu Beginn des Praktikums angekündigt wurde, stellte sich diese Phase als unproblematisch für die meisten Gruppen dar. Lediglich der Aufwand für die Anpassung des Systems variierte. So gab es Gruppen, die ihre Steuerung sehr generisch ausgelegt hatten während andere nicht so starken Wert auf Generizität gelegt hatten. Die durchschnittliche Zunahme des Codes lag bei 2500 Zeilen. Das führte zu einer durchschnittlichen Gesamtkomplexität des am Ende des Praktikums verfügba-

ren Codes von 10000 Zeilen. Im Vergleich zu anderen Praktika (z.B. [6] oder [7]) scheint diese Komplexität relativ hoch, weshalb man, auch unter Berücksichtigung der in Abschnitt 3 vorgestellten Resultate, über eine Vereinfachung der Aufgabe nachdenken sollte.

3 Die Durchführung des Praktikums

Für die Durchführung des Praktikums wurden zwölf Wochen veranschlagt, in welchen Gruppen von jeweils sechs Personen die einzelnen Phasen der Softwareentwicklung wie in Abbildung 1 gezeigt durchlaufen. Dabei beinhaltet die Phase der Iteration auch die Vorbereitung auf eine Abschlusspräsentation des Systems. Hier wird neben einer 20-minütigen Systemvorführung durch die ganze Gruppe, von jedem Gruppenmitglied auch ein kurzer, ca. fünf-minütiger Vortrag zu einem ausgewählten Gebiet der Praktikumsaufgabe verlangt.

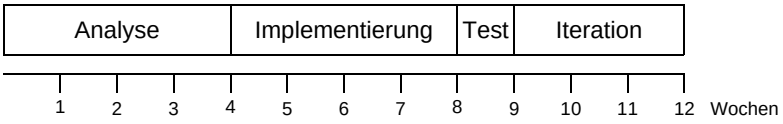


Abbildung 1: Verteilung der Phasen auf die Laufzeit des Praktikums

Das Softwarepraktikum ist als halbtägiges Praktikum geplant, weshalb für den Bruttoarbeitsaufwand der Studierenden von 12 Semesterwochenstunden ausgegangen wird. In diesem Aufwand sind neben der Zeit für das eigenständige Arbeiten in den Gruppen, die Vorstellungen der jeweiligen Aufgaben durch die Betreuer (ca. 30 Min. pro Woche) und sogenannte Präsenzzeiten (mind. eine Stunde pro Woche) enthalten. Die Studierenden werden dabei in den Präsenzzeiten permanent von Mitarbeitern oder Hilfskräften betreut, wodurch Fragen und Probleme sofort diskutiert werden können.

Als Motivationsschub für das Praktikum wird am Ende ein kleiner Wettbewerb durchgeführt, in welchem die Steuerungssysteme der Gruppen nach drei Kriterien beurteilt werden und die jeweilige Gewinnergruppe einen kleinen Preis und Urkunden erhält.

Ergebnisse

Die Einteilung in die in Abbildung 1 gezeigten Phasen hat sich als günstig erwiesen. Lediglich die gegen Ende des Semester angesetzte Systemvorführung hat sich als problematisch herausgestellt, da wenige Tage später die Semesterabschlussklausuren anderer Veranstaltungen stattfanden. Da für die Einzelvorträge nur jeweils fünf Minuten blieben, ist für die Zukunft geplant, die Vorträge über das ganze Semester verteilt und dann jeweils über ca. 20–30 Minuten halten zu lassen. Dies kann z.B. sinnvoll im Rah-

men der Präsenzzeiten erfolgen. Auch um die Präsentationsfähigkeiten der Studierenden zu verbessern scheint eine solche Erweiterung äußerst sinnvoll (siehe [15]).

Als vorteilhaft hast sich die Gruppengröße von sechs Studierenden herausgestellt, da dadurch die sinnvolle Aufteilung der Arbeit innerhalb der Gruppe auf Teams von zwei Personen möglich wird. Desweiteren erfordert diese Zahl an Gruppenmitgliedern eine organisierte Kommunikation. Ein Team von einem Mitarbeiter und einer wissenschaftlichen Hilfskraft konnte jeweils drei solcher Gruppen gut betreuen.

Zur Einführung von Präsenzzeiten entschlossen wir uns, da ein großer Anteil der Studierenden nur noch am heimischen PC arbeitete und der Kontakt zu den Betreuern ausschließlich bei kritischen Problemen gesucht wurde. Da dadurch der typische Praktikumscharakter verloren ging, versuchten wir mit den Präsenzzeiten einen Kompromiss zu finden. Desweiteren sahen wir darin eine sinnvollere Möglichkeit, die Individualleistung einzelner Teilnehmer zu beurteilen, als dies durch das bisherige Testieren möglich war. Das Resultat dieser Änderung wird in Abschnitt 3.1 diskutiert.

Die Durchführung des Wettbewerbs am Ende wurde von den Studierenden positiv aufgenommen. Interessant wäre, wenn man durch diesen Wettbewerb einen Zusatznutzen für den Lernerfolg der Teilnehmer erzielen könnte. Anregungen dazu finden sich in [16].

Erstmalig in diesem Jahr führte die Fachschaft Informatik eine Praktikumsumfrage durch, in welchem 40% der Studierenden den Gesamteindruck des Praktikums mit „sehr gut“ bis „gut“ bewerteten und 35% dem Praktikum ein „befriedigend“ gaben. Als Kritikpunkte wurden von dem jeweils in Klammern angegebenen Anteil an Studierenden genannt:

1. sehr hoher Arbeitsaufwand von mehr als zwölf Stunden (45%)
2. ungenaue Formulierung der Aufgabenstellung und der Dokumentation (30%)
3. hoher Schwierigkeitsgrad der Aufgaben (25%)

Bei der Beurteilung des ersten Punktes sollte man berücksichtigen, dass für viele der Teilnehmer die Alternative zum Mehraufwand das Nichtbestehen gewesen wäre. Dennoch sollte eine Reduzierung der Aufgabe (durch z.B. Verringerung der Anzahl betrachteter Kreuzungen) in Betracht gezogen werden, da eine Reduktion des Anteils der GUI am Gesamtsystem (siehe Abschnitt 2.2) zwar eine Verbesserung bringen könnte, aber vermutlich nicht in dem Maße wie nötig. Bei den Punkten 2 und 3 kommt sicherlich die in Abschnitt 2.2 erwähnte Problematik des Umgangs mit OO-Konzepten und Pattern zum tragen. Dieses wird momentan durch eine Umstellung der Vorlesungen angegangen.

Positiv wurde von den Studierenden der Praxisbezug bewertet. Desweiteren wurde die Arbeit im Team und die Anwendung des in den vorausgegangenen Vorlesungen gelernten Stoffes als Pluspunkte des Praktikums angegeben.

3.1 Benotung

Die Gesamtleistung eines jeden Teilnehmers setzt sich aus 50% Gruppenleistung und 50% Individualleistung zusammen. Zur Gruppenleistung tragen die Abgaben zu den einzelnen Aufgaben und die Abschlusspräsentation bei. In die Individualbewertung fließt die Leistung während der Präsenzzeiten und die Qualität des Einzelvortrags ein.

Ergebnisse

Abbildung 2 zeigt einen Vergleich der Noten der Jahre 2001 und 2002. In unseren Augen ergab sich die deutliche Notenverbesserung (um im Schnitt 0,7 Notenstufen) durch Einführung der Präsenzzeiten.

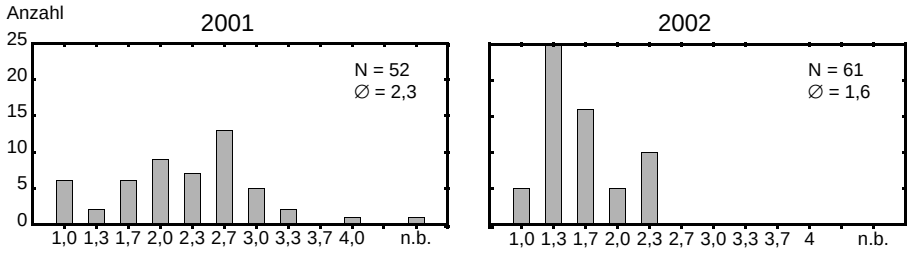


Abbildung 2: Notenverteilung

Neben der deutlichen Verbesserung der Noten aller Teilnehmer reduzierte sich zudem die Variation der Noten innerhalb der einzelnen Gruppen, wie Abbildung 3 zeigt. Diese Tatsache wirkt sich mit Sicherheit positiv auf den bleibenden Eindruck des Praktikums für die einzelnen Teilnehmer aus.

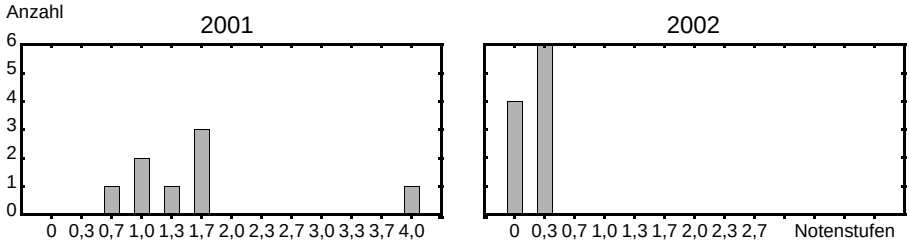


Abbildung 3: Maximale Abweichung der Noten innerhalb einer Gruppe

Wie in [16] richtig formuliert wird, sollte man an dieser Stelle aber die Frage stellen, ob eine Individualbewertung in einem Teampraktikum überhaupt sinnvoll ist. Mit den obigen Resultaten könnte man nun auch umgekehrt argumentieren und aus der sehr geringen Abweichung der Noten innerhalb einer Gruppe folgern, dass eine Individualnote nicht benötigt wird. Die Frage, ob die Notenverbesserung auch wirklich eine Verbesserung des Wissens der Teilnehmenden ergeben hat, lässt sich mit den vorliegenden Ergebnisse leider nicht beurteilen.

4 Schlussbetrachtungen

Das Softwarepraktikum im Grundstudium hat sich als eine sinnvolle Bereicherung des Curriculums der Informatikstudiengänge an der Universität Kaiserslautern herausgestellt. Wie an Hand der qualitativen und quantitativen Ergebnisse aufgezeigt wurde, kann und sollte eine stetige Verbesserung der Güte eines solchen Praktikums durch die Berücksichtigung der Ergebnisse vorausgegangener Jahre erzielt werden.

Zum Schluss gilt mein Dank allen Studierenden, Mitarbeitern und Dozenten, welche eine solch erfolgreiche Durchführung des Softwarepraktikums ermöglichten.

Literatur

1. Bunke, B.; Faber, K.; Kirchner, R. et al.: Informatik-Studium. Studienführer Informatik, Universität Kaiserslautern, 1999
   http://www.informatik.uni-kl.de/studium/studienfuehrer/html_noframe/node16.html
2. Metzger, A.: Softwarepraktikum – Teil: Eingebettete Systeme. Web-Seite, Universität Kaiserslautern, 2002
   http://wwwagz.informatik.uni-kl.de/courses/ss02/SWP
3. Booch, G; Rumbaugh, J; Jacobson, I: The Unified Modeling Language User Guide. Addison-Wesley Longman, Reading, Mass., 1999
4. Metzger, A.; Queins, S.: Specifying Building Automation Systems with PROBAnD, a Method Based on Prototyping, Reuse, and Object-orientation. Hofmann, P.; Schürr, A. (Hrsg.): OMER – Object-Oriented Modeling of Embedded Real-Time Systems. GI-Edition, Lecture Notes in Informatics (LNI), P-5, Köllen Verlag, Bonn, 2002, S. 135–140
5. Telelogic Tau SDL Suite. Web-Seite, Telelogic, AB, Schweden, 2002
   http://www.telelogic.com/products/tau/sdl/index.cfm
6. Demuth, B.; Hußmann, H.; Zschaler, S. et al.: Erfahrungen mit einem frameworkbasierten Softwarepraktikum. Dreher, B.; Schulz, Ch.; Weber-Wulff, D. (Hrsg.): Software Engineering im Unterricht der Hochschulen SEUH ‘99, B.G. Teubner, Stuttgart, 1999, S. 21–30
7. Bruegge, B.: From Toy Systems to Real System Development: Improvements in Software Engineering Education. Hußmann, H.; Paech, B. (Hrsg.): Software Engineering im Unterricht der Hochschulen SEUH ‘94, B.G. Teubner, Stuttgart, 1994, S. 62–72
8. Eckel, B: Thinking in Java. 2nd Edition. Prentice Hall, Upper Saddle River, New Jersey, 2000.
9. Vermeulen, A; Ambler, S.W.; Bumgardner, G. et al.: The Elements of Java Style. Cambridge University Press, Cambridge, 2000
10. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, Mass., 1995.
11. Java Remote Method Invocation Specification. Online-Dokument, Sun Microsystems, 2002
    ftp://ftp.java.sun.com/docs/j2se1.4/rmi-spec-1.4.pdf
12. Schneider, K.: Auf der Suche nach maßgeschneiderten Unterrichtsformen: Das angeleitete Praktikum. Raasch, J.; Bassler, T. (Hrsg.) : Software Engineering im Unterricht der Hochschulen SEUH ‘93, B.G. Teubner, Stuttgart, 1993, S. 66–77
13. Reus, B: Software-Entwicklung im Grundstudiumspraktikum: nein – ja – Java? SEUH ‘99, S. 45–54
14. Engelien, M.; Vocke, H.; Wunsch, R.: Zum Spannungsausgleich zwischen objektorientierter und herkömmlicher Softwareentwicklung. Spillner, A.; Breymann, U. (Hrsg.): Software Engineering im Unterricht der Hochschulen SEUH ‘95, B.G. Teubner, Stuttgart, 1995, S. 109–118
15. Weber-Wulff, D.: Teambildung in Programmierung und Software-Engineering Kursen. SEUH ‘95, S. 82–89
16. Hornecker, E.: Teamtraining und Präsentationstechniken für das Software-Engineering-Praktikum, SEUH ‘95, S. 69–81

VirtuOhm – Konzept einer virtuellen Lernumgebung

Hans-Georg Hopf,
Georg-Simon-Ohm-Fachhochschule Nürnberg, hans-georg.hopf@fh-nuernberg.de

Zusammenfassung

Effektives Lernen, ohne unnötigen Ballast, zeitlich flexibel und immer mit Unterstützung durch einen Mentor, wenn gerade Bedarf ist, eine fast ideale Lernsituation! Mit E-Learning sollen diese Ziele erreicht werden. Welche Anforderungen ergeben sich an virtuelle Lernumgebungen?

In diesem Artikel soll das Konzept der virtuellen Lernumgebung „VirtuOhm" vorgestellt werden. Ziel des VirtuOhm-Konzepts ist zunächst eine über das Internet zugreifbare Infrastruktur zur Verfügung zu stellen, die geeignet ist, an der eigenen Fortbildung bzw. Weiterbildung interessierten Personen eine kooperative und zeitlich flexible Bearbeitung des Lernangebots aus der Distanz zu ermöglichen, ohne in der Kommunikation Nachteile hinnehmen zu müssen. Das Konzept ist nicht nur auf ein reines Selbststudium ausgerichtet, sondern eignet sich auch für die Kombination mit der „klassischen" Präsenzlehre (blended learning). Die Lernumgebung VirtuOhm wurde an der Georg-Simon-Ohm-Fachhochschule in Nürnberg in den letzten Jahren konzipiert und in großem Umfang realisiert.

1 Motivation

Die Georg-Simon-Ohm-Fachhochschule Nürnberg bietet seit dem WS 1998/99 ein berufsbegleitendes zweisemestriges Weiterbildungsstudium Softwareengineering an. In den ersten drei Jahren haben ca. 80 Teilnehmer das Zertifikatsstudium absolviert. Die intensive Begleitung der Weiterbildungsteilnehmer, die üblicherweise in den späten Nachmittagsstunden und an Samstagen Lehrveranstaltungen besuchten, war ein Auslöser zur Beschäftigung mit E-Learning. E-Learning bot die Chance, Präsenzzeiten zu verringern und Teilnehmern in der angespannten zeitlichen Situation mehr Spielraum einzuräumen. Die Arbeiten an der E-Learning-Komponente von VirtuOhm begannen 1999. Es hat sich schnell gezeigt, dass E-Learning Präsenzanteile nicht vollständig ersetzen kann. Das vorgestellte VirtuOhm-Konzept ist deshalb nicht nur auf ein reines Selbststudium ausgerichtet, sondern ist auch für die Kombination mit der „klassischen" Präsenzlehre (blended learning) ausgelegt. Derzeit wird versucht, die Vorlesung „Softwarequalität" aus dem grundständigen Informationstechnikstudium durch virtuelle Vorlesungsanteile neu zu gestalten. Es wer-

den E-Learning- Anteile jeweils für einen Vorlesungstermin zugänglich gemacht. Damit sollen Vor- bzw. Nachbereitung des Lehrstoffes effektiver gestaltet werden.

2 Didaktisches Konzept einer virtuellen Lehrveranstaltung

2.1 Strukturierung des Lehrstoffes

Bücher geben ein Beispiel für die Strukturierung von Lernstoff und auch für den Umgang mit dem Lehrstoff. Ein Buch ist hierarchisch organisiert: Es besteht aus Kapiteln, die sich in Abschnitte und weitere Unterabschnitte untergliedern. Ein Buch wird üblicherweise zunächst sequentiell gelesen. Bei der sich anschließenden Arbeit mit dem Buch werden Informationen aus verschiedenen Kapiteln und Abschnitten miteinander in Beziehung gesetzt, um eine vorgegebene Aufgabenstellung zu lösen oder einfach das Gelesene zu begreifen. Die Beschäftigung mit den Inhalten eines Abschnitts in einem Buch ist geprägt von kreativem Arbeiten mit dem Text. Man blättert zurück, um eine Textpassage mit neuen, durch den bisherigen Erkenntnisgewinn beeinflussten Augen nochmals zu lesen, man blättert vor, um einen bekannten Sachverhalt zu überspringen, usw. Das sequentielle Lesen zu Beginn verschafft den Überblick, das Orientierungsraster, das logische Konzept.

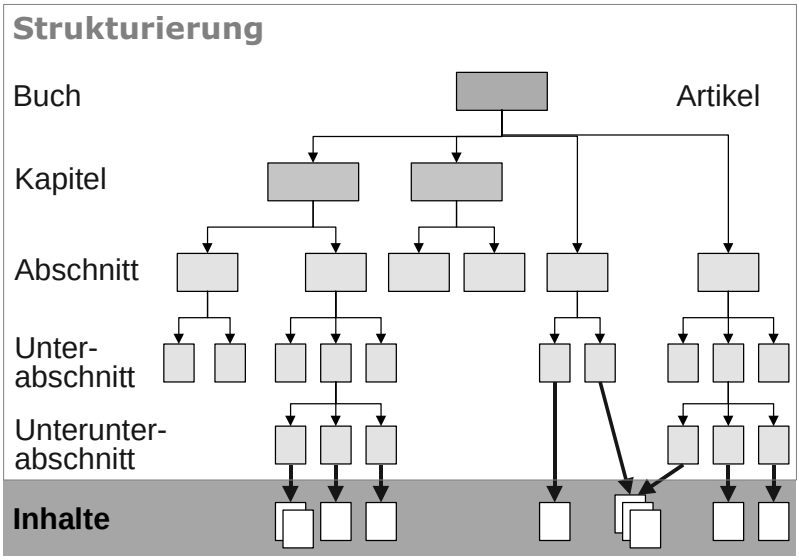


Abbildung 1: *Strukturierung von Lehrstoff in einem Buch oder einem Artikel*

Die zweite Nutzungsphase ist geprägt vom selbstgesteuerten explorativen Umgang mit dem angebotenen Buchinhalt. In einer dritten Phase dient das Buch als Nachschlagewerk. Man hat den Inhalt im Wesentlichen verstanden. Im Detail gibt es den einen oder anderen Sachverhalt, der erst durch eine aktuelle Aufgabe Bedeutung gewinnt und gezielt aus dem Aufgabenkontext lösungsdienlich exakt nachgeschlagen wird. Der bisher gewonnene Überblick über den Lehrstoff und das Nachschlageregister evtl. auch das Literaturverzeichnis leisten wertvolle Hilfe beim Auffinden der gesuchten Information.

Eine Lehrveranstaltung ist zunächst grundsätzlich ähnlich strukturiert: Sie wird zuerst hierarchisch organisiert und wird sequentiell durchlaufen. Es ergibt sich ein logisches Gerüst zum weiteren selbstgesteuerten explorativen Lernen. Der praktische Umgang mit der Materie, das situationsbedingte Nachschlagen, bewirkt eine Vertiefungsphase. Diese drei Phasen werden in Präsenzveranstaltungen oft nicht deutlich getrennt, sondern sind ineinander verzahnt enthalten. Der Dozent entwickelt sequentiell das logische Gerüst und stellt in seinen Kommentaren die Bezüge zu anderen Teilen der Lehrveranstaltung her. Integrierte Übungs- und Praxisphasen führen zum gesicherten Lernergebnis.

Beim Übertragen auf eine online angebotene Lehrveranstaltung sollte allen Strukturierungselementen Rechnung getragen werden. Aus diesem Grund unterscheidet man verschiedene Arten von Navigationssystemen: das hierarchische Navigationssystem, das logische oder assoziative Navigationssystem und das lexikalische Navigationssystem.

- Eine starke Führung des Benutzers wird durch ein hierarchisches Navigationssystem erreicht. Der Benutzer wird bei seinem sequentiellen Weg durch eine Anwendung unterstützt. Das Navigationssystem gibt ihm Hilfestellung, sich in unbekannten Strukturen zurechtzufinden und den „richtigen“ Anknüpfungspunkt zu finden.

- Ein dem explorativen Lernen angemessenes Navigationssystem unterstützt den kreativen Umgang mit den dargebotenen Inhalten. In einem logischen Navigationssystem kann der Benutzer seinen eigenen Vorstellungen kreativ folgen und wird insofern unterstützt, als er stets wissen sollte, an welcher Stelle der Struktur er sich befindet.

Eine lexikalische Suche nach Begriffen und Schlüsselworten schafft die Möglichkeit, das konkrete Informationsbedürfnis zur Lösung einer aktuellen Aufgabenstellung gezielt und effizient zu befriedigen. Hier steht das schnelle und sichere Suchen und Auffinden der Information im Vordergrund

Das im VirtuOhm-Konzept eingesetzte hierarchische Navigationssystem ist in der Abbildung 3 dargestellt. Den Hierarchiestufen einer Buchgliederung (Kapitel,

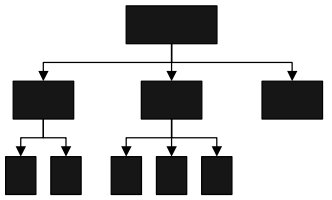
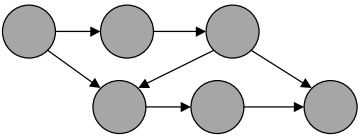
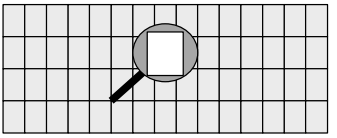
Hierarchisches Navigationsschema Gedankliche Führung	
Logisches / assoziatives Navigationsschema Kreativität	
Lexikalisches Navigationsschema Suche	

Abbildung 2: Klassifikation von Navigationssystemen

Abbildung 3: Realisierung eines hierarchischen Navigationssystems

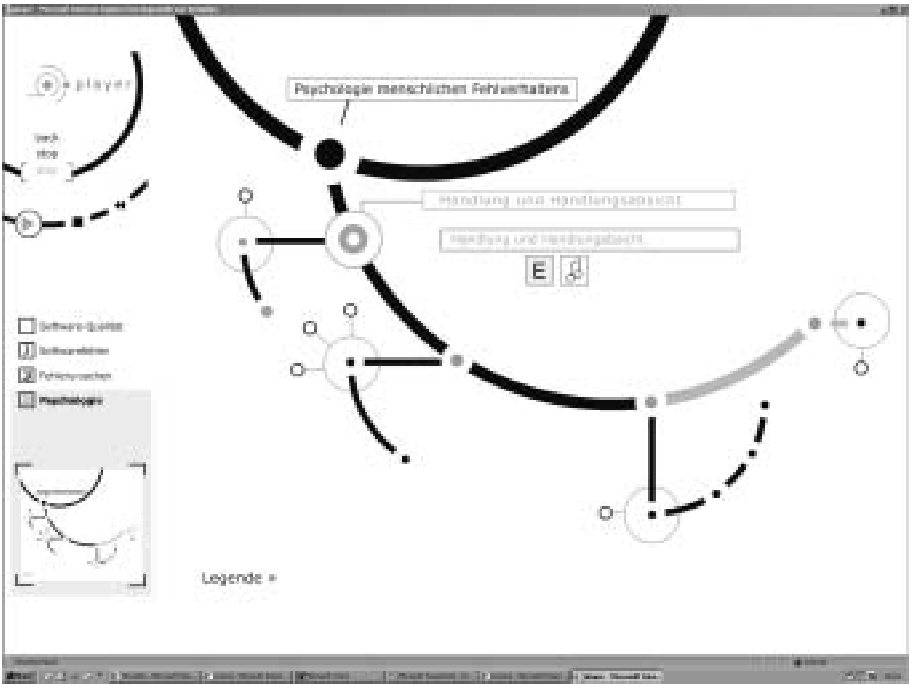


Abbildung 4: Realisierung eines logischen oder assoziativen Navigationssystems

Abschnitte, Unterabschnitte, ...) entsprechen die dargestellten Schalen. Auf jeder Schale stellen Auswahlpunkte die verschiedenen Kapitel dar. Nach Auswahl eines Abschnittes auf der letzten aktiven Schale verzweigt das Lernsystem in einen so genannten „player“, der zur Darstellung der eigentlichen Inhalte vorgesehen ist.

Zur Erschließung der Inhalte ist eine logische Navigationsstruktur vorgesehen. Abbildung 4 zeigt eine mögliche Realisierung für ein derartiges Navigationssystem. Dem Benutzer bleibt es überlassen, welche Abschnitte er zuerst bearbeiten möchte. Ein Leitsystem erschließt ihm abhängig von dem konkret gewählten Abschnitt die dazu gehörigen Inhalte.

Das lexikalische Navigationssystem ist durch eine Suchmaske realisiert. Wichtig ist die über die Suche angestoßene Funktionalität. Je mächtiger der Suchalgorithmus und je breiter die Suche, desto hilfreicher ist diese Funktionalität.

2.2 Interaktive Arbeit mit dem Lernangebot

Der Lehrstoff wird vom Teilnehmer aktiv erarbeitet. Dazu ist ein Leitsystem aus orange farbigen Punkten implementiert. Durch orangefarbige Punkte sind abhängig vom aktuell bearbeiteten Abschnitt mögliche Fortsetzungen gekennzeichnet. Je nach Lernsituation kann der Lerner Wissen vertiefen oder sich zunächst einen Überblick verschaffen oder einen komplexeren Gedanken in sinnvolle Schritte unterteilt nachvollziehen.

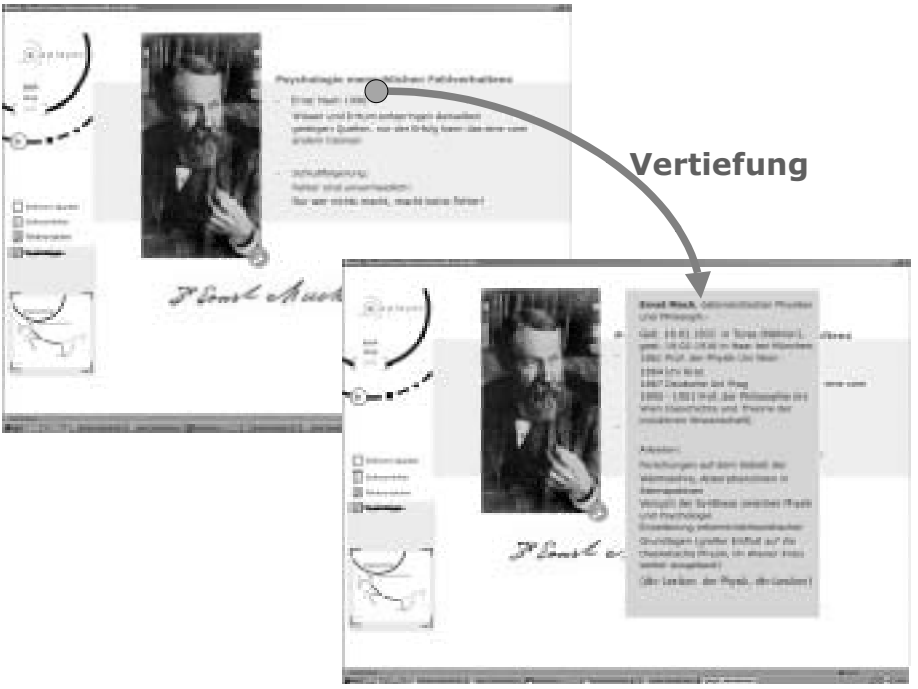


Abbildung 5: Interaktive Arbeit mit dem Lernangebot

Für persönliche Anmerkungen zum Lernstoff steht eine Annotationsfunktion zur Verfügung. Dieser „elektronische Notizzettel“ ermöglicht dem Lerner seine individuellen Anmerkungen einzubringen, Kommentare anzufügen, Referenzen auf eigene Quellen zu hinterlegen oder auch noch zu klärende Punkte aufzulisten. In einer Literaturdatenbank werden Fachartikel zur Verfügung gestellt. Diese können dann von den Teilnehmern kommentiert werden. Die Kommentare sind von allen Teilnehmern der Lerngruppe einsehbar. Die Erfahrungen eines Teilnehmers werden auf diese Weise allen anderen Mitlernern verfügbar.

2.3 Interaktive individuelle Lernzielkontrolle

Ein Online-Lernangebot verleitet leicht dazu, sich einige im ersten Augenblick interessante Aspekte herauszupicken und andere Abschnitte zu überspringen oder mit nur mäßiger Intensität zu bearbeiten. Damit entsteht der Eindruck, „alles“ gelernt zu haben. Um dem Lerner eine Möglichkeit zur Selbsteinschätzung zu geben, sind Tests eingebaut. Jedes größere Kapitel wird mit einem Test abgeschlossen. Das Erreichen einer pro Test festzulegenden Punktezahl ist die Voraussetzung für die Fortsetzung des Studiums mit dem nächsten Kapitel. Bei Nichtbestehen ist eine beliebige Anzahl an Wiederholungen möglich. Der Test

besteht aus einem Fragenkatalog. Die Fragen sind z.B. als Multiple-Choice-Aufgaben formuliert, die unter Zeitvorgabe gelöst werden müssen.

2.4 Kommunikation

Ein wesentliches Erfolgsmerkmal für Lernen ist die Kommunikation mit dem Lehrer aber auch mit anderen, die am gleichen Problem arbeiten, also mit den Kommilitonen. Die Kommunikation muss im Fall einer Online-Lernveranstaltung durch technische Hilfsmittel unterstützt werden:

- Für organisatorische Mitteilungen des Dozenten steht eine Anwendung (News, schwarzes Brett) zur Verfügung. Auf diese Weise gibt der Dozent Anweisungen zur Bearbeitung der einzelnen Abschnitte oder generelle Hilfestellungen.
- Weiter steht ein Diskussionsforum (Bulletin-Boards, Chat-Forum, Newsgroups) zur Verfügung. Auf diese Weise wird asynchron ein „Gespräch" zwischen allen Beteiligten möglich, unabhängig, zu welchem Zeitpunkt eine Frage, ein Kommentar oder eine Antwort in das Chat-Forum eingestellt wird. Es entsteht ein nachlesbares Kommunikationsprotokoll.
- Für synchrone Kommunikation ist ein „live Chat" möglich. Hier soll bewusst den gerade aktiven Lernern die Möglichkeit gegeben werden, sich aktuell auszutauschen. Diese Möglichkeit kommt sicher bei den häufig ungewöhnlichen Lernzeiten gerade von Berufstätigen der Motivation zugute: Man kann sich unter „Leidensgenossen" gegenseitig ermuntern und so auch unter schwierigen Verhältnissen den Lernerfolg unterstützen.

2.5 Wissenskonstruktion

Ziel der virtuellen Lehrveranstaltung ist Wissen aufzubauen. Der angebotene strukturierte Lehrstoff bietet zunächst nur die Wissensgrundausstattung. Durch die Integration von „Wissensdatenbanken" sollte es jedem Teilnehmer möglich sein, individuell Wissen zu vertiefen. Einschlägige Artikel zu dem angesprochenen Thema können hier hilfreich sein. Es bietet sich an, eine vertiefende Darstellung standardmäßig verfügbar zu machen. Im VirtuOhm-System wird über ausdruckbare Textdokumente eine ausführliche zusammenfassende Darstellung des Inhalts angeboten. Diese Darstellung ergänzt die unter didaktischen Gesichtspunkten für „Einsteiger" optimierte Online-Darstellung. Ob das Gelernte tatsächlich zum aktiven Wissen geworden ist, zeigt sich oft erst, wenn es zur Anwendung kommt. Die Sicherung des Gelernten erfolgt im Kontext eines realen Projektes (siehe Abbildung 6). Hier muss sich zeigen, ob Inhalte verstanden und Fertigkeiten erworben wurden. Treten Defizite auf, ist dies der Anlass zur erneuter Beschäftigung mit dem Lernstoff. Dabei kann der Dozent individuell und gezielt eingreifen und Hilfestellung leisten.

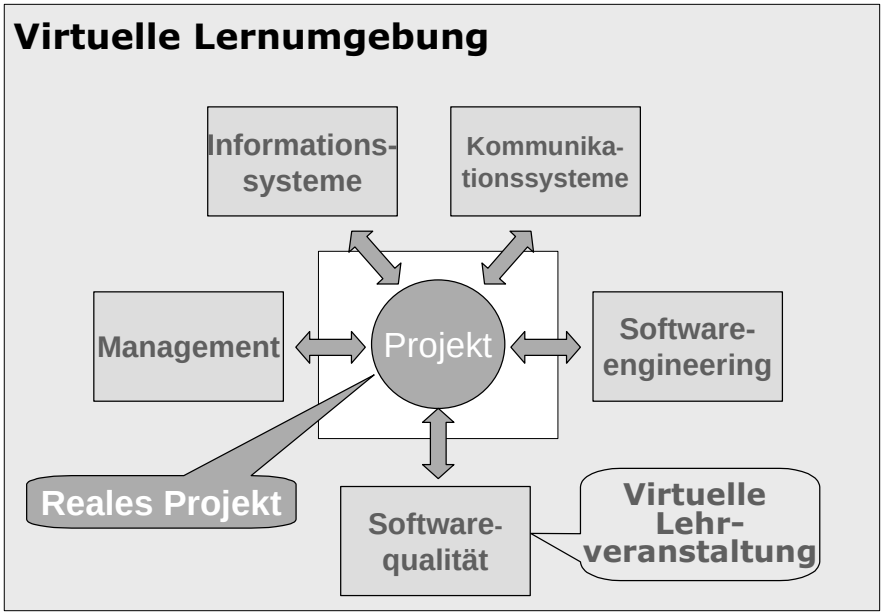


Abbildung 6: Verbindung zwischen Präsenz- und Online-Lehre„blended learning“

3 Virtuelle Lernumgebung

Lehrveranstaltungen sind in Lernumgebungen eingebettet. So finden sich z.B. reale Vorlesungen im Kontext eines Präsenzstudiums. Natürlich muss auch für virtuelle Lehrveranstaltungen eine angepasste Lernumgebung zur Verfügung gestellt werden. Eine virtuelle Lernumgebung muss folgende Funktionalität anbieten:

- Zugang zur Lernumgebung
- Auswahl der Lehrveranstaltung, die der Lerner gerade bearbeiten möchte
- Besuchen der ausgewählten virtuellen Lehrveranstaltung

Die hier vorgestellte VirtuOhm-Lernumgebung realisiert jede dieser Funktionen auf einer eigenen Ebene (siehe Abbildung 7). Auf jeder Ebene wird das hierarchische Navigationssystem zur raschen Auswahl der gewünschten Funktion angeboten. Ein logisches Navigationssystem bleibt der Inhaltsebene vorbehalten. Die Suche erstreckt sich auf Auswahl- und Inhaltsebene, erschließt aber zusätzlich auch die gesamte über die konkrete Lehrveranstaltung hinausreichende Wissensbasis (andere Lehrveranstaltungen, Literaturbibliothek, Notizzettel, ...).

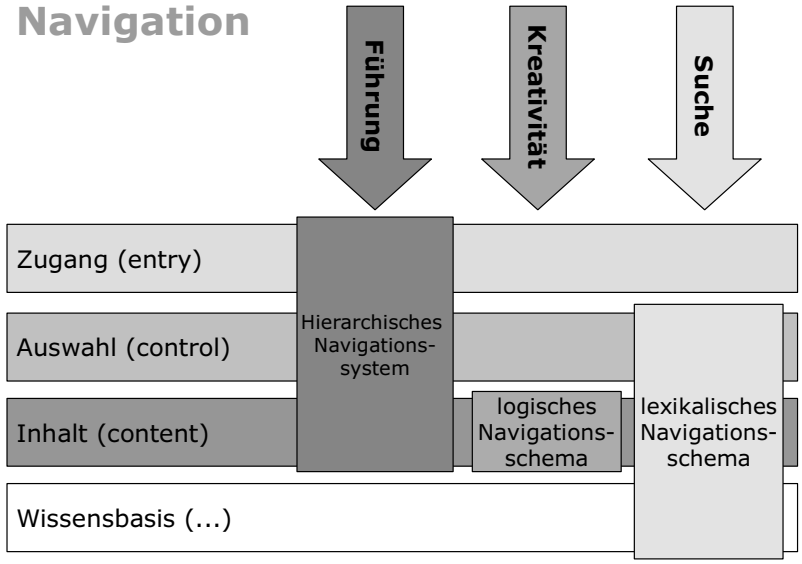


Abbildung 7: Navigationskonzepte einer virtuellen Lernumgebung

Die Zugangsmaske „entry“ ermöglicht die Anmeldung: Der Benutzer kann sich mit einer vorher vergebenen Benutzerkennung dem System bekannt machen. Dem Gast werden allgemeine Informationen zugänglich gemacht oder z.B. ein Schnupperkurs zum Kennenlernen des Angebots präsentiert. Es werden allgemeine Informationen angeboten: Bedienungshinweise erklären die Benutzung, ein schwarzes Brett informiert über Aktuelles, ein Chat-Forum erlaubt den Dialog mit Veranstaltern, Teilnehmern oder anderen Interessenten. Weiter wird über Studienveranstaltungen informiert und der Kontakt zu den Veranstaltern und Dozenten ermöglicht.

Ein eingeschriebener Teilnehmer, der sich mit seiner Benutzerkennung anmeldet, kommt automatisch in seine persönliche Lernumgebung. Er findet das von ihm gebuchte Lehrveranstaltungsprogramm vor. Er kann eine Lehrveranstaltung auswählen. Für jede Lehrveranstaltung bietet sich nun die Möglichkeit, das Studium aufzunehmen, sich über organisatorische Aspekte zu informieren oder mit Teilnehmern dieser Lehrveranstaltung oder dem Dozenten zu kommunizieren. Im Bereich Organisation werden der Stundenplan für Präsenzveranstaltungen oder Zeitfenster für die regelmäßigen Treffen im „virtuellen Klassenzimmer“ veröffentlicht. Ein Veranstaltungskalender macht auf zusätzliche „Events“ aufmerksam. Außerdem kann die Information über Prüfungen und die Anmeldung zur Prüfung hier erfolgen. Unter dem Menüpunkt Kommunikation finden sich die Angebote: Schwarzes Brett, Newsgroup, live Chat, Kontakte und Feedback.

Der Inhalt des Abschnitts präsentiert sich dann in einem so genannten „player“, einem eigenen Fenster zur multimedialen Darstellung der entsprechenden Information. Auf dieser Ebene steht mit dem logischen Navigationssystem ein dem explorativen Lernen angemessenes Hilfsmittel zur Verfügung. Bezüglich der Präsentation der Inhalte hat sich der Grundsatz bewährt: Der Einsatz von Medien bei der Aufbereitung der einzelnen Lernabschnitte orientiert sich an den jeweiligen Inhalten und den dafür geeigneten Lehrmethoden. Es erscheint nicht sinnvoll, ein Medium nur deshalb einzusetzen, weil man es gerade zur Verfügung hat. Das Ziel einer Lerneinheit ist es, dem Lernenden den Lernstoff möglichst effizient zu vermitteln. Je nach Lernstoff, den didaktischen Zielen und den methodischen Vorstellungen können hier unterschiedlichste Medien hilfreich sein, dieses Ziel zu erreichen. Es bleibt den Autoren (Lehrmittelerstellern) vorbehalten, mit welchen Werkzeugen sie die Präsentation erstellen. Beispiele für Werkzeuge sind: HTML-Editoren, Powerpoint, Flash ...

4 Schlussbemerkung

Es gibt eine Reihe von Beispielen für interessante Ansätze bei der Konstruktion von Lernumgebungen. Die Unternehmen Blackboard Inc. und eCollege.com gehören zu den Marktführern, aber auch viele Hochschulen sind auf diesem Gebiet tätig (siehe z.B. [3]). Das Engagement von Hochschulen mag darin begründet liegen, dass sie beim Schritt in die „virtuelle Welt“ nicht nur in der Lehre, sondern auch in der Verwaltung von gravierenden Veränderungen betroffen sind. Ein Student erwartet über das virtuelle Lehrangebot hinaus weitere sog. Selbstbedienungsfunktionen, die es ihm erlauben, sich über Netz zu Prüfungen anzumelden, Noten einzusehen oder andere Verwaltungsvorgänge abzuwickeln. Diese hochschulspezifischen Aspekte sind in heute verfügbarer Software oft nicht oder nur unzureichend berücksichtigt. Der Einstieg in die „eUniversity“ bedeutet jedoch immer eine strategische Entscheidung von großem Ausmaß: Neben der Entscheidung über Entwicklung oder Kauf muss auch die aufwändige Pflege und Wartung der umfangreichen Softwareanwendungen und der angebotenen Inhalte berücksichtigt werden.

Literatur

1. Hans-Georg Hopf, Harald Mohr, Christian Bretting, Katrin Proschek, Werner Aurich, Alexander Kotik: VirtuOhm – ein multimediales e-university Portal. Proceedings, Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik / Workshop „Unternehmen Hochschule 2002“, Dortmund, 02.10.2002
2 Christian Bretting: Konzeption und prototypische Realisierung der informationstechnischen Infrastruktur einer virtuellen Hochschule (VirtuOhm). Diplomarbeit, Georg-Simon-Ohm Fachhochschule, Nürnberg, 2000.
3 Rolf Schulmeister: Virtuelle Universität – Virtuelles Lernen, Oldenbourg Verlag, München, 2001

Entwicklung eines E-Learning-Moduls für das Softwareprojektmanagement

Axel Buhl, André Knuth, Ute Werner
Fachhochschule Stralsund, FB Wirtschaft, Zur Schwedenschanze 15
axel.buhl | andre.knuth | ute.werner@fh-stralsund.de

Zusammenfassung

Für die Virtuelle Fachhochschule (VFH) wurde ein E-Learning-Modul entwickelt. Dabei wurde auf die Existenz verschiedener Lerntypen eingegangen, indem die Inhalte in getrennten, parallelen Lernsträngen umgesetzt wurden. So gibt es das textorientierte Lehrbuch mit Übungsaufgaben incl. kommentierten Lösungen, interaktive Animationen in spielerischer Form, Videovorlesungen für den audiovisuell orientierten Benutzer und eine komplexe Gruppenaufgabe, bei der über die Stoffvermittlung hinaus soziale Kompetenzen der Teilnehmer gefragt sind.

Bei der Entwicklung des Lernmoduls wurden ergonomische Anforderungen beachtet. Es konnte auf Erfahrungen, die sich in Evaluierungsergebnissen der VFH zeigten, aufgebaut werden. Die beschriebene Vorgehensweise bewährte sich in den bisherigen Praxistests.

1 Einleitung

Die Entwicklung eines E-Learning-Moduls für das Softwareprojektmanagement ist Teil des Projekts „Virtuelle Fachhochschule für Wirtschaft, Informatik und Technik“ (VFH). An der VFH werden telematische Fachhochschulstudiengänge in Medieninformatik und Wirtschaftsingenieurwesen entwickelt. Seit Herbst 2001 hat der reguläre Studienbetrieb im Bachelor-Studiengang Medieninformatik mit 166 Studierenden begonnen. Dieser wird von einem Teil der kooperierenden Projektpartner in einem bundesländerüberschreitenden Fachhochschulverbund durchgeführt. Projektpartner sind zwölf Fachhochschulen in Norddeutschland, die Universität der Bundeswehr Hamburg, die Medizinische Universität Lübeck sowie weitere Partner aus der Wirtschaft.

Zunächst werden die allgemeinen Anforderungen an Stoffauswahl und Stoffpräsentation in einem virtuellen Kurs aus unserer Sicht beschrieben. Danach erläutern wir unser Konzept zum Modul Softwareprojektmanagement und gehen dabei auf den Ablauf und die Struktur des Kurses ein. Abschließend berichten wir von ersten Erfahrungen aus dem Testeinsatz des Moduls in der Praxis.

2 Anforderungen an die Stoffauswahl und Stoffpräsentation in einem virtuellen Kurs

Lernen ist ein individueller Vorgang. Kursteilnehmer sprechen auf eine gegebene Lernform – z. B. eine Vorlesung – unterschiedlich gut an.

Aus unserer Erfahrung sind die folgenden Lernformen dazu geeignet, eine möglichst große Anzahl von Teilnehmern an einem Kurs zum Erfolg zu führen:

Die Teilnehmer brauchen die **Prüfungsanforderungen und Beispielaufgaben**, damit sie wissen, was von ihnen verlangt wird und zwar gleich zu Beginn der Lehrveranstaltung oder des Kurses.

Sie brauchen ein Lehrbuch (Skript) oder eine gute Literaturliste mit konkreten Hinweisen, wo was steht. Diese Literatur muss einem Studenten erlauben, sich den Stoff weitestgehend selbstständig zu erarbeiten.

Die meisten brauchen für schwierigere Abschnitte zusätzlich eine **Vorlesung**, Erklärungen einer Bezugsperson, ein Gesicht, dem sie vertrauen, ein gesprochenes Wort, das auch durch Gestik, Mimik und Stimme einprägsam ist in einem Situationsumfeld, das den Lernvorgang unterstützt. Die starke Persönlichkeit eines Professors in der spannungsvollen Atmosphäre eines gefüllten Hörsaals kann genau das bewirken.

Viele benötigen **Übungen**, etwas Anfassbares in einem handlungsorientierten Kontext. Diese pragmatischen Teilnehmer wollen Ergebnisse sehen und Situationen erleben. Nur dadurch prägt sich dauerhaft der geforderte Inhalt ein. An Hochschulen gibt es Labore, in denen Studenten etwas realisieren können oder man arbeitet mit regionalen Unternehmen zusammen, die einen „mitmachen“ lassen. Es gibt Praktika und Projektarbeiten.

Aus unserer Erfahrung benötigt ein großer Teil der Studenten auch die **Gruppe** als Umfeld zum Lernen. Kombiniert man die Gruppenarbeit mit projektorientierten Aufgabenstellungen, können gute Lernerfolge erzielt werden. Durch die eigene Dynamik, die sich in solchen Gruppen bildet, wird das Lernen unterstützt. Grundqualifikationen wie Teamfähigkeit werden herausgebildet.

3 Das Modul Softwareprojektmanagement

3.1 Konzeption und Struktur des Moduls

Die Evaluationsergebnisse der Pilotphase zeigten, dass die ersten Module nur wenige wirklich interaktive Inhalte aufwiesen [6]; Multimedia-Elemente hatten meist nur illustrierenden Charakter und waren wie Bilder in den Text integriert. Multimedia-Elemente wurden auch nicht dazu genutzt, um unterschiedliche Lerntypen ganz speziell zu unterstützen.

Wir wollen deshalb einen anderen Weg gehen: Der Lernstoff wird in parallelen Einheiten unter einer gemeinsamen Oberfläche präsentiert, in denen jeweils ein anderes Medium vorherrschend die Vermittlung der Inhalte übernimmt. Wir nennen diese Einheiten Lernstränge oder -pfade (vgl. Abbildung 1). Im Einzelnen sind das:

- Das **Lehrbuch**; es bildet die Basis des Kurses und ist neben der im virtuellen Modul integrierten HTML-Fassung auch als PDF abrufbar.
- Komplexe interaktive **Animationen**, die besonders schwierige Themen spielerisch vermitteln. Wir bezeichnen sie deshalb auch als Lernspiele.
- **Videovorlesungen**, die ebenfalls besonders abstrakte Abschnitte aufgreifen und aus einer anderen medialen Sicht darstellen.
- Ein **Gruppenspiel** mit der Möglichkeit, gemeinsam eine komplexe Aufgabe zu lösen und das bis dahin erfasste Wissen anzuwenden.

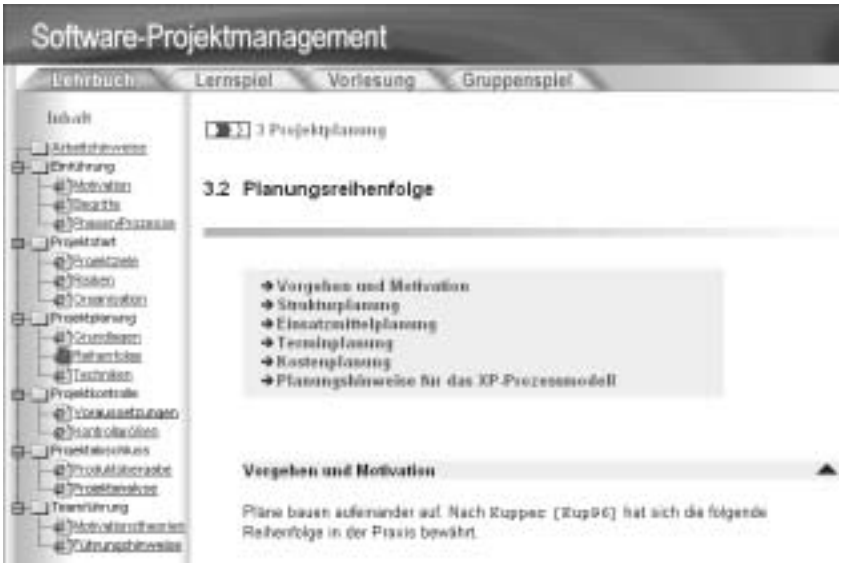


Abbildung 1: Das Modul Softwareprojektmanagement mit aufgeschlagenem Lehrbuch

Die Lernstränge sind zur asynchronen, selbständigen Bearbeitung geeignet. Man sollte das Studium mit dem Lehrbuch beginnen. Wann man Animationen oder Videovorlesungen zur Vertiefung nutzt, unterliegt technisch keiner Beschränkung. Jeder Student kann – entsprechend seines Typs – die Lernformen wählen, die ihm liegen und es ihm ermöglichen, den Lehrstoff möglichst vollständig zu erfassen. Das Gruppenspiel nimmt eine Sonderstellung ein; es erfordert teilweise synchrones Arbeiten und ist als Abschluss des Kurses gedacht.

Das gesamte Modul wird in den Lernraum der VFH eingestellt und kann so von jedem eingeschriebenen Studenten genutzt werden.

3.2 Das Lehrbuch

Das Lehrbuch kann, wie die anderen Lernpfade auch, durch die Auswahl des entsprechenden Karteikarten-Reiters aktiviert werden. Es zeigt den klassischen Aufbau aus Kapiteln und weiteren Untergliederungen, der Text wird durch Abbildungen ergänzt. Durch interaktive Elemente wie integrierte Aufgaben und Lösungen, ein

über Textlinks erreichbares Glossar und eine Suchfunktion wird eine Funktionalität erreicht, die die eines gedruckten Buches weit übertrifft.

Es wird neben der in Abbildung 1 dargestellten Ordnerstruktur zusätzlich eine zweite Navigation eingesetzt: sie bringt den Benutzer immer genau einen Abschnitt weiter – man kann also im Modul vor- und zurückblättern wie in einem Buch. Der Styleguide nennt eine solche Navigation den „sicheren Weg“ – es wird sichergestellt, dass der Lernende keine Abschnitte auslassen kann. Außerdem sind hier Links zu den Aufgaben und Literaturhinweisen des betreffenden Abschnitts integriert.

Übungsaufgaben

Aus der Evaluierung in der VFH wissen wir, dass einerseits automatisch auswertbare Aufgaben gewünscht, andererseits Multiple-Choice-Aufgaben auch kritisch gesehen werden („eher für Quizshow als zum Lernen geeignet“ [8, S. 63]). Da es keine einheitliche Lösung für dieses Dilemma geben kann, haben wir für jede Aufgabe die Frageform individuell gewählt, also Multiple Choice, wenn möglich und sinnvoll, ansonsten die freie Formulierung einer Antwort. Das wichtigste didaktische Mittel sind jedoch die kommentierten Lösungen, die dem Studenten über „Richtig“ bzw. „Falsch“ hinaus Begründungen und Lösungsweg vermitteln – und zwar für jede Frageform.

Der gesamte Ablauf – Fragestellung, Beantwortung, Lösung mit Kommentar – wird online im Lernraum realisiert und bedarf im Normalfall nicht mehr dem Eingreifen eines Tutors. Das setzt eine besonders sorgfältige Formulierung der Aufgaben voraus; evtl. verbliebene Missverständnisse können in Testeinsätzen erkannt und beseitigt werden.

3.3 Lernspiele

Konzept

Im Lernpfad **Lernspiele** soll durch komplexe, hochwertige Animationen der Stoff des Lehrbuches spielerisch aufgearbeitet und vertieft werden. Sie werden für Abschnitte eingesetzt, die erfahrungsgemäß den Lernern größere Schwierigkeiten bereiten, also z.B. solche mit hoher Komplexität und/oder Abstraktion.

Die einzelnen Animationen sind inhaltlich als Einheit zu betrachten und können dadurch unabhängig voneinander bearbeitet werden.

Ein Hilfsmittel zur Konzipierung einer multimedialen Animation stellt das Storyboard dar. Mit seiner Hilfe können Abläufe sequentiell bis auf Szenenebene beschrieben werden, es ist also ein chronologisches Drehbuch. Das Storyboard ist auch eine Richtlinie für die eigentliche Umsetzung der Animation und besteht grob aus den nachfolgenden Teilen (eine verbindlicher Standard liegt noch nicht vor):

1. Metadaten
    a. Bezeichnung, Szenennummer
2. Beschreibung
    a. Bewegung/Ablauf
        i. Sekundär
        ii. Primär
    b. Abhängigkeiten
    c. Skizze
3. Interaktivität
    a. Navigation
    b. Events
4. Audio
    a. Sprecher
    b. Geräusche/Effekte
    c. Musik
5. Weitere Dokumente

Umsetzung

Bei der Umsetzung der Animationen wurde besonderes Augenmerk auf die didaktischen Konzepte gelegt. Dabei wurde nach folgenden Punkten vorgegangen.

- Vom Einfachen zum Komplexen [1, S. 25]
- Falsches Wissen zu korrigieren ist schwieriger als neues Wissen aufnehmen
- Der kognitive Prozess soll aktiv, konstruktiv, kumulativ und zielgerichtet sein [2, S. 311]

Der erste Punkt unterstützt unser natürliches Lernverhalten und ist am einfachsten handhabbar.

Beim zweiten Punkt treten schon erhebliche Schwierigkeiten auf, denn hier muss das vermittelte Wissen verifiziert werden, um etwaige falsch aufgenommene Informationen zu erkennen und möglichst umfassend zu berichtigen. Nicht zu unterschätzen ist die Erfahrung, dass der Lernende selbst dann, wenn die falsche Information als solche erkannt wurde, nur sehr mühsam die richtig gestellte Information an- und aufnimmt [1, S. 30].

Der dritte Punkt besteht aus mehreren Unterpunkten. So bedeutet aktiv und im Falle des computergestützten Lernens interaktiv, dass der Lernende mehrere Events erfährt, bis er die anknüpfende Information so verarbeiten kann, dass diese in verständlicher Form gelernt wird. Konstruktivität wird dadurch erreicht, dass die durch mehrere Events angehäuften Informationen in Beziehung gestellt werden müssen, damit der Lernende einfache Informationen behält und komplexeres Material versteht. Kumulativ bedeutet, dass jeder Lernprozess auf vorhergehende Lernprozesse aufbaut, oder dass das Bekannte in einer solchen Weise genutzt wird, dass es bestimmt, was und wie viel gelernt wird. Wenn der Lernende ein Ziel vor Augen hat,

also der Zweck des Lernens gegenwärtig ist und damit Erwartungen einhergehen, ist der Lernprozess zielgerichtet und verläuft damit am erfolgreichsten [2, S. 311].

3.4 Videovorlesungen

Im Lernpfad **Vorlesungen** werden ausgewählte Themen des Lehrbuches als Videos aufbereitet – der Lehrstoff wird durch reale Personen vermittelt. Auch hier wurden Abschnitte gewählt, die neben der textlichen Beschreibung im Lehrbuchteil – aus Erfahrungen heraus – weiterer Erläuterungen bedürfen. Dazu wurden nicht einfach Vorlesungen des laufenden Studienbetriebes aufgezeichnet, sondern spezielle Drehbücher entwickelt. Es gibt Abschnitte, die moderiert werden, nachgespielte Situationen in einem angenommenen Projekt und auch Interviews mit Projektleitern aus realen Softwareunternehmen.

3.5 Das Gruppenspiel

Die bisher vorgestellten Lernpfade konnten asynchron abgearbeitet werden. Der Lernpfad **Gruppenspiel** erfordert zeitweise die synchrone Arbeit der Lerner. Diese finden sich in Gruppen zusammen, koordinieren ihre Arbeiten untereinander und fällen gemeinsame Entscheidungen. Als Aufgabe wurde ein aufzulegendes Softwareprojekt in einer Reederei gewählt. Um möglichst realistisch zu bleiben, gibt es Firmeninformationen und Interviewprotokolle, die auszuwerten sind. Das Gruppenspiel wird tutoriell unterstützt und ist in einen Zeitplan eingebettet. Der Tutor übernimmt dabei die Rolle des übergeordneten Managements, an das berichtet wird. Die Kommunikation der Gruppen wird über den Lernraum unterstützt (Gruppenseiten, Dateiaustausch, Chatrooms, Email-Listen). Als weitere Infrastruktur der VFH stehen auch Konferenzserver zur Verfügung. Das bereits erwähnte Gruppenspiel bildet den Abschluss der Ausbildung im Fach Softwareprojektmanagement. In den bisher durchgeführten Tests benötigten die Studenten 3 Tage von jeweils 9:00 bis 16:00 Uhr.

Der eigentlichen Aufgabe ist ein MS-Project-Tutorial vorgeschaltet. Anschließend lösen die Studenten eine erste Teilaufgabe in Einzelarbeit. Nach einer gemeinsamen Auswertung werden Gruppen gebildet, in denen dann der zweite Teil der Aufgabe bearbeitet wird.

Alle Textinhalte, MS-Project-Tutorial, Firmeninformationen und ausführliche Aufgabenstellung werden in derselben Weise zugänglich gemacht wie beim Lehrbuch bereits beschrieben.

Um die Gruppenarbeit durchzuführen, müssen neben der Aufgabenbeschreibung und den erforderlichen Lösungshinweisen Kommunikationskanäle eingerichtet werden, die eine Zusammenarbeit der Studenten untereinander und mit den Betreuern erlauben.

Die Kommunikation baut auf den Lernraum auf. Dessen Funktionalitäten (Chat, Group Pages, Dokumentenmanagement) liefern die erforderliche Plattform, um untereinander bzw. mit den Betreuern zu kommunizieren.

4 Entwicklungsprozess eines Lernmoduls – Theorie und Praxis

4.1 Wissenschaftliche Begleitung der Entwicklung von Lernmodulen in der VFH

Beim Aufbau der VFH ist die Arbeit der Entwickler eingebettet in die Vorarbeit der Ergonomen und Didaktiker und die Nachbearbeitung der Evaluierungsgruppe. Dazu wurde ein Qualitätssicherungsprozess in das Projekt eingeführt, bei dem man von der ISO 13407 (1999) und dem „Prüfhandbuch Gebrauchstauglichkeit“ der DATech ausging. Während dieses Prozesses entstanden mehrere Dokumente, die einen einheitlichen Qualitätsstandard ermöglichen. Die genannte ISO-Norm wurde durch eine Abfolge aus Datenerhebung, Validierung und weiterer Verfeinerung der Dokumente umgesetzt. So wurden bereits in Testeinsätzen erster Module – lange vor dem Start der eigentlichen VFH – Evaluierungen durchgeführt [6].

Die wichtigsten Dokumente seien hier kurz beschrieben:

- Das Ergonomiehandbuch: Die Hinweise dieses Dokuments beginnen bei methodischen Aspekten wie der Verwendung eines Prozessmodells und einer iterativen Herangehensweise, setzen sich fort mit konzeptionellen Fragen und reichen bis zu konkreten, an Beispielen erläuterten Hinweisen didaktischer, gestalterischer und funktioneller Art [4].
- Der Styleguide als verbindlicher Produktionsleitfaden, der Anforderungen aus dem Bereich Didaktik, Technik, Formate und Ergonomie enthält. Beispiele sind die Festlegung bestimmter Webbrowser und Einstellungen, mit denen ein Modul funktionieren muss, konkrete Anforderungen an zu erstellende Dokumente in den Planungsphasen oder die Einhaltung bestimmter Schriftgrößen [5].
- Der Prozessleitfaden; hier werden die im Ergonomiehandbuch allgemein dargestellten Anforderungen an den Entwicklungsprozess konkretisiert.

Bei der Evaluation des Einsatzes von Studienmodulen im Test- oder Regelstudienbetrieb wird das Qualitätssicherungskonzept einer Erfolgskontrolle unterzogen. Gleichzeitig werden daraus Hinweise für die weitere Arbeit abgeleitet. Die oben genannten Dokumente werden im Laufe des Projektes ständig überarbeitet und angepasst.

4.2 Der Entwicklungsprozess des Moduls Softwareprojektmanagement

Ausgangspunkt für die Gestaltung des Entwicklungsprozesses ist der Prozessleitfaden. Auf dessen Grundlage haben wir einen inkrementellen, iterativen Entwicklungsprozess entworfen. Es sind mehrere Releases vorgesehen; das Release 3, welches hier als Beispiel stehen soll, ist wie folgt definiert:

Release 3: Lehrbuch HTML Abschluss, Lehrbuch PDF Abschluss, Gruppenspiel Abschluss, 2 Animationen Flash Vers. 2 und 2 Animationen Flash Vers. 1, 2 Vorle-

sungen Video Vers.1, Einsatz an der FH Stralsund im Sommersemester mit ca. 80 Studenten, Termin Februar 2003.

Die Inhalte der Releases zeigen den typischen Aufbau von Multimediaprojekten. Es ist keine reine Programmentwicklung. Es werden HTML- und PDF-Dateien erzeugt, es entstehen Flash-Dateien und Videoformate – also Dokumente, Multimedia-Elemente und auch Quellcode. Diese unterschiedlichen Komponenten verlangen einen sehr flexiblen Entwicklungsprozess. Die Videoproduktion erfordert eigentlich ein Wasserfallmodell. Bevor Schauspieler und Sprecher engagiert werden, müssen die Drehbücher erstellt und jeder Dialog genau ausformuliert werden. Prototyping schließt sich aus. Eine Wiederholung ist nur in seltenen Fällen möglich. Die Navigation und die Textgestaltung in HTML-Dokumenten erfordern dagegen meistens etliche Varianten, bevor man ein befriedigendes Ergebnis vorlegen kann. Auch das Erstellen von komplexen interaktiven Animationen erfordert die Ausarbeitung detaillierter Storyboards für den Handlungsablauf – insbesondere dann, wenn der Auftrag an einen externen Auftragnehmer geht und Kosten sowie Entwicklungszeit festzulegen sind. Auf der anderen Seite wird eine Animation auch ganz wesentlich durch das GUI geprägt, das durch Vorstellungen der späteren Anwender bestimmt wird. Diese Vorstellungen konkretisieren sich aber erst nach der Erstellung von Szenen. Ein frühes Prototyping ist daher sehr empfehlenswert.

Eingebettet in den Prozess sind externe und interne Reviews. Die externen Reviews ermöglichen eine Qualitätssicherung für alle Module der VFH, die an ganz unterschiedlichen Orten und Zeiten entwickelt werden. Die internen Reviews werden an der Fachhochschule organisiert. Der Kreis der eingeladenen Gäste variiert je nach Fertigstellungsgrad der Einheit. Beschränkt sich der Kreis anfangs auf die Gruppe der Entwickler und Betreuer, so werden später auch weitere Kollegen und Studenten herangezogen. Einheiten, die einen befriedigenden Entwicklungsstand erreicht haben, werden dann in der Präsenzlehre der FH eingesetzt und evaluiert. Für die internen Reviews werden Fragenkataloge eingesetzt. Zur Verdeutlichung sollen einige Fragen zur Prüfung einer Animation vorgestellt werden:

- Wird am Anfang eine Fokussierung auf das Thema gegeben – hat der Lerner ein eindeutiges Lernziel vor Augen?
- Ist der Beginn der Animation konzentrationsfördernd?
- Gibt die Umsetzung der Animation tatsächlich den Charakter des Lehrinhalts wieder – gibt es einen Bezug zwischen dem Thema und den handelnden Einheiten?
- Wird Spannung aufgebaut – durch Interaktion oder gestalterische Mittel (Bewegung und Farben)?
- Wird das Thema wirklich vertieft – geht die Darstellung über das Lehrbuch hinaus und wird dabei das Wesentliche herausgearbeitet?
- Wird am Ende eine eindeutige Auswertung und Zusammenfassung gegeben?

5 Testeinsatz und eigene Evaluierung an der FH Stralsund

Auch wenn unser Modul noch nicht in der VFH zum Einsatz kam, so verfügen wir an der FH Stralsund bereits über eigene Praxiserfahrungen.

5.1 Durchführung

Wir haben das erste Release des Moduls im Rahmen der regulären Lehrveranstaltungen an der FH Stralsund „im Feldversuch“ getestet und dabei auch durch einen Fragebogen evaluiert. Die Bedingungen unterschieden sich in einigen wesentlichen Punkten von einem Einsatz in der VFH:

1. Vom Modul lag nur das Lehrbuch vor, die Vorteile der verschiedenen Lernstränge kamen also noch nicht zum Tragen.

2. Dieser eine virtuelle Kurs konkurrierte mit allen übrigen in Präsenz abgehaltenen Fächern um die Zeit und Aufmerksamkeit der Studenten.

3. Die Initiative, das Fach virtuell zu studieren, ging nicht von den Studenten aus, wie man es in der VFH voraussetzen kann, sondern vom Professor. Ein wichtiger Punkt, denn die Studenten schätzten ihre eigene Motivation häufig als nicht ausreichend für das virtuelle Lernen ein.

Aus den Evaluationen an der FH wussten wir bereits, dass trotz freier Zeiteinteilung beim virtuellen Lernen eine gewisse zeitliche Taktung gewünscht wird. Deshalb stellten wir einen orientierenden Zeitplan, welches Kapitel in welcher Woche zu bearbeiten sei, zur Verfügung. Darüber hinaus wurde das Angebot durch eine wöchentliche Chat-Konsultation ergänzt, die sich thematisch ebenfalls an diesem Zeitplan orientierte.

5.2 Evaluierung

Sehr wichtig waren den Studenten Verlässlichkeit in technischen und organisatorischen Dingen – wenn also der Server ständig ausfällt, Termine für Konsultationen, Chats usw. oft und kurzfristig verschoben werden, nützt das beste Lernmodul nichts. Die Studenten verlieren die Lust und wohl auch das Vertrauen in das virtuelle Lernen.

Dass **Übungsaufgaben** unverzichtbar sind, wurde bereits ausgeführt. Besonders bei komplizierten und mit Berechnungen verbundenen Sachverhalten kam es gut an, wenn theoretische Erläuterungen mit komplett durchgerechneten Beispielen ergänzt und ähnliche Übungsaufgaben angeboten wurden. Besonders positiv wurde unsere Vorgehensweise, dem Studenten nach Eingabe einer Lösung nicht nur „falsch“ oder „richtig“ zurückzugeben, sondern auch den kompletten Lösungsweg zu erklären, bewertet – eher wünschte man sich die Antworten noch ausführlicher.

Zum Komplex der Übungsaufgaben wurde als weiterer wichtiger Punkt eine eindeutige Formulierung der Fragen genannt. Eine solche Forderung scheint selbstverständlich, doch es kommt immer wieder vor, dass in Feedbackangeboten (Mail,

Chat-Konsultationen) meist nicht die erwarteten Fachfragen gestellt werden. Es wird um Erklärungen gebeten, wie eine Aufgabe gemeint sei. Im virtuellen Raum ist der Aufwand, ein Feedback zu erlangen, im Allgemeinen höher – deshalb ist dieser Punkt noch wichtiger als z.B. in einem Präsenzseminar. Man erleichtert also nicht nur den Studenten die Arbeit, sondern auch den Betreuern.

Ein Vorteil des **Chats** ist die Auflösung von Hierarchien und damit der Abbau von Hemmungen. Ausgelöst wird dieser Effekt durch die Reduktion aller Beteiligten auf den jeweiligen Namen oder „nickname“ und der daraus resultierenden Anonymität. Aus den Ergebnissen des Evaluationsberichtes geht weiterhin hervor, dass Chats aus der Sicht der Tutoren als „unendlich langsam“ beurteilt werden und die Studenten teilweise den Chat nur unkonzentriert mitverfolgen [8, S. 40]. Beide Aussagen können wir aus eigenen Erfahrungen bestätigen, jedoch beurteilen Studenten die Geschwindigkeit eines Chats oft anders: Auf die Frage, was die meisten Probleme verursacht hatte, erhielt der Punkt „Fragestellungen erforderten komplexe Antworten, aber zu wenig Zeit, um diese einzutippen“ die meisten Nennungen. Das zeigt, dass sich beide Seiten – Tutoren und Studenten – auf die Besonderheiten dieser Kommunikationsform noch stärker einstellen müssen. In [8, S. 130] wird empfohlen, klare Rahmenbedingungen für einen Chat zu schaffen, wie Termin, Dauer und Thema. Das hatten wir im Prinzip bereits so gehandhabt, es lassen sich aber sicher noch Verbesserungen vornehmen. Als vorteilhaft für nicht Anwesende oder einfach zum späteren Nachlesen stellte sich die Aufzeichnung und Archivierung von Chats heraus.

Das bereits erwähnte **Gruppenspiel** ist der am längsten existierende Teil unseres Lernmoduls – wir setzten es in diesem Jahr bereits zum dritten Mal ein. Diesmal waren unsere Studenten durch die virtuelle Nutzung des Lehrbuchs bereits mit den Rahmenbedingungen und dem Lernraum vertraut. Das brachte eine deutliche zeitliche Entspannung, die Studenten konnten sich gut auf das inhaltliche Arbeiten konzentrieren. Die Hilfestellung durch die Lehrkräfte konnte von Arbeitsphase zu Arbeitsphase reduziert werden: Direkte persönliche Unterstützung am Rechner während des MS-Project-Tutorials, vorrangig per Chat während der Einzelarbeit und nur noch sporadisch in der Gruppenarbeitsphase.

6 Fazit

Das Erstellen einer E-Learning-Einheit ist eine nichttriviale Angelegenheit.

Für den Professor ist es sehr zeitraubend, da er zwar über das Fachwissen verfügt, sich aber in die Problematik der mediengerechten Aufarbeitung erst einarbeiten muss. In Präsenzveranstaltungen kann man auf ein bestimmtes Verhalten von Lernern unmittelbar reagieren, z. B. nachlassende Konzentration. Auch unterschiedliche Lerntypen können erkannt werden. Man kann seine Methoden entsprechend abstimmen.

Für die virtuelle Lehre müssen solchen Überlegungen schon vorher gemacht werden. Eine Lösungsmöglichkeit ist das Angebot von unterschiedlichen Lernsträn-

gen mit der Konzentration von bestimmten Medien, die einen ausgemachten Lerntyp unterstützen.

Inkrementelle, iterative Prozessmodelle sind gut geeignet, um ein E-Learning-Projekt zu realisieren. Reviews müssen in regelmäßigen Abständen durchgeführt werden. Das größte Problem ist die Erstellung guter Storyboards, Drehbücher und Regieanweisungen. Aus diesem Dilemma kommt man nur heraus, wenn man kleine Inkremente wählt, die fertige Einheiten sofort in die Lehre übernimmt und evaluiert, um sie dann schrittweise zu verbessern.

Literatur

1. Mietzel, G.: "Pädagogische Psychologie des Lernens und Lehrens", 6. korrigierte Auflage, Hogrefe Verlag, 2000; S. 25, 30
2. Shuell, T. J.: “The role of the student in learning from instruction.”, Contemporary Educational Psychology 1988, S. 311
3. Amy Jo Kim: Community Building, Galileo Press, 2001
4. Hartwig, R.; Triebe, J.K.; Herczeg, M.: Ergonomie-Handbuch zur Gestaltung virtueller Lerneinheiten - Version 1.0.4. Medizinische Universität zu Lübeck - Institut für Multimediale und Interaktive Systeme, 2002
5. Hartwig, R.; Triebe, J.K.; Herczeg, M.: Styleguide - Richtlinien zur Qualitätssicherung bei der Realisierung von Studienmodulen im Projekt VFH. Medizinische Universität zu Lübeck - Institut für Multimediale und Interaktive Systeme, 2002
6. Hartwig, R.; Triebe, J.K.; Herczeg, M.: Software-ergonomische Evaluation im Kontext der Entwicklung multimedialer Lernmodule für die virtuelle Lehre. Erschienen in: Herczeg, M.; Prinz, W.; Oberquelle, H.(Hrsg.): Mensch & Computer 2002: Vom interaktiven Werkzeug zu kooperativen Arbeits- und Lernwelten. Stuttgart: B.G. Teubner, 2002, S. 313-322
7. Hinze, U.; Blakowski, G.: Virtuelle Organisation und Neue Medien 2002, Tagungsband, Josef Eul Verlag, 2002, S. 162 – 187
8. Arnold, P., Kilian, L., Thillosen, A.: Didaktisch-methodische Evaluation des Regelstudienbetriebs der Virtuellen Fachhochschule für Technik, Informatik und Wirtschaft (VFH), Wintersemester 2001 / 2002; Projektinterner Evaluationsbericht; S. 134-137.
9. Triebe, J.: Ergonomische Gestaltung von Lerneinheiten. In: Bildung online – Die Virtuelle Fachhochschule. Symposium 2002, Dokumentation.

[10] Sawhney, M.: Entwicklung eines Vorgehensmodells für die Multimedia-Anwendungsentwicklung am Beispiel eines Informations- und Orientierungssystem für eine Universität, Diplomarbeit, Universität Osnabrück, FB Wirtschaftsinformatik, 1995

[11] Hitzges, A. und Laich, U.: Projektmanagement bei der Entwicklung multimedialer Anwendungen. Technical Report, Fraunhofer Institut für Arbeitswissenschaft und Organisation, Stuttgart, 1995

MuSofT: Multimedia in der Softwaretechnik

Klaus Alfert, Ernst-Erich Doberkat
Lehrstuhl für Software-Technologie, Fachbereich Informatik, Universität Dortmund,
alfert@ls10.de, eed@ls10.de

Gregor Engels, Marc Lohmann, Johannes Magenheim
Fachbereich Mathematik/Informatik, Universität Paderborn,
engels@upb.de, macke@upb.de, jsm@uni-paderborn.de

Andy Schürr
Fachbereich 18, TU Darmstadt,
schuerr@es.tu-darmstadt.de

Zusammenfassung

Im BMBF-Verbundprojekt MuSofT werden multimediale Lehrmaterialien für die Lehre der Softwaretechnik entwickelt. In diesem Papier stellen wir die Bemühungen innerhalb von MuSofT vor, eine qualitativ hochwertige und nachhaltige Entwicklung von Lernobjekten zu realisieren. Wir legen den Fokus dabei auf die didaktischen Grundannahmen, die inhaltliche und stilistische Abstimmung zwischen den Materialien sowie die gleichförmige Beschreibung der Materialien durch Metadaten, die eine effektive Recherche des Materials innerhalb des MuSofT-Portals ermöglichen.

1 Einleitung

Das Projekt *MuSofT - Multimedia in der SoftwareTechnik* hat sich zum Ziel gesetzt, multimediale Lehreinheiten zur Unterstützung der Präsenzlehre in der Softwaretechnik zu entwickeln. Das Projekt wird im Rahmen des Programmes „Neue Medien in der Bildung“ des BMBF vom 01.03.2001 bis zum 31.12.2003 gefördert. Die Projektpartner sind (von Nord nach Süd) die FH Lübeck (Frau Prof. Seehusen), Uni Magdeburg (Prof. Saake), Uni Paderborn (Prof. Engels, Prof. Magenheim), Uni Dortmund (Prof. Doberkat), Uni Siegen (Prof. Kelter), TU Darmstadt (Prof. Schürr) sowie die Uni Stuttgart (Prof. Schmidt); die Konsortialleitung liegt gemeinschaftlich bei Prof. Doberkat und Prof. Engels. Das Themenspektrum in MuSofT lässt sich in drei große Bereiche, zu welchen die einzelnen Projektpartner Lehrmaterialien mit unterschiedlichen Themenschwerpunkten entwickeln, gliedern [4]:

1. Vorgehensweisen in Anforderungsanalyse, Entwurf und Realisierung
2. Prozess- und Projektmanagement in der Software-Entwicklung
3. Softwaretechnische Fachdidaktik

Die in MuSofT betrachteten Themengebiete decken einen großen Teil der Lehre im Grundstudium ab, ohne dabei speziellere Themen im Hauptstudium zu vernachlässigen. Ein wesentliches Anliegen besteht darin, ausgewählte Lehrinhalte durch multimediale Gestaltung besser, d.h. anschaulicher und themengerechter präsentieren zu können. Dies bedeutet aber auch, dass die Erarbeitung vollständiger multimedialer Vorlesungen nicht das vordringliche Ziel von MuSofT ist, sondern ebenso auch die kleinerer in sich abgeschlossener Einheiten, die bestimmte Teilaspekte der Softwaretechnik behandeln. Im Folgenden verwenden wir (in Anlehnung an der LOM-Standard, siehe Abschnitt 3) den Begriff *Lernobjekt* als Oberbegriff für Einheiten von Lehrmaterialien auf einer beliebigen Granularitätsstufe.

Um die in MuSofT an unterschiedlichen Standorten entwickelten Lernobjekte zur Realisierung einer Vorlesung einsetzen zu können, ist eine weitere wichtige Zielsetzung des MuSofT-Projektes, die unterschiedlichen Lehreinheiten aufeinander abzustimmen. Diese inhaltliche Abstimmung erfordert übergreifende Aktivitäten, die maßgeblich zur Qualitätsförderung in MuSofT beitragen. Hierzu wurden zu Beginn des Projekts Koordinationsteams eingerichtet, welche über die einzelnen Standorte hinweg tätig sind. Für die Belange dieses Papieres sind die folgenden Koordinationsteams mit ihren teilprojektübergreifenden Aktivitäten zuständig:

KT 1 für die Erarbeitung von Richtlinien für die didaktische Konzeption der Lernobjekte,

KT 2 für die inhaltliche und stilistische Abstimmung von Lernobjekten, die u.a. die Verwendung durchgängiger Fallbeispiele, die Einigung auf verwendete Modellierungs- und Programmiersprachen sowie die zugehörigen Richtlinien umfasst, und

KT 4 für die Bereitstellung eines Internetportals, über welches die erstellten Lernobjekte und die dazugehörigen Werkzeuge angeboten werden.

Im Folgenden sollen die Aktivitäten und Ergebnisse der unterschiedlichen Koordinationsteams detailliert dargestellt werden. In Abschnitt 2 betrachten wir die didaktischen Konzeptionen, gefolgt von der Diskussion der Abstimmungen zwischen den einzelnen Teilprojekten (Abschnitt 3), die zu einheitlichen Metadaten für die Lernobjekte geführt hat. Diese Metadaten sind in das MuSofT-Portal eingeflossen (Abschnitt 4).

2 Didaktische Grundüberlegungen

Die im Projekt MuSofT entwickelten Lernobjekte zur Softwaretechnik sind primär zur Unterstützung der universitären Präsenzlehre bestimmt. Um ihre nachhaltige Nutzung in einer Vielzahl von unterschiedlichen Einsatzszenarien zu sichern, müssen didaktische Überlegungen zu ihrer Gestaltung unterschiedlichsten Einsatzbedingungen gerecht werden und von multivarianten Lernumgebungen ausgehen. Die Lernobjekte sollen sowohl als multimediale Elemente in eine traditionelle Vorlesung

integrierbar sein als auch zur Unterstützung des Übungsbetriebs genutzt werden können. Sie sollen der individuellen Nachbereitung des Stoffes durch Studierende dienen und darüber hinaus teilweise auch die kooperative Erarbeitung des Fachgebiets in Form des Selbststudiums ermöglichen. Außerdem können trotz gleicher Themen die inhaltlichen Schwerpunkte von Lehrveranstaltungen und das Anforderungsniveau an die Studierenden in Abhängigkeit vom jeweiligen Studiengang erheblich variieren.

Die Lernobjekte können diesen zentralen Rahmenbedingungen nur gerecht werden, wenn sie nicht in der Form von monolithischen, komplexen Einheiten eines nicht separierbaren Lernstoffes angeboten werden. Die angestrebte Feingranularität der Lernobjekte erlaubt es Lehrenden, sich Teile des Studienmaterials entsprechend ihren speziellen Lehrbedürfnissen und inhaltlichen Schwerpunkten zu neuen Lernobjekten, ggf. ergänzt mit Eigenentwicklungen, zusammenzustellen. Auf diese Weise kann eine vielfältige, sich weiterentwickelnde Vernetzung der einzelnen Lernobjekte entstehen, die durch weitere neu zu entwickelnde Materialien seitens der Nutzer ergänzt und aktualisiert werden kann.

Angesichts des didaktisch-methodischen Postulats der „flexiblen Einsetzbarkeit und nachhaltigen Nutzung" erschien es den Projektpartnern wenig hilfreich, verbindliche Vorgaben für die Gestaltung der Lernobjekte zu vereinbaren, weder für die didaktische Konzeption noch für das Layout. Da es in MuSofT im Gegensatz etwa zur Virtuellen Fachhochschule (VFH) keinen institutionellen Überbau gibt, soll die Autonomie der Lehrenden und ihrer Hochschulen gewahrt bleiben. Aus dem selbigen Grund haben wir von der Vorgabe eines MuSofT-spezifischen Lernmanagementsystem, wie z.B. Blackboard [2], Clix-Campus [3] oder WebCT [11], Abstand genommen.

Unabhängig von dem Anspruch, die in MuSofT entwickelten Materialien flexibel einsetzen zu können, wurde die Vereinbarung getroffen, die Lernobjekte auf der Basis einiger durchgängiger Fallstudien anzulegen. Mit dieser didaktisch-methodischen Grundsatzentscheidung soll eine konsequente Anwendungsorientierung der vermittelten Fachinhalte gewährleistet werden. Schließlich wird damit zusammen mit der oben beschriebenen Feingranularität die Kombinierbarkeit der Lernobjekte erhöht.

Wir möchten anmerken, dass die beschriebenen didaktischen Überlegungen nur zentrale Fragestellungen berühren und nicht alle didaktischen Aspekte en detail betrachten.

3 Inhaltliche und stilistische Abstimmung von Lernobjekten

Die inhaltliche und stilistische Abstimmung von Lernobjekten stellt die Grundlagen dafür bereit, dass sich Lernobjekte für neue Zielgruppen durch Kombination und Adaption bereits vorhandener Lernobjekte erzeugen lassen und dass die Anforde-

rungen an ein gemeinsames MuSofT-Portal genauer festgelegt werden können. Um diesem Anspruch gerecht zu werden, wurden folgende drei Primärziele formuliert:

1 die Nutzung von Lernobjekten in anderen Lernobjekten verschiedener Teilprojekte und die gleichzeitige Vermeidung von Redundanzen, die durch Doppelentwicklungen in verschiedenen Teilprojekten entstehen können,
2 die Identifikation fehlender Lernobjekte zur Vermittlung von Basiswissen, das von den erstellten Lernobjekten vorausgesetzt wird, und die Vergabe entsprechender Aufträge an Teilprojekte,
3 die Erstellung neuer Lernobjekte durch die Wiederverwendung vorhandener Objekte bzw. die Anpassung bereits erstellter Lernobjekte an neue Zielgruppen.

Aus diesen Primärzielen lässt sich ableiten, dass die inhaltliche und stilistische Abstimmung von Lernobjekten bzw. ihrer Bestandteile notwendig ist und welche Punkte dabei zu berücksichtigen sind. Auf inhaltlicher Ebene spielen die Dokumentation verschiedener Arten von Beziehungen zwischen einzelnen Lernobjekten, die Auswahl durchgängiger Fallstudien, die Festlegung der verwendeten Modellierungs- und Programmiersprachen sowie die einheitliche Beschreibung aller erstellten Lernobjekte eine herausragende Rolle. Aus der großen Menge der Methoden und Techniken der Softwaretechnik haben wir uns in MuSofT auf die obligatorische Verwendung von Java und UML geeinigt, als Entwicklungsprozess wird der Unified Process betrachtet, gemeinsame Fallstudien diskutieren Themen aus der Logistik als Grundlage (Lagerverwaltung, Kommissionierung, Speditionswesen). Auf der stilistischen Ebene geht es unter anderem darum, Modellierungs- und Programmierkonventionen festzulegen sowie die Menge der alternativ eingesetzten Modellierungs- und Programmierwerkzeuge einzuschränken.

Neben diesen inhaltlichen und stilistischen Abstimmungen ist es für die Abarbeitung dieses Aufgabenkatalogs notwendig, dass zu allen im Projekt erstellten Lernobjekten einheitliche Beschreibungen in Form so genannter Metadaten bereitgestellt werden. Hierzu wurde auf den IEEE LOM-Standard Learning Objects Metadata [7] zurückgegriffen, der ein konzeptionelles Datenschema vorgibt, welches die Struktur von Metadaten für Lernobjekte beschreibt. Die Verwendung eines Metadatenstandards erlaubt nicht nur eine einheitliche Beschreibung der Lernobjekte innerhalb von MuSofT, sondern erlaubt auch eine Interpretation der verwendeten Metadaten über die Projektgrenzen hinaus. Der Grund für die Verwendung des LOM-Standards ist zum einen seine weite Verbreitung. Zum anderen wird der LOM-Standard nahezu unverändert in verschiedene Standardisierungsbemühungen, wie z.B. dem IMS Global Learning Consortium [9], SCORM [1] oder ARIADNE [5], integriert.

Obwohl LOM aus den oben genannten Gründen der für unsere Zwecke am besten geeignete Standard für die Beschreibung von Lernobjekten ist, war sein Einsatz im MuSoft-Projekt mit umfangreichen Vorarbeiten verbunden. So erschien es uns vor allem unrealistisch, alle von LOM vorgeschlagenen Metaattribute (etwa 60 Attribute, die in neun Kategorien unterteilt sind) für die Dokumentation unserer Lern-

objekte einzusetzen und zudem die tatsächliche Verwendung der LOM-Attribute ohne weitere Richtlinien den einzelnen Teilprojekten zu überlassen. Ein solcher ungeregelter Einsatz von LOM würde möglicherweise zu weit voneinander abweichenden Metabeschreibungen einzelner Lernobjekte führen und damit die oben aufgeführten Zielsetzungen konterkarieren.

Deshalb wurde der LOM-Standard wie folgt an die Bedürfnisse von MuSoft angepasst:

- von den neun Attributkategorien werden zunächst nur die fogenden fünf gekürzt für die Beschreibung von Lernobjekten verwendet:

    1. *General* für allgemeine Informationen,
    2. *Technical* für technische Eigenschaften,
    3. *Educational* für didaktische Eigenschaften,
    4. *Relation* für die Beschreibung von Beziehungen zwischen Lernobjekten,
    5. *Classification* für Klassifikationshierarchien.

  Dies reduziert die Zahl der Attribute auf 19. Die Bedeutung dieser Teilmenge von Attributen wird durch zusätzliche Erläuterungen und Einschränkungen präzisiert.

- die im LOM-Standard vorgesehene vierstufige Hierarchisierung von Lernobjekten wird wie folgt präzisiert: auf der obersten Ebene gibt es Lehreinheiten (LE), die aus wiederverwendbaren Lernmodulen (LM) aufgebaut sind. Lernmodule bestehen ihrerseits aus so genannten Gruppenobjekten (GO), die atomare Medienobjekte (MO) enthalten. Beispielsweise entspricht eine Vorlesung dabei einer LE, die sich in thematische Blöcken zergliedert (mehrere LM). Diese können wiederum in einzelne Abschnitte zerlegt werden (GO), die sich aus Medienobjekten (z.B. Folien, Animationen, Videos, ...) zusammensetzen. Es ist aber nicht notwendig, dass alle vier Ebenen durchgängig benutzt werden.

In Abbildund 1 sehen wir einen Ausschnitt der Metadaten eines fiktiven Medienobjektes zur Programmierung in Java.

Die Richtlinien für die didaktische Konzeption der Lernobjekte in MuSofT legen vier didaktische Kriterien fest, die für die Lernobjekt dokumentiert werden müssen. Im Folgenden charakterisieren wir diese Kriterien anhand derjenigen LOM-Attribute der Kategorie Educational, die auch im MuSofT-Portal benutzt werden (s. Abb. 1):

Abbildung 1:
Metadaten eines fiktiven Lernobjektes, wie sie sich im MuSofT-Portal darstellen

Leitbilder definieren den Kontext für den Einsatz des Lernobjektes. Hierzu zählen Informationen über den Studiengang (z.B. Uni/FH, Nebenfach, Hauptfach) im Attribut Kontext, technische Rahmenbedingungen (Betriebssystem, Browser, etc.) im Attribut Anforderungen, sowie Lernvoraussetzungen (z.B. in Form von Relationen zu anderen Lernobjekten) in der Rubrik „Beziehungen zu anderen Lernobjekten".

Lernszenarien geben genauere Hinweise zum Einsatz des Lernobjektes. Hierzu zählen Anspruch, Abstraktionsniveau, Medientypen sowie Angaben zur Interaktivität des Lernobjektes aus Sicht des Lernenden, die als Attribute unter der Rubrik „Didaktische Angaben" zu finden sind.

Lernziele erläutern, was Lernende anhand des Lernobjektes lernen sollen (Attribut Lernziele).

Benutzungshinweise geben Hinweise für den Einsatz des Materials und des didaktisch-methodischen Arrangements aus der Sicht des Autors (Attribut Benutzungshinweise).

Als besonders schwierig bei der Anpassung des LOM-Standards für MuSofT erwies sich zum einen die Festlegung einer festen Hierarchie von Lernobjekten sowie zum anderen die Auswahl und Beschreibung der Metaattribute der Kategorie Educational. So ist es fraglich, ob eine vierstufige Hierarchie mit genau festgelegten Rollen der Lernobjekte auf den einzelnen Ebenen immer ausreichend ist oder ob man nicht lieber die Erstellung beliebig tiefer Hierarchien unterstützen sollte. Unbehagen löst auch die Tatsache aus, dass alle Metaattribute für die Beschreibung von Lernobjekten auf allen Hierarchiestufen zugelassen sind. Technische Attribute, wie etwa das Format eines Lernobjektes, scheinen auf den oberen Ebenen wenig Sinn zu machen, während hingegen didaktische Attribute, die z.B. den eingesetzten Studiengang eines Lernobjektes beschreiben, auf den unteren Ebenen fraglich erscheinen. Zusätzlich hat sich gezeigt, dass die didaktischen Attribute oft eine unklare Semantik besitzen.

Wir erwarten, mit der skizzierten Anpassung des LOM-Standards einen gangbaren Weg gefunden zu haben, der den Aufwand für die Erstellung von Metadaten nicht in unrealistische Höhen treibt und trotzdem die für die (Wieder-) Verwendung von Lernobjekten benötigten Informationen bereitstellt. Hier sind unsere Erfahrungen mit dem flächendeckenden Einsatz des MuSofT-Portal abzuwarten, das die Verwaltung entsprechender Metadaten bereits unterstützt.

4 Das MuSofT-Portal

Das webbasierte MuSofT-Portal (http://www.musoft.org) ermöglicht die Archivierung und den Austausch der im Rahmen von MuSofT erstellten Lehreinheiten und den dazugehörigen Werkzeugen. Das MuSofT-Portal bietet den Mitgliedern des Projektes MuSofT die Möglichkeit zur Distribution ihrer erstellten Lernobjekte. Das Portal ist öffentlich zugänglich, so dass nicht nur Projektinterne Zugriff auf die im Portal abgelegten Daten haben. Wir hoffen, dass so die in MuSofT erstellten multimedialen Materialien eine größere Verbreitung finden können. Auch externen Nutzern wird die Möglichkeit geboten, selber Materialien auf dem MuSofT-Server abzulegen, so dass auf dem MuSofT-Portal eine Sammlung von Materialien für die gesamte Bandbreite der softwaretechnischen Lehre entstehen kann.

4.1 Technische Realisierung

Das MuSofT-Portal kann man als eine spezielle Variante eines Content-Management-Systems (CMS) auffassen, das als Inhalte die multimedialen Lernobjekte zusammen mit Metadaten für die Recherche verwaltet. Die verfügbaren CMS haben allerdings unsere Anforderungen nicht erfüllt. Klassischerweise unterscheiden CMS zwischen einer Entwicklungssicht mit und einer Präsentationssicht ohne (ausgefeilte) Recherchemöglichkeiten, dies widerspricht aber der Arbeitsweise im MuSofT-Portal. Als schwierig hat sich auch die Behandlung der Metadaten gezeigt, da wir nicht nur eine vorgegebene Menge von Metaattributen benötigen. Da zusätzlich

der Satz an Metaattributen und ihrer Wertemengen sich aufgrund unserer künftigen Erfahrungen mit dem Portal ändern kann, brauchen wir eine leichte Änderbarkeit der Metaattribute und ihrer Ausprägungen im laufendem Betrieb. Schlussendlich sind für eine nutzerfreundliche Bedienung spezielle Suchanfragen entlang der Hierarchie der Lernobjekte und der Hierarchie des Klassifikationsschemas nötig. Wir haben uns daher für eine Eigenentwicklung auf der Basis einer existierenden objektorientierten Datenbanklösung entschieden.

Das MuSofT-Portal kann von jedem (neuerem) HTML-Browser aus bedient werden. Der Server basiert auf dem Infolayer-System [6], das als Servlet realisiert wurde. Das Infolayer-System ist eine objektorientierte Datenbank, deren Schema mit durch OCL annotierten UML-Klassendiagrammen spezifiert wird. Als Abfragesprache wird ebenfalls OCL verwendet. Die Daten werden als XML-Dateien abgespeichert. Eine Standardweboberfläche zur Navigation durch das Schema und durch die vorhandenen Objektinstanzen wird vom Infolayer-System automatisch zur Verfügung gestellt und kann sukzessive durch Schablonen an eigene Anforderungen angepasst werden, so dass Entwicklungsarbeiten sehr schnell zu bereits produktiven Prototypen führen. Diese Eigenschaften erlauben eine einfache Anpassung der Datenbank und des Portals, wenn sich die Metadaten aufgrund von Erfahrungen beim Einsatz des MuSofT-Portals ändern.

4.2 Verwendung von Metadaten

Das MuSofT-Portal dient zur Archivierung von Lernobjekten, die innerhalb (und außerhalb) von MuSofT erstellt werden. Die wichtigsten Aktivitäten, die mit dem Portal ausgeführt werden können, sind das Einfügen, Aktualisieren und Suchen von Lernobjekten. Beim Einfügen eines neuen Lernobjekts oder beim Modifizieren eines bestehenden Lernobjekts sind die bereits erwähnten Vorgaben für Metadaten zu berücksichtigen, damit innerhalb des MuSofT-Portals eine einheitliche Beschreibung der Lernobjekte erfolgt. Die Oberfläche des Systems unterstützt den Benutzer dabei, in dem es Eingabefelder für die erforderlichen Attribute anbietet und für Attribute mit fester Wertemenge eine Auswahlbox verwendet, so dass dort keine ungültigen Werte eingegeben werden können. Abbildung 1 zeigt einen Ausschnitt der Benutzeroberfläche des MuSofT-Portals zur Beschreibung eines Lernobjekts.

Neben dem reinen Hochladen von Lernobjekten unterstützt das MuSofT-Portal die in Kapitel 3 beschriebene und vom LOM-Standard vorgeschriebene Hierarchisierung von Lernobjekten. So kann z.B. eine neue Lehreinheit aus verschiedenen Lernmodulen, die zuvor in dem MuSofT-Portal erstellt wurden, zusammengesetzt werden.

Da Metadaten für Lernobjekte immer einen sehr subjektiven Charakter haben, kann es schwierig sein, passende Lernobjekte zu finden, wenn man sich ausschließlich auf Freitexteingaben für Stichworte und inhaltliche Beschreibungen verlässt. Wir verwenden daher für die inhaltliche Beschreibung zusätzlich eine festgelegte Taxonomie, die einen Ausschnitt aus dem Klassifikationsschema der ACM für Informatikliteratur darstellt, zugeschnitten auf Themenbereiche der Softwaretechnik

und des verwandten Gebiets der Datenbanktechnologie. Wir sollten anmerken, dass die Metadaten mit dieser Ausnahme nicht spezifisch auf Softwaretechnik zugeschnitten sind. Bei Bedarf kann dieses Klassifikationsschema sowohl um zusätzliche neue Themenbereiche als auch um verfeinerte Klassifikationen erweitert werden. Ebenso können bei Bedarf weitere unabhängige Klassifikationssysteme zur Verfügung gestellt werden. Lernobjekte können mit einem oder mehreren Einträgen aus dem Klassifikationsschema versehen werden, so wie dies bei Artikeln üblich ist. Wir erwarten, dass die inhaltliche Recherche wesentlich über dieses sehr bekannte und allgemein akzeptierte Klassifikationsschema stattfinden wird.

4.3 Interoperabilität und Nachhaltigkeit

Im Rahmen des MuSofT-Portals ist weiterhin die nachhaltige Nutzung der erstellten Materialien sicherzustellen. Nachhaltiger Einsatz bedeutet in diesem Fall, dass der wiederholte Einsatz der Materialien während der Laufzeit und nach der Beendigung des Projektes sichergestellt wird. Hierzu wurden, wie bereits in Abschnitt 3 beschrieben, Schritte unternommen, um die Nachhaltigkeit auf inhaltlicher und stilistischer Ebene zu ermöglichen. Die Sicherstellung der nachhaltigen Nutzung der in das Portal eingestellten Materialien hat jedoch auch einen technischen Aspekt: Es muss möglich sein, die Lernobjekte einschliesslich ihrer Metadaten auf unterschiedlichen Lernmanagementsystemen (LMS) zu verwenden, ohne die auf dem MuSofT-Portal erzeugten Metadaten erneut erstellen zu müssen. Ein einfacher Download der unterschiedlichen Lernobjekte ist in diesem Fall also nicht ausreichend. Zu diesem Zweck wird ein (standardisiertes) Exportformat benötigt, welches den Austausch einer Menge vernetzter Lernobjekte inklusive ihrer Metadaten ermöglicht. Hierfür wird das IMS Content Packaging-Format vom IMS Global Learning Consortium verwendet [8]. Die IMS Content Packaging-Spezifikation beschreibt Datenstrukturen, die den Austausch von Inhalten zwischen verschiedenen LMS und entsprechenden Autorensystemen standardisieren. So können Systeme Pakete mit Lernobjekten erstellen, die von anderen unabhängig entwickelten Systemen aufgrund der standardisierten Struktur des Pakets eingelesen werden können. Nur die Verwendung eines standardisierten Datenaustauschformats ermöglicht die Weiterverarbeitung der Lernobjekte in unterschiedlichen LMS, die an den Hochschulen eingesetzt werden.

Die Wahl bei dem Exportformat fiel auf das IMS Content Packaging-Format, da es zum einen auch den LOM-Standard berücksichtigt, d.h. die mit dem MuSofT-Portal erstellten Metadaten können mit den Lernobjekten exportiert und in anderen Systemen weiterverwendet werden. Zum anderen wird das IMS Content Packaging-Format bereits von einer Reihe von LMS (in Teilen) unterstützt. Kurze Übersichten, welche Programme den IMS Content Packaging-Standard unterstützen finden sich unter [9, 10]. Weiterhin beteiligen sich die wichtigsten Hersteller von E-Learning-Plattformen an der Entwicklung der IMS-Spezifikationen. Das IMS Content Packaging-Format wird auch im Rahmen des SCORM-Standards [1] verwendet. Mit dem Exportformat ist die Interoperabilität in einer Richtung, nämlich von MuSofT in LMS hinein, sichergestellt. Zusätzlich beabsichtigen wir zu einem späteren Zeit-

punkt auch einen Import des IMS Content Packaging-Formats zur Verfügung zu stellen, um so eine volle Interoperabilität in beiden Richtungen zu gewährleisten.

Ein weiterer zentraler technischer Aspekt für die mittel- und langfristige Nutzung der Lehrmaterialien ist die Änderbarkeit von bereits im Portal abgelegten Lernobjekten. Um diese Genese der Lernobjekte nachvollziehen zu können, stellen wir eine Versionsverwaltung zur Verfügung. Wir unterscheiden dabei zwischen Revisionen und Varianten eines Lernobjektes. Revisionen können nur vom Autor angelegt werden und sollen für relativ kleine Änderungen und Fehlerkorrekturen genutzt werden, etwa Beseitigung von Tippfehlern oder die Anpassung auf eine neue Version der verwendeten Notation. Soll das Lernobjekt im größeren Umfang umgestaltet werden, so wird dies als Variante bezeichnet. Ein Beispiel könnte die Umgestaltung einer Vorlesung für Informatiker zu einer Vorlesung für Nebenfächler sein. Benutzt man ein Lernobjekt aus dem Portal und modifiziert es, so kann man es als Variante des ursprünglichen Materials wieder in das Portal einstellen. Durch die Revisions- und Variantenverwaltung kann so der Ursprung des adaptierten Materials explizit deutlich gemacht werden, ebenso die Urheberrechte am Material. Um über Veränderungen an Lernobjekten informiert zu werden, können angemeldete Nutzer sich für einzelne Lernobjekte in Notifikationslisten eintragen lassen.

Als problematisch erweisen sich Verknüpfungen zwischen einzelnen Lernobjekten, wie sie etwa auftreten, wenn man eine Vorlesung in kleinere Lernobjekte zerlegt, aber auf ein verlinktes Gesamtinhaltsverzeichnis nicht verzichten möchte. Auch die Versionierung der Verknüpfungsstruktur ist nicht unproblematisch. Da Lernobjekte typischerweise keine Schnittstelle in Form von möglichen Linkzielen anbieten, ist bei einer neuen Version eines Lernobjektes unklar, ob die früher vorhandenen Linkziele innerhalb des Lernobjektes immer noch vorhanden sind. Hier ist noch weitere Arbeit notwendig, um zu einer auch pragmatisch handhabbaren Lösung zu kommen.

5 Zusammenfassung und Ausblick

Wir haben in diesem Papier die Aktivitäten des Projektes MuSofT vorgestellt, die für eine qualitativ hochwertige und nachhaltige Entwicklung von Lehrmaterialien zuständig sind. Gemeinsame didaktische Grundannahmen sowie eine inhaltliche und stilistische Abstimmung zwischen den Lernobjekten sind wichtige Vorausetzungen. Die Distribution der Lernobjekte an interessierte Lehrende ist für die Nachhaltigkeit von entscheidender Bedeutung, hierzu ist eine gleichförmige und leicht recherchierbare Beschreibung der Lernobjekte durch Metadaten notwendig. Operationalisiert wird dies im MuSofT-Portal.

Das MuSofT-Portal wird zum Wintersemester 2002/2003 freigeschaltet. Wir erwarten, dass erste Versionen der Lehrmaterialien aus MuSofT im Laufe des Wintersemesters eingestellt werden. Damit kann die Evaluation sowohl der Materialien als auch der Distributionsplattform mit seinen Metadaten beginnen.

Danksagung

Wir danken Jörg Pleumann für seine hilfreichen Anmerkungen.

Literatur

1. Advanced Distributed Learning. SCORM – Sharable Content Object Reference Model. http://www.adlnet.org, 2002.
2. Blackboard. http://www.blackboard.com, 2002.
3. CLIX-Camous. http://www.im-c.de, 2002.
4. Doberkat, E.-E. und Engels, G.MuSofT – Multimedia in der Softwaretechnik. Informatik Forschung und Entwicklung, 17(1):41-44,2002.
5. Duval, E., Forte, E.,Cardinaels, K.,Verhoeven, B., Durm, R.V. Hendrikx, K., Forte, M. W., Ebel, N., Macowicz, M., Warkentype, K. und Haenni, F. The ARIADNE Knowledge pool System. Communications of the ACM, 44(5):72-78, Mai 2001.
6. Haustein, S. und Pleumann, J. Is Paricipation in the Semantic Web Too Difficult? In I. Horrocks und J. Hendler (Herausgeber), The Semantic Web – Fist International Semantic Beb Conference, Band 2342 von LNCS. Heidelberg: Soringer, 2002.
7. IEEE Learning Technology Standards Committee, IEEE, 3 Oark Avenue New Work, NY 10016-5997, USA. Final Draft of the IEEE Standard for Lerning Objects and Metadata, Juni 2002. Online erhältlich unter http://ltsc.ieee.org/wg12/
8. IMS Global Learning Consortium , Inc. IMS Content Packaging Secification V. 1.1.2. http://www.imsproject.org/content/oackaging/index.cfm, August 2001
9. IMS Global Learning Consortium , Inc. Product Directory. http://www.imsproject.org/direct/getproducts.cfm, 2002
10. The University of Edinburgh, Deparment of Meteorology. IMS ContentPackaging Specification and Tolols – An Overview. http://www.met.ed.ac.uk/pac-man/cpoverview.html, 2002.
11. WebCT. http// www.webct.com, 2002.

Erste Erfahrungen mit dem Virtuellen Softwareprojekt

Ludger Bischofs, Wilhelm Hasselbring
Universität Oldenburg, FB Informatik, Abteilung Software Engineering, 26111 Oldenburg
ludger.bischofs@informatik.uni-oldenburg.de, hasselbring@informatik.uni-oldenburg.de

Hans-Jürgen Appelrath
Universität Oldenburg, FB Informatik, Abteilung Informationssysteme, 26111 Oldenburg,
appelrath@informatik.uni-oldenburg.de

Jürgen Sauer
Universität Oldenburg, FB Informatik, Software-Labor, 26111 Oldenburg,
sauer@informatik.uni-oldenburg.de

Oliver Vornberger
Universität Osnabrück, FB Mathematik/Informatik, 49069 Osnabrück,
oliver@uos.de

Zusammenfassung

Das Pilotprojekt „Virtuelles Softwareprojekt“ ermöglicht Osnabrücker Studierenden erstmals, hochschulübergreifend an der in Oldenburg durchgeführten Lehrveranstaltung „Softwareprojekt“ teilzunehmen. Neue Medien wie beispielsweise Videokonferenzsysteme werden eingesetzt, um die Veranstaltung über die reine Präsenzlehre hinaus durchführen zu können. Ein wesentliches Ziel des Projektes ist neben der Durchführung von hochschulübergreifenden Lehrveranstaltungen die Evaluation von Multimedia-Angeboten für die Lehre. Der erste Durchlauf des Softwareprojektes erfolgt im WS 2002/2003, so dass über erste Erfahrungen berichtet werden kann.

1 Einleitung

Softwarepraktika werden im Grundstudium vieler Informatikstudiengänge angeboten. An einigen Informatikfachbereichen werden auch im Hauptstudium Projektgruppen durchgeführt, in denen Gruppen von ca. zehn Studierenden im Zeitraum von mehreren Semestern gemeinsam ein größeres Projekt bearbeiten (Beispiele sind Dortmund, Oldenburg und Paderborn). Ein zentrales Ziel bei diesen Praktika, ob im Grund- oder Hauptstudium, ist die praktische Vermittlung von Techniken zur systematischen Arbeit im Team.

Das virtuelle Softwareprojekt ist ein Leitprojekt zur virtuellen Lehre im Rahmen des für das eLearning Academic Network Niedersachsen (ELAN) geförderten ELAN-Piloten Oldenburg/Osnabrück „epolos“ (siehe auch [1]). Dieses verteilte, virtuelle Softwareprojekt erlaubt es Studierenden an mehreren Standorten ab WS 2002/2003 an einem Softwareprojekt teilzunehmen, das zentral in Oldenburg koordiniert wird (siehe [3]). Damit können auch Standorte ein Softwareprojekt anbieten, die dies bisher aus personellen oder Kapazitätsgründen nicht konnten, wie es z.B. in Osnabrück der Fall war. Weiterhin müssen Vorlesungen durch die virtuelle Ausführung nur einmal gehalten werden und Tutoren können effizient eingesetzt werden.

2 Multimedia-Einsatz im Virtuellen Softwareprojekt in epolos

Die „Leitprojekte virtueller Lehre“ (zu denen das Softwareprojekt zählt) zeigen beispielhaft, wie die neuen Medien genutzt werden können, um über die reine Präsenzlehre hinaus neuartige Lehr- und Veranstaltungsformen zu konzipieren. Sie erschließen im Sinne eines Blended Learning die Bereiche der Weiterbildung, der verteilten kooperativen Seminar- und Projektarbeit sowie der zielgruppengerechten Adaption breit angelegter Lehrinhalte. Darüber hinaus ist die Stärkung der Medienkompetenz – nicht nur als technische, sondern auch als telekommunikative Kompetenz – der Studierenden ein wichtiges Ziel.

3 Das konventionelle Softwareprojekt

Aktuell ist das Softwareprojekt eine stark auf die Präsenz von Studierenden ausgelegte Veranstaltung, die sich über zwei Semester erstreckt. Wesentliches Ziel ist die Konstruktion eines Softwaresystems im Team, wobei Methoden des Software Engineering angewendet und eingeübt werden. Wichtig sind dabei auch die Projektorganisation und das Gruppenverhalten sowie die Präsentation von Ergebnissen. Im Rahmen des angegliederten Proseminars werden aktuelle Themen rund um das Software Engineering aufbereitet. Wesentliche Säulen der Veranstaltung sind:

Vorlesungsblock: Im Vorlesungsblock werden zu Beginn jeden Semesters Themen aus dem Bereich Software Engineering zusammengefasst, wiederholt und vor allem zusätzlich wichtige Bereiche wie z.B. Oberflächengestaltung, spezielle Programmiertechniken (GUI, DB) sowie Qualitätssicherung und Vorbereitung für das Proseminar behandelt. Hier werden auch die Aufgabenstellung und weitere Vorgaben (Standards, Projektplan, ...) präsentiert und die Gruppeneinteilung vorgenommen.

Gruppensitzungen: Die Gruppentreffen bestehen aus mindestens einer gemeinsamen Sitzung pro Woche, zu der auch der Tutor anwesend ist. Hier werden die wichtigen organisatorischen und inhaltlichen Entscheidungen getroffen,

Vorträge zu Themen des Praktikums (Auffrischung bereits behandelter und Aufbereitung neuer Themen mit Bezug zur Aufgabenstellung) gehalten und die Ergebnisse mit dem Tutor diskutiert.

Präsentationen: Die Präsentation von Ergebnissen des Softwareprojekts ist ebenfalls ein wichtiger Aspekt für das Einüben von Projektabläufen. Konkret müssen die Gruppen zum einen in einer Art Messepräsentation die Fähigkeiten des konstruierten Systems präsentieren. Zum anderen werden in einer Detailpräsentation am Ende die einzelnen Möglichkeiten des Systems vorgestellt, was unter anderem auch zur Bewertung des Systems herangezogen wird.

Softwareentwicklung: Die Entwicklung der Software erfolgt im Team. Kenntnisse aus der Vorlesung finden ihre praktische Umsetzung. Für Gruppen-arbeit geeignete Werkzeuge wie das Versionsverwaltungssystem CVS kommen zum Einsatz, um die Softwareentwicklung zu unterstützen. Die praktische Umsetzung des Gelernten mit dem Ergebnis einer lauffähigen Software inklusive Dokumentation stellen das zu erreichende Ziel dar.

4 Die Gestaltung eines virtuellen Softwareprojektes

Mit dem Vorhaben „Virtuelles Softwareprojekt“ werden im Wesentlichen zwei Zielsetzungen verfolgt. Eine davon ist die Durchführung einer Lehrveranstaltung über Hochschulgrenzen hinweg. Die Evaluation von Multimedia-Angeboten für die Lehre ist die zweite Zielsetzung und wird im folgenden Unterabschnitt kurz diskutiert, bevor dann die geplante Durchführung skizziert wird.

4.1 Evaluation von Multimedia-Angeboten für die Lehre, insbesondere für die Ausbildung in der Softwaretechnik

Das virtuelle Softwareprojekt bietet sich als Fallbeispiel für die Evaluation von Multimedia-Angeboten an, da es sich – schon in seiner bisherigen Form als standortgebundenes Projekt – um eine Veranstaltung handelt, die verschiedenartige klassische Lehr- und Lernformen in sich vereint. Es gibt einen einführenden Vorlesungsblock und einen Seminarteil, Materialien müssen zeitnah bereitgestellt werden, Gruppenarbeit zur Lösung von Aufgaben ist nötig und erzielte Ergebnisse müssen präsentiert werden.

Durch die Erweiterung zu einer über Hochschulgrenzen hinweg durchführbaren Veranstaltung mit virtuellen (verteilten) Teams werden zusätzliche Anforderungen an die Gestaltung und Durchführung einer solchen Veranstaltung gestellt, z.B. in Bezug auf die bereitgestellten Materialien, die Durchführung von Gruppensitzungen an verteilten Orten und die Kontrolle von erstellten Ergebnissen. Insgesamt stellt das Softwareprojekt einen sehr vielseitigen Kandidaten in Bezug auf die Virtualisierung dar.

4.2 Virtualisierung des Softwareprojektes

Die Softwareentwicklung in Open Source-Projekten wie Linux erfolgt bereits seit längerem verteilt über den gesamten Globus. In der Industrie sieht die globale Zusammenarbeit oft etwas anders aus. Sie besteht mehr aus einem Auftragnehmer-Auftraggeber-Verhältnis, wobei vor Ort in geschlossenen Gruppen an bestimmten Teilaufgaben gearbeitet wird [6,7].

Die Zielvorstellung der Virtualisierung des Softwareprojektes reicht so weit, dass Studierenden an unterschiedlichen Standorten ermöglicht werden soll, zusammen als ein Team am Softwareprojekt teilzunehmen. Inwieweit dies in der Praxis relevant ist, hängt jedoch von der Akzeptanz der Teilnehmer ab. Für erste Erfahrungen bietet sich die einfachere Variante des besagten Auftragnehmer-Auftraggeber-Verhältnisses an, wodurch die Kommunikation innerhalb der Teams aufgrund der räumlichen Nähe stark vereinfacht wird.

Innerhalb des Softwareprojektes erbrachte Leistungen müssen an den Standorten gleichermaßen anerkannt werden. Dies ist vor dem Beginn eines Softwareprojektes zu klären. Benotungen basieren neben der Gruppenarbeit als Ganzes ebenso auf den Einzelleistungen der Teilnehmer wie den anfänglichen Programmieraufgaben, Vorträgen und Ausarbeitungen.

Der folgende Abschnitt beschreibt die Erfordernisse eines virtuellen Softwareprojektes unabhängig von technischen Gegebenheiten.

Vorlesungsblock: Die Vorlesung sollte allen Teilnehmern des Softwareprojektes zugänglich sein. Als sinnvoller Weg erscheint die Übertragung der Vorlesung in Form einer Videokonferenz für die Teilnehmer, die aufgrund der Entfernung nicht präsent sein können. Dadurch wird eine bidirektionale Kommunikation mit der Möglichkeit eventueller Rückfragen an den Lehrenden ermöglicht.
Die Lehrinhalte sollten zusätzlich im WWW abgelegt werden, so dass alle Teilnehmer Zugriff darauf haben. Learning-Management-Systeme (LMS) bzw. Lernplattformen bieten entsprechende Funktionalitäten.

Gruppensitzungen: Die Gruppentreffen zwischen den verteilten Teilnehmern einer Projektgruppe und dem Tutor sollten als Videokonferenz einmal wöchentlich durchgeführt werden. Neben dem Führen von Diskussionen können so auch Vorträge übertragen werden.

Präsentationen: Die Präsentation der Ergebnisse des Softwareprojektes erfolgt über eine eigene Internetpräsenz der jeweiligen Gruppe. Die Endabnahme der Software kann in einer Präsenzveranstaltung am Ende des Semesters erfolgen.

Softwareentwicklung: Benötigte Hard- und Software muss den Projektteilnehmern an den jeweiligen Standorten zur Verfügung stehen. Versionsverwaltungssysteme vereinfachen die Gruppenarbeit. Die verteilte Arbeit an Quelltexten und Dokumentation wird dadurch ermöglicht. Die Gruppenarbeit wird durch Lernplattformen unterstützt, welche die Abläufe der kooperativen web-

basierten Arbeitsprozesse koordinieren helfen. Lernplattformen können für den Austausch von Dokumenten genutzt werden und stellen zusätzliche Kommunikationshilfsmittel zur Verfügung.

5 Erste praktische Erfahrungen

Im Wintersemester 2002/2003 wird zurzeit auch Osnabrücker Studierenden die Teilnahme an dem in Oldenburg stattfindenden Softwareprojekt ermöglicht. Oldenburger Hörsäle wurden technisch so erweitert, dass eine Übertragung der Vorlesung möglich ist. Darüber hinaus wurde in Oldenburg die Lernplattform Blackboard installiert, die für den Dokumentenaustausch genutzt wird und zusätzliche Kommunikationshilfsmittel wie Diskussionsplattformen (Foren), Chats, Whiteboards und E-Mail bereitstellt.

Die bisherigen Erfahrungen in den vier Säulen des zur Zeit durchgeführten Softwareprojektes werden im Folgenden beschrieben.

Vorlesungsblock: Die in Oldenburg gehaltene Vorlesung zum Softwareprojekt erstreckte sich über einen Zeitraum von fünf Wochen zu Beginn der Veranstaltung und wurde aus dem Hörsaal heraus über Polycom-Kameras [9] als Videokonferenz nach Osnabrück ausgestrahlt. Der Vortragende konnte die Osnabrücker Zuhörer, die auch Zwischenfragen stellen konnten, über einen Kontrollbildschirm sehen.

Die alleinige Übertragung des Vortragenden vor der Leinwand erschien als ungeeignet, weil die Auflösung der Übertragung nicht genügend hoch ist, um Details der Präsentation auf der Leinwand in ausreichend hoher Qualität sehen zu können.

Die Präsentation wurde deshalb über die Desktop-Freigabe von Microsoft NetMeeting [11] auf eine weitere Leinwand in Osnabrück projiziert. Zwei Rechner wurden für diesen Zweck – unabhängig von der Polycom-Technologie – über das Internet miteinander verbunden. Obwohl die Polycom-Kameras die Übertragung von Microsoft PowerPoint-Präsentationen unterstützen, wäre diese Form der Übertragung nicht ausreichend gewesen. Die Desktop-Freigabe von Microsoft NetMeeting ermöglichte dagegen die Freigabe beliebiger Anwendungen zu Präsentationszwecken, so dass neben Präsentationsfolien im Microsoft PowerPoint-Format ebenfalls Beispielanwendungen in Java vorgestellt werden konnten.

Die Vorbereitungszeit für das Einrichten einer akzeptablen Übertragungsqualität der Vorlesung, an der zwei Oldenburger Techniker arbeiteten, belief sich anfangs auf ca. 30 Minuten. Im Laufe der Zeit konnte die Vorlaufphase auf etwa 15 Minuten verkürzt werden. Von Vorlesung zu Vorlesung wurden Er-

fahrungen gesammelt, welche die Übertragungsqualität z.B. durch den Test unterschiedlicher Mikrofone verbesserten.

Alle Lehrinhalte (der Content) wurden auch im WWW abgelegt. Die Bereitstellung erfolgt über die Lernplattform Blackboard [8] und vorlesungseigene WWW-Seiten, die bereits seit längerem existieren und ebenfalls Informationen zu vorherigen Softwareprojekten beinhalten.

Gruppensitzungen: Die Gruppentreffen zwischen den Osnabrücker Teilnehmern und dem Tutor in Oldenburg werden als Videokonferenz über die Polycom-Technologie einmal wöchentlich durchgeführt. Zusätzlich bereitet jeder Teilnehmer eine Teilaufgabe vor, die er dem Tutor und den anderen Teilnehmern innerhalb einer Gruppensitzung vorträgt. Diese Präsentation erfolgt ähnlich der Vorgehensweise im Vorlesungsblock über die Desktop-Freigabe bestimmter Anwendungen über Microsoft NetMeeting. Beispielsweise wurden auf diese Weise die Entwicklungstools Sun ONE Studio, Poseidon und eclipse von einem der Osnabrücker Teilnehmer vorgestellt.

Da die Gruppensitzungen im Gegensatz zur Vorlesung im kleinen Rahmen stattfinden, erwies sich der technische Aufwand für die Durchführung als geringer.

Präsentationen: Die Präsentationen der Ergebnisse der Projektgruppen werden auf einem WWW-Server in Oldenburg abgelegt. Das gilt auch für die Osnabrücker Gruppe. Die Endabnahme der Software wird in einer Präsenzveranstaltung am Ende des Semesters erfolgen.

Softwareentwicklung: In Osnabrück stehen für die Softwareentwicklung benötigte Rechner bereit. Das Sun ONE Studio in Verbindung mit Poseidon für die UML-Modellierung wurden als Entwicklungswerkzeuge vorgegeben. CVS kommt als Versionsverwaltungssystem zum Einsatz und ermöglicht die verteilte Arbeit an Quelltexten und Dokumentationen. Für die Erstellung der Dokumentation entschied sich die Osnabrücker Gruppe für LaTeX. Die Gruppenarbeit wird durch die Kommunikationshilfsmittel der Lernplattform Blackboard unterstützt, dessen Akzeptanz bisher jedoch gering ist. Lediglich der Dateiaustausch und die Diskussionsplattform wurden intensiver genutzt. Die Ursachen für die bisher geringe Nutzung der Lernplattform werden zurzeit untersucht.

In einem Gespräch mit der Neuen Osnabrücker Zeitung wurden die Osnabrücker Teilnehmer über ihre ersten Eindrücke zum Virtuellen Softwareprojekt befragt. Demnach liegt das Interesse der Studierenden nicht nur in der Durchführung eines Softwareprojektes, sondern zu einem großen Teil auch im Erlernen des Umgangs mit neuen multimedialen Technologien. Das Virtuelle Softwareprojekt wird als eine Veranstaltungsform angesehen, die in Zukunft zum Standard avancieren könnte.

6 Verwandte Arbeiten

Die Software Engineering Lehrveranstaltung der TU Darmstadt ist um ein Projektpraktikum mit realen Auftraggebern organisiert [10]. Der Schwerpunkt für den Einsatz neuer Medien liegt dort auf der Kommunikation zwischen Anwendern und Entwicklergruppe. Im Gegensatz dazu ist die Zielvorstellung des Virtuellen Softwareprojektes eine andere. Neue Medien sollen bei der Kommunikation zwischen Lehrenden und Studierenden helfen, eine hochschulübergreifende Lehrveranstaltung durchführen zu können.

Das Projekt iBistro [2] stellt eine Lernumgebung für verteilte Softwareentwicklungsprojekte dar und wird in einem ähnlichen Szenario wie in epolos zwischen Singapur und München eingesetzt. Der Fokus des Projektes liegt allerdings aufgrund des Zeitunterschiedes eher auf der asynchronen Kommunikation. Ein geplanter Erfahrungsaustausch zwischen epolos und iBistro führt in Zukunft möglicherweise zu Synergieeffekten und neuen Erkenntnissen für die Durchführung von virtuellen Softwareprojekten.

7 Zusammenfassung und Ausblick

Die ersten Schritte in der Durchführung des Virtuellen Softwareprojektes sind getan und verliefen bis zum jetzigen Zeitpunkt relativ reibungslos. Die technische Infrastruktur für die Videokonferenzen wurde installiert und im bisherigen Verlaufe des Softwareprojektes – bis auf kleinere technische Schwierigkeiten – erfolgreich eingesetzt. Organisatorische Dinge wie die Anerkennung der Leistungen der Osnabrücker Projektteilnehmer wurden vorab geklärt.

Das Ziel der Evaluation von Multimedia-Angeboten für die Lehre betrifft den Einsatz von Softwaresystemen, die generell zur Unterstützung multimedialer Lehre entwickelt wurden [5], aber auch spezifische Angebote für die Informatik-Ausbildung (z.B. MuSoft-Lehrvideo [4]). Nach der ersten Durchführung des Virtuellen Softwareprojektes wird für die zukünftigen Praktika eine umfassendere Analyse bezüglich der Evaluation erfolgen, da noch weitere Erkenntnisse im Laufe des Projektes einfließen werden.

Der Erfolg des Virtuellen Softwareprojektes hängt in Zukunft vor allem von der Akzeptanz der Studierenden ab. Insbesondere der technische und organisatorische Mehraufwand für die Durchführung der virtuellen Veranstaltung wird zu bewerten sein.

Literatur

1. SBMM: Strategischer Beraterkreis Multimedia, http://www.sbmm-niedersachsen.de - besucht am 18.11.2002
2. Braun, Andreas; Dutoit, Allen H.;G. Harrer, Andreas; Brügge, Bernd: iBistro: A Learning Environment for Knowledge Construction in Distributed Software Engineering Courses. In: Proceedings: 9th Asia-Pacific Software Engineering Conference – IEEE CS Press, Dezember 2002
3. Hasselbring, W.; Appelrath, H.-J.; Sauer, J.; Vornberger, O.: Verteiltes, virtuelles Softwareprojekt. In Softwaretechnik-Trends 22(3): 40-42, August 2002
4. Kelter, U.: Versions- und Konfigurationsmanagement in der Ausbildung in praktischer Informatik. In: Softwaretechnik-Trends 22(1): 26-27, Februar 2002
5. Veglis, A.A.: Web-based teaching systems. In: IEEE Distributed Systems Online 4(3), April 2002
6. Englert, Sylvia: Keine Ruhepause im virtuellen Team-Room. In: changeX (http://www.changex.de), September 2001
7. Englert, Sylvia: So fern und doch so nah. In: changeX (http://www.changex.de), September 2001
8. Blackboard: Blackboard Learning System ML Entry Page, http://www.blackboard.com - besucht am 08.10.2002
9. Polycom: Your video conferencing and speaker phone source! Polycom Worldwide, http://www.polycom.com - besucht am 08.10.2002
10. Schroeder, Ulrike; Brunner, Michael: Aktive Anwenderbeteiligung in Ausbildungsprojekten. In: SEUH ’99 - German Chapter of the ACM Berichte Nr. 52, Teubner Verlag, 1999
11. Microsoft GmbH: NetMeeting Home, http://www.microsoft.com/windows/netmeeting/ - besucht am 29.11.20

Erfahrungen mit einem Workshop-Seminar im Software Engineering-Unterricht

Horst Lichter[1], Ralf Melchisedech[2], Oliver Scholz[2], Thomas Weiler[1]
[1] Lehr- und Forschungsgebiet Softwarekonstruktion, RWTH Aachen
{lichter, thweiler}@cs.rwth-aachen.de
[2] sd&m AG, Köln
{ralf.melchisedech, oliver.scholz}@sdm.de

Zusammenfassung

Seminare sind eine bewährte Lehrform in allen Studiengängen. Um die Mitarbeit und den Lerneffekt der Studierenden in Seminaren zu Themen des Software Engineering zu erhöhen, führte das Lehr- und Forschungsgebiet Softwarekonstruktion der RWTH Aachen zusammen mit dem Softwarehaus sd&m Köln ein Workshop-Seminar durch. Die Lehrveranstaltung gliederte sich in zwei Teile: Im ersten Teil bearbeiteten die Studierenden ein inhaltliches Thema. Der zweite Teil war ein zweitägiger Workshop, bei dem die Studierenden in Gruppen realitätsnahe Problemstellungen bearbeiten und präsentieren mussten. Die mit dieser Lehrform erzielten Ergebnisse waren durchweg positiv.

1 Einleitung

Die Workshop-Reihe SEUH beschäftigt sich seit einigen Jahren ganz speziell mit Lehrformen im Bereich Software Engineering. In den publizierten Beiträgen, aber auch in den Plenumsdiskussionen wird immer wieder darauf hingewiesen, dass SE-Lehrveranstaltungen besondere Ausbildungsziele verfolgen und entsprechend darauf angepasst sein müssen. Nachfolgend werden diese Ziele noch einmal kurz zusammengefasst:

- Vermittlung von gesichertem *Fachwissen* im Bereich Software Engineering
- *Anwenden* des Fachwissens an realitätsnahen Problem- und Aufgabenstellungen.
- *Reflexion* über die Anwendung des Fachwissens (Lewerentz, 2001)
- *Praxisnahe Ausbildung* der Studierenden durch Beteiligung der Industrie und Kennenlernen von industriell benutzten Methoden, Sprachen und Werkzeugen.

Mindestens ebenso wichtig wie die fachlich-technische Ausbildung ist die Vermittlung von so genannten "Soft Skills" (Denert, 2000). Dazu zählen:

- *Kommunikationsfähigkeit*, also die Gabe, sich anderen mitzuteilen, aber auch zuhören zu können. Man muss sich mündlich verständlich machen können, in Gesprächen, Diskussionen und Vorträgen, aber auch schriftlich.
- *Teamgeist und Integrationsfähigkeit*, um mit Kollegen und Kunden reibungslos und harmonisch zusammenzuarbeiten, ohne die eigene Persönlichkeit aufzugeben.
- *Begeisterungsfähigkeit und Engagement* bei der Arbeit, damit überdurchschnittlich gute Ergebnisse erzielt werden können.
- *Kreativität und logisch-strukturelles Denken*, aber gelegentlich muss auch assoziativ, sprunghaft und „unlogisch" gedacht werden, damit neue Ideen und Lösungen entstehen.
- *Nützlichkeitsstreben*, denn der Software-Ingenieur muss etwas bauen wollen, das andere, die Anwender, wirklich brauchen. Eine tolle technische Lösung allein reicht nicht; Kosten und Nutzen müssen in einem vernünftigen Verhältnis stehen.
- *Offenheit und Kritikfähigkeit*. Offene Kritik ist nicht nur erlaubt, sondern erwünscht. Sie soll allerdings möglichst konstruktiv sein, und es wird erwartet, dass auch das Positive erkannt und gewürdigt wird.

Lehrveranstaltungen im Bereich Software Engineering sollten versuchen, diese Ausbildungsziele – zumindest einige davon – zu erreichen. Dies ist sicherlich nicht immer einfach und mit erhöhtem Aufwand verbunden. Gute Beispiele, insbesondere im Bereich der Praktika (z.B. Glinz 1999, Schröder 1999), zeigen jedoch, dass dies möglich ist.

In diesem Beitrag berichten wir über die Erfahrungen, die wir mit einem Workshop-Seminar gewonnen haben. Zuerst betrachten und bewerten wir die Rolle von Seminaren im Software Engineering-Unterricht. Anschließend beschreiben wir die Ideen und Konzepte der neuen Lehrveranstaltung und schildern deren Aufbau und Durchführung. Abschließend bewerten wir diese und fassen die gemachten Erfahrungen zusammen.

2 Seminare in der Hochschulausbildung

Seminare gehören neben Vorlesungen und Praktika zu den klassischen Lehrformen in jedem Studiengang. Sehr häufig sind Seminare Pflichtveranstaltungen; die dabei erworbenen Leistungsnachweise sind Voraussetzung für die Diplomvor- bzw. die Diplomhauptprüfung. Die Studienordnung des Diplomstudiengangs Informatik der RWTH Aachen beschreibt die Ziele der Lehrform Seminar wie folgt:

"Seminare haben im Rahmen des Hauptstudiums eine zentrale Bedeutung. Sie dienen der Aneignung und Präsentation wissenschaftlicher Erkenntnisse. Darüber hinaus können sie zur Vorbereitung der Anfertigung der Diplomarbeit im Sinne einer Einarbeitung in ausgewählte Gebiete der Informatik dienen."

2.1 Das klassisch organisierte Seminar

Seminare werden häufig nach folgendem Schema organisiert und durchgeführt: Zu einem übergeordneten Gebiet werden anhand einzelner Literaturreferenzen Themen formuliert, die an die Studierenden vergeben werden. Diese arbeiten sich auf Basis der Literatur und weiteren selbst gesuchten Quellen in die Themenstellung ein. Anschließend verfassen die Studierenden eine schriftliche Ausarbeitung zum gewählten Thema und extrahieren daraus eine Präsentation. Die Präsentationen werden entweder wöchentlich oder in Form eines Blockseminars an ein bis zwei Tagen gehalten.

Damit ein Seminar erfolgreich durchgeführt werden kann, müssen die Studierenden durchgängig betreut werden. Hier stehen folgende Aspekte im Vordergrund:

- Die Studierenden erhalten eine Anleitung, wie Ausarbeitungen strukturiert und Präsentationen aufgebaut und durchgeführt werden sollten.
- Die Betreuer sind die Ansprechpartner, um thematische und inhaltliche Probleme zu besprechen und zu klären.
- Die Betreuer leiten die Studierenden beim Verfassen der schriftlichen Ausarbeitungen und der Präsentationen an und bewerten diese.

2.2 Erfahrungen und Bewertung

Klassisch durchgeführte Seminare sind geeignet, das vorher beschriebene Ausbildungsziel – die Aneignung und Präsentation wissenschaftlicher Erkenntnisse – zu erreichen. Der Arbeitsaufwand, den die Studierenden für eine Seminararbeit investieren müssen, ist zum Teil erheblich. Unsere Erfahrung zeigt jedoch, dass sich die einzelnen Studierenden sehr stark – wenn nicht ausschließlich – auf das von ihnen gewählte Thema konzentrieren. Das führt dazu, dass bei den Präsentationen sehr wenig Diskussion entsteht. Diese muss in der Regel von den Betreuern angestoßen werden. Die Studierenden handeln dabei oftmals nach dem Prinzip "Ich frage dich nichts, du fragst mich nichts". Dies führt zu Vorträgen, die wenig lebendig diskutiert werden und deren Inhalte von den Zuhörern auch wenig aufgenommen werden. Dies ist jedoch sicherlich nicht gewünscht, und es gilt, dieses zu vermeiden.

3 Konzept des Workshop-Seminars

Während bei Vorlesungen begleitende Übungen dazu dienen, dass die Studierenden nicht nur Wissen vermittelt bekommen, sondern auch Erfahrungen beim Anwenden des Wissens erhalten, so ist das bei Seminaren der klassischen Prägung nicht der Fall. Hier steht die Präsentation von Wissen und nicht die Anwendung des Wissens im Vordergrund.

In einem Gespräch zwischen Mitarbeitern des Softwarehauses sd&m und des Lehr- und Forschungsgebiets Softwarekonstruktion der RWTH Aachen wurde die Idee entwickelt, die Lehrform Seminar zu einer workshopartigen Lehrveranstaltung

umzugestalten. Dadurch sollte erreicht werden, dass die Studierenden sich nicht nur Wissen im Bereich des Software Engineering aneignen, sondern auch die Möglichkeit erhalten, dieses anzuwenden und zu bewerten. Es sollte eine Lehrveranstaltung konzipiert werden, die zum einen den Anforderungen an ein Seminar des Diplomstudiengangs Informatik genügt, zum anderen den Studierenden einen möglichst praxisnahen Einblick in die Probleme von Softwareentwicklungsprojekten vermittelt.

Da im Vorlesungskanon bereits Vorlesungen zu den Themen „Software-Qualitätssicherung“ und „Projektmanagement“ enthalten sind, bot es sich an, ein Seminar aus diesen Themenbereichen anzubieten. Dabei sollten die Studierenden, nach der Beschäftigung mit spezifischen Fragestellungen der Qualitätssicherung und des Projektmanagements, in einem Workshop mit realitätsnahen Problemstellungen konfrontiert werden, die alle Studierenden gemeinsam lösen sollten. Reale Projektsituationen sollten durch den Partner sd&m eingebracht werden, indem echte Projektdaten und -dokumente in anonymisierter Form im Workshop verwendet werden sollten.

4 Durchführung des Workshop-Seminars

4.1 Aufbau der Lehrveranstaltung

Die neue Lehrveranstaltung bestand aus zwei Teilen: Im ersten Teil bearbeiteten die Studierenden verschiedene Themen aus den Bereichen Projektmanagement und Qualitätssicherung und erstellten Ausarbeitungen hierzu. Diese sollten im zweiten Teil – einem Workshop – als Informationsquelle und Nachschlagewerk genutzt werden. Die ausgegebenen Themen wurden in drei Bereiche gruppiert. Folgende Themen wurden von den 14 Studierenden bearbeitet:

Prozesse	Projektmanagement	Qualitätssicherung
• Prozessmodelle • Prozessbewertung mit SPICE • Prozessbewertung mit Bootstrap	• Projektorganisation • Planungstechniken • Fortschrittskontrolle • Risikomanagement • Konfigurationsmanagement • Der Faktor Mensch	• QS –Überblick • Reviews • Testen und Testprozesse • Metriken • Prototyping

Tabelle 1: Themen des Workshop-Seminars

In einer Einführungsveranstaltung, die ca. eine Woche vor dem Workshop durchgeführt wurde, wurden die Ziele und der Ablauf des Workshops erläutert. Ein sd&m-Mitarbeiter stellte die Projektvorgehensweise (Meier 2000) und das Qualitätsmanagementsystem vor. Dadurch wurden eine gemeinsame Basis und insbesondere ein

gemeinsames Verständnis der Projektterminologie geschaffen. Des Weiteren erläuterte er den Studierenden die im Projekt erstellten Dokumenttypen.

Zu Beginn des zweitägigen Workshops präsentierten sd&m-Mitarbeiter den Studierenden ein (fiktives, aber an der Realität orientiertes) Projekt. In diesem Projekt war sehr vieles falsch gelaufen. Aufgabe der Studierenden war es, die Probleme zu erkennen und Lösungen und Verbesserungsvorschläge zu erarbeiten. Die Studierenden arbeiteten dabei in Gruppen zusammen, die jeweils von einem Mitarbeiter betreut wurden. Die erzielten Ergebnisse wurden vor Ort schriftlich fixiert, in kurzen Vorträgen im Plenum vorgestellt und diskutiert. Tabelle 2 zeigt den Zeitplan des Workshop-Seminars.

Zeitpunkt	Aktivität
Mitte Juli 2001	Informationsveranstaltung für Studierende Konzept und Ziele wurden erläutert. Es wurde darauf hingewiesen, dass mit einem höheren Arbeitsaufwand auch auf Seiten der Studierenden zu rechnen sei.
Anfang Oktober 2001	Ausgabe der Themen an die Studierenden (siehe Tabelle 1)
Mitte Dezember 2001	Abgabe der ersten Fassung der Ausarbeitungen
Ende Januar 2002	Abgabe der überarbeiteten Ausarbeitungen. Diese wurden zusammengebunden und an die Teilnehmer verteilt.
Anfang Februar 2002	Einführungsveranstaltung mit Ausgabe der Projektdokumentation
Mitte Februar 2002	Zweitägiger Workshop

Tabelle 2: Zeitplan der Lehrveranstaltung

4.2 Das fiktive Projekt

Am vorgestellten Projekt waren zwei fiktive Firmen und ein externer Berater beteiligt. Der Auftraggeber war die Firma Cyberdine, eine Versicherungsgesellschaft. Auftragnehmer war Skynet, ein Softwarehaus, das sich auf Informationssysteme und Internetanwendungen spezialisiert hat. In diesem Projekt sollte eine Online-Anbindung für Außendienstmitarbeiter an das zentrale Informationssystem realisiert werden. Die Entwicklung wurde in einem gemischten Team durchgeführt: die eine Hälfte der Mitarbeiter stellte die interne Entwicklungsabteilung von Cyberdine, die andere Hälfte Skynet. Die Projektleitung wurde zwischen Cyberdine und Skynet aufgeteilt. Das Projekt zeigte an seinem Ende die folgenden Merkmale:

- Das auf ursprünglich 12 Monate geplante Projekt wurde erst nach 18 Monaten abgeschlossen.
- Der Aufwand hatte sich gegenüber der ursprünglichen Planung verdoppelt.
- Die erste Stufe des erstellten Systems wurde mit einer Verspätung von neun Monaten ausgeliefert, das Gesamtsystem mit einer Verspätung von sechs Monaten.
- Nach ca. sieben Monaten wurde ein externer Berater engagiert, der die Software-Architektur überarbeiten sollte.

Die Stimmung zu Projektende war denkbar schlecht. Die Fachabteilung von Cyberdine war mit der Qualität sehr unzufrieden. Jeder gab anderen die Schuld an der Misere.

Die Projektschilderung orientierte sich, was die Fachlichkeit betrifft, an einem realen Projekt. Der Projektablauf und die geschilderten Probleme waren zwar konstruiert, basierten aber auf Erfahrungen der sd&m-Mitarbeiter aus verschiedenen Projekten.

4.3 Ablauf des zweitägigen Workshops

Ein wichtiges Ziel war es, den Studierenden die Projektsituation so realistisch wie möglich darzustellen. Deshalb haben sd&m-Mitarbeiter bei der Schilderung des Projekts verschiedene Rollen eingenommen:

- Frank Necker: Leiter der Fachabteilung von Cyberdine
- Oliver Krause: Leiter der Informatikabteilung von Cyberdine
- Ralf Käufer: Chef-Designer von Skynet

Jeder hat die Projektprobleme aus "seiner" Sicht geschildert und den anderen Versagen im Projekt vorgeworfen. Die Vorwürfe waren mit Absicht nicht immer sachlich. Nachfolgend ist ein Dialog wiedergegeben, der die Stimmung zeigt, in der den Studierenden die Probleme des Projektes präsentiert wurden.

Frank Necker: Ich habe die glasklare Forderung nach einem möglichst schnellen System gestellt. Stattdessen haben Sie mir ein unmöglich langsames System geliefert!
Ralf Käufer: Performanz-Probleme gibt es immer, wenn Sie während des laufenden Betriebes eine Massenreplikation der Daten durchführen. Im Fachkonzept steht nicht geschrieben, dass dies im laufenden Betrieb möglich sein soll.
Frank Necker: Das weiß doch jeder, das muss ich Ihnen bei Ihren Berater-Stundensätzen doch nicht extra sagen. Und können Sie mir verraten, wie ich in Ihrem 500-Seiten-Dokument meine Anforderungen erkennen kann?
Oliver Krause: Die Systemarchitektur ist schlecht. Zum Glück habe ich einen externen Berater, der übrigens ein guter Freund von mir ist, hinzugezogen. Der hat mir das dann auch prompt bestätigt.
Frank Necker: Versuchen Sie bloß nicht, den Schwarzen Peter los zu werden. Sie als Projektleiter tragen doch die Verantwortung dafür, dass das System auch noch die Wünsche der Marketingabteilung abdecken soll. Da hat sich mein Marketing-Kollege mal wieder voll ins Licht gestellt.
Ralf Käufer: Wir mussten in der Tat Kompromisse zwischen den Wünschen des Verkaufes und des Marketings machen. Das ist aber im Fachkonzept ab Seite 475 beschrieben...
Oliver Krause: ... und Sie, Herr Necker, haben keine Einwände gehabt. Ja, ja, ich kenne schon Ihre Ausrede „das Tagesgeschäft". Organisieren Sie Ihre Abteilung besser, dann haben Sie auch Zeit, ein Konzept zu lesen.

Die Situationsbeschreibung und Einführung in das Projekt dauerte ca. einen halben Tag. Anschließend waren die Studierenden an der Reihe. Es wurden drei Gruppen zu je 4-5 Studierenden gebildet. Jede Gruppe sollte folgende Fragestellungen, jeweils unter einem anderen Schwerpunkt (Projektmanagement, Qualitätssicherung, Projektdurchführung), bearbeiten:

- *Ist-Analyse*: *Was ist wirklich falsch gelaufen?*
  Ziel war es, aus den subjektiven Ansichten die eigentlichen Probleme zu extrahieren.
- *Soll-Analyse Teil 1*: *Was kann man besser machen?*
  Welche Maßnahmen können ergriffen werden, um diese Probleme zu vermeiden? Ergebnis sollte eine noch unbewertete Liste von Maßnahmen sein.
- *Soll-Analyse Teil 2*: *Welche dieser Maßnahmen können in einem nachfolgenden Projekt sinnvoll umgesetzt werden?*
  Die Maßnahmen sollten auf ihre Anwendbarkeit in der Projektumgebung hin untersucht werden. Es sollte ein sinnvolles Maßnahmenpaket geschnürt werden, denn schließlich kann nicht alles auf einmal umgesetzt werden.

Zwischen den Gruppenarbeiten waren mehrere Frage- und Diskussionsrunden vorgesehen, in denen die sd&m-Mitarbeiter wieder ihre Rollen als Frank Necker, Oliver Krause und Ralf Käufer eingenommen haben. Am Ende jeder Gruppenarbeit präsentierten und diskutierten die Gruppen ihre Ergebnisse im Plenum. Der detaillierte Zeitplan des Workshops ist in Tabelle 3 wiedergegeben.

Zeitplan 1. Tag	
09:00 – 09:15	Einführung
09:15 – 10:15	Projektvorstellung-1
10:15 – 10:30	Pause
10:30 – 11:30	Projektvorstellung-2
11:30 – 12:00	Aufgabenverteilung
12:00 – 13: 00	Mittagspause
13:00 – 14:30	Gruppenarbeit-1
14:30 – 14:45	Pause
14:45 – 15:15	Fragerunde
15:15 – 16:45	Gruppenarbeit-2
16:45 – 17:00	Pause
17:00 – 18:30	Präsentationen
ab 19:00	gem. Abendessen

Zeitplan 2. Tag	
09:00 – 09:15	Aufgabenverteilung
09:15 – 10:45	Gruppenarbeit-3
10:45 – 11:00	Pause
11:00 – 12:30	Präsentationen
12:30 – 13:15	Mittagspause
13:15 – 15:45	Gruppenarbeit-4
15:45 – 17:15	Abschlusspräsentationen
ab 17:15	Feedback-Runde

Tabelle 3: Zeitplan des Workshops

4.4 Ergebnisse der Studierenden

Die Ergebnisse des Workshops füllten 70 Folienseiten, deren Inhalt hier nur kurz skizziert werden kann.

- Die Projektmanagement-Gruppe gliederte ihre Überlegungen unter den Aspekten Planung, Kontrolle und Steuerung des Projektes. Für die Maßnahmenliste wurden auch Vorschläge für die Initialphase gemacht (z.B. Festlegung des Berichtswesens und klare Definition der Rollen und Verantwortlichkeiten).
- Die Mitglieder des Teams Projektdurchführung orientierten sich an den Phasen Projekteinstieg, Spezifikation, Entwurf, Implementierung, Test und Inbetriebnahme. Bei der Problemanalyse wurde zusätzlich der Aspekt „Zwischenmenschliches“ identifiziert (z.B. fand in dem Projekt keine Teambildung statt, und es gab einen Kulturbruch zwischen Skynet und Cyberdine).
- Die Qualitätsmanagement-Gruppe stellte zunächst die Mängel des Projektergebnisses dar und identifizierte dann QS-Schwächen unter den Aspekten Planung, Lenkung und Durchführung des Projektes sowie sonstige Mängel wie z.B. schlechte Lesbarkeit der Dokumente. Ausgehend von der Existenz eines Qualitätsbeauftragten und eines Qualitätsplanes wurden dann für die verschiedenen Projektphasen konkrete Verbesserungsvorschläge erarbeitet. Darüber hinaus wurden projektübergreifende und langfristige Strategien zur Qualitätsverbesserung vorgeschlagen (z.B. Knowledge Base von/für Entwickler schaffen, Maßnahmen zur Aus- und Weiterbildung).

Für die Vorbereitung der Vorträge wurden verschiedene Arbeitstechniken wie zum Beispiel Mind Mapping angewendet, und die Studierenden konnten lernen, wie man unter Zeitdruck (jede Gruppenarbeit dauerte 1-2 Stunden) präsentable Ergebnisse zustande bringt.

Die Schwachstellenanalyse wurde von den Studierenden ohne größere Schwierigkeiten durchgeführt, da in der Projektvorstellung bereits sehr viele Schwachstellen deutlich wurden. Schwieriger gestaltete sich die Erarbeitung der Verbesserungsvorschläge. Anfangs mussten die Betreuer bei den vorgeschlagenen Maßnahmen nachhaken; die Vorschläge waren oft sehr vage und allgemein. Im Laufe des Workshops haben die Studierenden jedoch von sich aus angefangen, die vorgeschlagenen Maßnahmen bzgl. der praktischen Umsetzbarkeit zu bewerten und zu hinterfragen.

5 Erfahrungen

5.1 Unsere Erfahrungen

Die Durchführung des Workshop-Seminars erforderte im Vergleich zu klassisch organisierten Seminaren größeren planerischen wie organisatorischen Aufwand, jedoch waren die erreichten Ergebnisse qualitativ besser.

Das aktive Arbeiten an einer – wenn auch fiktiven, so aber doch realitätsnahen – Aufgabenstellung im Team förderte bei den Studierenden spürbar das Interesse und damit auch die Bereitschaft zu zielgerichtetem und ergebnisorientiertem Arbeiten. Die bei klassischen Seminaren zu beobachtenden „Pflichtveranstaltungs-Effekte“ blieben bei diesem Seminar aus, was auch die Rückmeldungen der Studierenden in einer Rückbetrachtung des Seminars zum Ende der Veranstaltung zeigten. Die spürbar gesteigerte Motivation der Teilnehmer führte so auch dazu, dass sich alle Teilnehmer aktiv in den Workshop einbrachten, so dass am Ende auch allen Teilnehmern die erfolgreiche Teilnahme an dem Workshop-Seminar bescheinigt werden konnte.

Es war sehr interessant zu beobachten, wie die Studierenden mit ihrem umfassenden theoretischen Wissen auf die von den sd&m-Mitarbeitern dargestellte Wirklichkeit trafen. Sie waren erst einmal etwas vor den Kopf gestoßen (Anmerkung eines Studenten in der Kaffeepause: „Ich glaube, ich bleibe an der Uni...“), haben dann aber alltagstaugliche Maßnahmen erarbeitet.

Die abgeänderte und damit auch neue Organisationsform machte die Teilnehmer neugierig und motivierte zur aktiven Mitarbeit. Nicht zu vernachlässigen ist in diesem Zusammenhang auch die Tatsache, dass „gestandene“ Praktiker beim Workshop mitwirkten.

Das Lösen und Diskutieren von Problemen in der Gruppe zeigte die folgenden auch aus didaktischer Sicht positiven Effekte:

- Alle beteiligten sich an den Diskussionen. Auch Teilnehmer, die ansonsten zurückhaltend sind, haben sich dadurch einbringen können.
- Die Vielfalt der Meinungen und Denkanstöße zu den betrachteten Fragestellungen führte zu besseren Ergebnissen, zu einer erweiterten Sicht und zu einem gesteigerten Erkenntnisgewinn.
- Trotz der Kürze der Vorbereitungszeiten wurden beachtliche Ergebnisse erzielt.

Die realistischen und praxisorientierten Fragestellungen zusammen mit den realen Dokumenten führten dazu, dass die Teilnehmer sehr gut bewerten konnten, welche in den Lehrveranstaltungen vermittelten Inhalte eingebracht und umgesetzt werden können. Theoretisches Grundwissen und industrielle Praxisanforderungen wurden auf diese Art und Weise zusammengebracht.

Durch die engagierte Beteiligung aller Teilnehmenden (Betreuer und Studierende) entstand ein hervorragendes Arbeitsklima, das die Grundlage für die durchweg guten Ergebnisse, die an den beiden Workshop-Tagen erzielt wurden, bildete.

Die straffe Zeitorganisation an den beiden Workshop-Tagen führte dazu, dass die Gruppen ergebnis- und zeitorientiert arbeiten mussten. Dies wurde von vielen Teilnehmern als herausfordernd und nützlich bewertet. Zwei interessante Beobachtungen möchten wir hier noch hervorheben:

- Die Studierenden haben versucht, sich aller greifbaren Informationsquellen zu bedienen. Vieles von dem, was in der Einführungsveranstaltung über die Projektvorgehensweise vorgestellt wurde, floss wie selbstverständlich in die Lösungsvorschläge mit ein.
- Andere, eher offensichtliche Punkte wurden hingegen nicht entdeckt. Ein Beispiel: Das Projekthandbuch, das die Studierenden zu Beginn des Workshops erhielten, zeigte die ursprünglich geplante Aufwandsverteilung. Diese war stellenweise haarsträubend und widersprach sämtlichen Richtlinien des Software Engineering. Hierauf sind die Studierenden trotz konkreter Hinweise von Seiten der Betreuer nicht weiter eingegangen.

5.2 Feedback der Studierenden

Auch bei den Studierenden stieß diese neue Lehrveranstaltung auf eine durchweg positive Resonanz. Nachfolgend sind einige Aussagen der Studierenden wiedergegeben, die in der abschließenden Feedback-Runde geäußert wurden:

- *Die Gruppenarbeit war gut; es hat viel Spaß gemacht, es war besser als typische Seminarpräsentationen; wir haben Teamwork gelernt.*
- *Es war praxisnah, insbesondere die harten Zeitanforderungen der Gruppenarbeit und Präsentationen.*
- *Die Präsentation der Standpunkte aus verschiedenen subjektiven Sichten (im Gegensatz zu einer "objektiven" Sicht) war gut.*
- *Der Arbeitsaufwand im Vergleich zu anderen Seminaren war gleich (andere Teilnehmer behaupteten auch geringer).*
- *Danke für das Essen, das Weihnachtsgeschenk und zwei Tage Aufwand.*

5.3 Nutzen und Aufwand aus Sicht der sd&m AG

sd&m verfolgt mit dem Workshop mehrere Ziele: Neben dem ideellen Ziel, die Hochschulausbildung der Studierenden zu unterstützen, werden auch ganz konkrete Effekte erreicht. Eine solche Veranstaltung bietet die Möglichkeit, die Zusammenarbeit mit Universitätslehrstühlen zu verbessern. Durch die Aushänge und durch "Mund-zu-Mund-Propaganda" unter der Studierenden tritt ein deutlicher Werbeeffekt ein. Die Studierenden werden auf sd&m aufmerksam, auch wenn sie selbst an

der Lehrveranstaltung nicht teilnehmen. Die Veranstaltung selbst ermöglicht sd&m, Kontakt zu einem interessanten Teilnehmerkreis aufzubauen. Interessant deswegen, weil die Studierenden durch ihre Studienausrichtung ein starkes Interesse am Software Engineering zeigen. Langfristig könnten sich daraus sehr geeignete Mitarbeiter gewinnen lassen.

Der Aufwand von sd&m für den Workshop betrug für Vorbereitung und Durchführung insgesamt 19 BT (Bearbeitertage). Hiervon fielen 13 BT auf die Vorbereitung, d.h. auf Organisation, Erarbeitung des fiktiven Projekts, Anonymisierung der Projektunterlagen und Vorbereitung der Veranstaltung. 7 BT fielen auf die Durchführung, d.h. auf den Einführungsvortrag (1 BT) und den eigentlichen Workshop (2 Tage mit jeweils drei Personen). Bei einer weiteren Iteration wird sich der Vorbereitungsaufwand deutlich reduzieren (auf ca. 3 BT), der Durchführungsaufwand bleibt konstant.

Prinzipiell geeignet für eine solche Veranstaltung ist jedes Projekt, das eine nicht allzu spezielle Fachlichkeit hat, so dass sich die Teilnehmer relativ schnell in die Aufgabenstellung einarbeiten können. Da der Projektablauf fiktiv ist, spielen die Details des ursprünglichen Projektablaufs keine entscheidende Rolle. Die Anonymisierung beschränkt sich somit auf die Änderung von Namen und Bezeichnern und der Anpassung der Projektplanung an das fiktive Projekt. Auch wenn alle auszuteilenden Dokumente überarbeitet werden müssen, bleibt der Aufwand somit überschaubar.

6 Zusammenfassung

Zusammenfassend kann festgestellt werden, dass die gewählte Organisationsform, die gewählten praxisorientierten Inhalte und Problemstellungen und insbesondere die engagierte Mitarbeit aller Beteiligten dazu geführt hat, dass die Studierenden einen vertieften Einblick in die Bereiche Qualitätssicherung und Projektmanagement erhalten haben.

In dieser Lehrveranstaltung ist es uns gelungen, dass die Studierenden mit Problemen realer Softwareentwicklungsprojekte konfrontiert wurden, ohne wirklich im Projekt gewesen zu sein. Das im ersten Teil erarbeitete Fachwissen war dazu eine notwendige Voraussetzung. Im Laufe des zweitägigen Workshops konnten wir feststellen, dass sich viele Studierenden in den Eigenschaften Kommunikationsfähigkeit, Teamgeist, Begeisterungsfähig und Kreativität positiv verändert haben, was für uns wichtig war und den Erfolg der Veranstaltung ausmacht.

Da sich diese neue Form einer Lehrveranstaltung insgesamt bewährt hat, wird sie – auch auf Grund der positiven Rückmeldungen und zahlreicher Nachfragen von Studierenden – wiederholt werden.

Literatur

1. Denert, E.(2000): Was wir vom Software-Ingenieur erwarten: Gleichgewicht von Fachwissen und Persönlichkeit, sd&m-intern.
2. Lewerentz, C., H. Rust (2001): Die Rolle der Reflexion in Softwarepraktika. In Lichter, Glinz (Hrsg.) SEUH 7, dpunkt.Verlag.
3. Meier, J. (2000): Projektmodell. In Brösler, Siedersleben (Hrsg), Softwaretechnik, Hanser Verlag.
4. Ryser, J., M. Glinz (1999): Konzeption und Durchführung eines Software-Praktikums – Ein Erfahrungsbericht. In Dreher, Schulz, Weber-Wulf (Hrsg.), SEUH'99, German Chapter of the ACM Berichte 52, pp 55-68.
5. Schröder, U., M. Brunner (1999): Aktive Anwenderbeteiligung in Ausbildungsprojekten. In Dreher, Schulz, Weber-Wulf (Hrsg.), SEUH'99, German Chapter of the ACM Berichte 52, pp 103 – 116.

Software Engineering kompakt: interne Schulungen bei sd&m

Christiane Stutz, Axel Burghof
sd&m Research, software design & management AG, München
[Christiane.Stutz|Axel.Burghof]@sdm.de

Zusammenfassung

Für viele Mitarbeiter in Software-Unternehmen stellen die Möglichkeiten zur Weiterbildung ein wichtiges Kriterium für die Zufriedenheit mit ihrem Arbeitsplatz dar. Das interne Schulungsangebot eines Unternehmens muss dabei zwei Anforderungen erfüllen: Zum einen sucht der Mitarbeiter die Möglichkeit zur Weiterbildung über sein bisheriges Fachwissen hinaus. Zum anderen möchte das Unternehmen die Mitarbeiter möglichst gezielt auf zukünftige Aufgaben vorbereiten. Dieser Artikel stellt das interne Schulungswesen eines Münchner Softwarehauses vor. Wir erläutern das Konzept der so genannten Schools, die in fünf Veranstaltungen zu jeweils drei bis fünf Tagen die wichtigsten Phasen der Software-Entwicklung abdecken. Dabei handelt es sich um Schulungen, die von internen Mitarbeitern gestaltet und gehalten werden und aus einem Wechsel von Vorlesungen und Übungen zusammengesetzt sind. Die Schulungsinhalte werden an einem durchgängigen Beispiel in Gruppen geübt.

1 Einleitung

Die software design & management AG (sd&m) ist ein Softwarehaus mit Hauptsitz in München und Niederlassungen in ganz Deutschland sowie einer Niederlassung in der Schweiz. Wir entwickeln Individualsoftware für betriebliche Informationssysteme und technische Anwendungen. Unsere Projekte umfassen dabei den gesamten Entwicklungsprozess von der Anforderungsanalyse bis zur Inbetriebnahme. Dafür beschäftigten wir im Jahr 2002 ca. 950 Mitarbeiterinnen und Mitarbeiter[1], wobei nur ein relativ kleiner Teil mit Verwaltungsaufgaben betraut ist, während der überwiegende Teil der Beschäftigen bei sd&m an der Durchführung von Software-Projekten beteiligt ist. Unsere Mitarbeiter sind zu einem großen Teil studierte Informatiker. Unter den Nichtinformatikern sind naturwissenschaftliche und technische Ausbildungsrichtungen am häufigsten vertreten.

[1] Im Folgenden bleiben wir der Lesbarkeit halber für Personen bei der männlichen Form, ohne dabei unsere Kolleginnen ausschließen zu wollen.

Die meisten Mitarbeiter von sd&m bringen aus ihrer Ausbildung bereits ein gutes Rüstzeug für den Software-Ingenieur mit: Bei der Einstellung legen wir Wert auf Programmierkenntnisse (die Sprache spielt dabei nur eine untergeordnete Rolle), die Kenntnis von Grundlagen guten Softwaredesigns sowie Erfahrung in der Teamarbeit im Rahmen von Studienprojekten oder bereits Berufserfahrung. Was kann man einem derart ausgebildeten Software-Ingenieur noch beibringen?

2 Überblick über das Schulungsprogramm

Das Ausbildungsprogramm bei sd&m umfasst drei Schienen: (1) Soziale Kompetenz und Persönlichkeit, (2) Management und (3) Software Engineering. Schulungen zur sozialen Kompetenz und Persönlichkeit werden durch einen externen Veranstalter durchgeführt. Zum Management nutzen wir sowohl externe als auch interne Schulungen. Für das Software Engineering haben wir bei sd&m eine eigene interne Schulungsart eingeführt, die so genannten "Schools", die in diesem Artikel beschrieben werden.

Diese Schools werden individuell auf die Bedürfnisse von sd&m abgestimmt. Grundlagenkenntnisse in Programmierung und Software-Entwurf werden dabei vorausgesetzt. In den Schools betrachten wir besonders gute Lösungen für immer wiederkehrende Projektaufgaben und werden so auf die neuen Herausforderungen vorbereitet. Jenseits von Projektdruck können hier elegante und zukunftsweisende Lösungen entwickelt und das Vorgehen aus bereits durchgeführten Projekten reflektiert werden.

3 Konzept der sd&m Schools

Der Name "School" steht bei sd&m für mehrtägige Schulungsveranstaltungen, bei denen in einem Wechsel aus Theorie und Praxis intensiv ein Ausschnitt aus dem Software Engineering gelehrt wird.

3.1 Die richtige Mischung

Vorlesungen und Übungen finden im Wechsel statt, wobei die Übungen mindestens die Hälfte der Zeit einnehmen. Teamarbeit ist ein wichtiger Bestandteil der Übungen – wir bilden Teams aus zwei bis fünf Teilnehmern (je nach Aufgabenstellung), die gemeinsam eine Aufgabe lösen und sich die Arbeit dabei selbstständig aufteilen. Selbst auf Schulungen darf der Aufwand für die Teamfindung nicht unterschätzt werden: In der Regel arbeiten Kollegen miteinander, die sich noch nicht aus der Projektarbeit kennen und den Arbeitsstil der anderen erst einmal kennen lernen müssen. Diesem sozialen Faktor tragen wir dadurch Rechnung, dass die Teams innerhalb einer Schulung nicht häufiger als einmal wechseln.

Wichtig ist die Auswahl des richtigen Beispiels für die Übungen. Viele bekannte Schulungsaufgaben langweilen durch allzu triviale Fachlichkeit. Deshalb wählen wir eine aus realen Projekten bekannte Fachlichkeit. Den Umfang des Beispiels begrenzen wir, indem wir nur einen Ausschnitt der Fachlichkeit behandeln, diesen jedoch in seiner ganzen Tiefe. Soweit möglich werden alle Inhalte an einem durchgängigen Beispiel erarbeitet.

3.2 Zeit und Muße zum Lernen

In unsere Ausbildung investieren wir viel – vor allem Zeit. Die Dauer einer School beträgt drei bis fünf Tage. Für diese Zeit ziehen wir uns ins Kloster Zangberg zurück. Dieses Kloster ist der ideale Ort für eine Schulung: Mitten im Grünen zwischen München und Mühlheim am Inn gelegen gibt es hier keinerlei Ablenkung von der Arbeit. Die technische Infrastruktur besteht aus mitgebrachten Laptops, die bei Bedarf (z.B. in der Programmer School) durch ein lokales Netzwerk verbunden werden. Internet-Anschluss gibt es im Kloster nur über Modem an der Klosterpforte.

Das Programm einer Schulung umfasst täglich mindestens acht Stunden, das Ende für die Übungen ist offen. Und so wird auch noch mancher Abend bis spät in die Nacht zum Programmieren und Reviewen von Software-Architekturen genutzt. Dafür opfern unsere Mitarbeiter nicht nur einen ruhigen Abend im Kloster, sondern auch ihre Freizeit. Denn ein Drittel der Schulungszeit gilt als Freizeit und wird durch Überstunden oder Urlaub abgegolten.

3.3 Teilnehmer und Trainer

Schools werden von sd&m für sd&m gehalten. Die Teilnehmer sind jeweils 20-25 Mitarbeiter von sd&m, gemischt aus allen Niederlassungen. Auch die Durchführung erfolgt durch Mitarbeiter von sd&m, in Einzelfällen mit Unterstützung eng verbundener externer Experten. Wir sehen für jeweils zwei bis drei Arbeitsgruppen einen Trainer von sd&m vor. So kann man sich bei den Vorlesungen abwechseln, was für Teilnehmer wie Vortragende weniger anstrengend ist, und in den Übungen stehen immer genügend Ansprechpartner für Fragen zur Verfügung.

Die Trainer werden aus allen Unternehmensbereichen und Niederlassungen von sd&m rekrutiert. Stets ist mindestens ein Trainer dabei, der die Schulung bereits gehalten hat. Dazu kommen aber auch immer Interessenten, die an der Schulung bereits früher teilgenommen haben und ihre Erfahrungen weitergeben wollen. Es gibt keine Trainer, die nur Schulungen machen, sondern alle Trainer arbeiten auch in Projekten und haben so stets einen ausreichenden Praxisbezug.

Die Mischung von Mitarbeitern über die Grenzen von Abteilungen und Niederlassungen hinweg sowie das Zusammenbringen von erfahrenen Mitarbeitern als Trainer und weniger erfahrenen Mitarbeitern als Teilnehmer trägt wesentlich zum Knüpfen persönlicher Netzwerke in der Firma bei.

4 Schulungsinhalte

Durch die Schools werden die wichtigsten Phasen der Software-Entwicklung abgedeckt: Studie / Spezifikation, Konstruktion, Realisierung und Test.

4.1 Specification School

Die Specification School deckt in fünf Schulungstagen die frühen Phasen Studie und Spezifikation ab. Inhalte dieser School sind:

- Techniken zum Sammeln von Anforderungen und Abgrenzung des Systemkontextes
- Modellierung von Geschäftsprozessen und Anwendungsfällen
- Datenmodellierung
- Gestaltung von Benutzeroberflächen und Dialogspezifikation

Damit werden wichtige Kernthemen der frühen Phasen der Software-Entwicklung behandelt. Grundlage für die Schulungsinhalte sind die bei sd&m erarbeiteten Bausteine zur Spezifikation (siehe [1], Kapitel 7, und [3]). Diese beschreiben das Vorgehen in der Spezifikation bei sd&m sowie die Ergebnistypen. Während die Grundlagen der Modellierung in der Schulung nur sehr kurz behandelt werden, konzentrieren wir uns auf die Faktoren für eine erfolgreiche Spezifikation: Entdecken aller wichtigen Anforderungen, Konzentration auf die wesentlichen Fakten, Wahl der richtigen Darstellungsform.

Als Beispiel wird in den Übungen eine Skischule behandelt. Die organisatorischen Prozesse der Skischule (Anmeldung, Planung von Kursen, Abrechnung) stellen eine Herausforderung dar, die mit realen Projektinhalten durchaus mithalten kann.

Der letzte Tag ist der Analyse von Projekten gewidmet. Auch ohne die konkrete Fachlichkeit des Beispielprojekts zu verstehen, erkennen die Teilnehmer nun Stärken und Schwächen im Vorgehen und der Darstellung der Ergebnisse in den Spezifikationen echter Projekte, die einige Teilnehmer mitgebracht haben.

4.2 Designer School

Für betriebliche Informationssysteme, die den größten Anteil der Projekte bei sd&m ausmachen, hat die sd&m AG aus vergangenen Projekten die Standardarchitektur Quasar [2] entwickelt, die sie ihren Mitarbeitern in den fünf Tagen der Designer School vermittelt. Dabei kommen auch allgemeine Grundsätze guten Designs nicht zu kurz. In der bewährten, hälftigen Mischung aus Vorlesungen und Übungen lernen die Teilnehmer:

- Muster für das Design im Kleinen
- Trennung von Technik und Fachlichkeit bei grafischen Benutzeroberflächen
- technische Strukturierung des Anwendungskerns

- Anbindung des Anwendungskern an einen modernen Persistenzmanager
- Beherrschung von Problemen verteilter Anwendungen

Den Höhepunkt der School stellt die Projektanalyse dar. Eigens für diesen Tag eingeladene Chefdesigner von sd&m stellen ihr aktuelles Projekt vor und stehen danach bis zum Abend Rede und Antwort. Dabei diskutieren die Teilnehmer, wie die Standardarchitektur im Projekt umgesetzt und an welchen Stellen mit welcher Begründung davon abgewichen wurde.

4.3 Technische Chef-Designer (TCD) School

Während die Designer School sich an den angehenden technischen Berater wendet, besteht die Zielgruppe der TCD School aus erfahrenen Chef-Designern. Die Teilnehmer dieser School haben bereits in mindestens einem großen Projekt die Rolle des Chef-Designers übernommen. Die Anzahl der Teilnehmer ist auf 12-15 Personen beschränkt, um rege Diskussionen zwischen allen Teilnehmern zu fördern. Vorlesungen nehmen in dieser School nur noch einen sehr kleinen Raum ein. Im Vordergrund steht die Arbeit in Arbeitsgruppen. Zuerst wird in Gruppen jeweils die Architektur eines großen Software-Systems von sd&m analysiert. Hierbei geht es insbesondere darum, ob und wie die Konzepte der sd&m Standardarchitektur Quasar umgesetzt wurden.

In einer zweiten Gruppenarbeit werden Erfahrungen und Ideen rund um Themen der Konstruktion ausgetauscht, z.B. die Ergebnistypen der Konstruktion oder der Einfluss des Chef-Designers auf die Beschleunigung des Projekts. Die Ergebnisse dieser Gruppenarbeiten werden später bei sd&m konsolidiert und weiterverbreitet.

Die drei Tage TCD School dienen somit weniger dem Vermitteln von Wissen als dem Austausch von Erfahrungen und dem Reflektieren von Projektergebnissen.

4.4 Programmer School

An fünf Tagen motiviert die Programmer School ihre ca. 25 Teilnehmer zu begeistertem, zeitweise auch verzweifeltem Programmieren oft bis spät in die Nacht. Lernziel dieser School ist es, mit moderner Technologie große und langlebige Anwendungen zu bauen. Wir betrachten diese School im Folgenden etwas ausführlicher.

Als Beispiel verwenden wir in dieser School ein Telekom-Abrechnungssystem bestehend aus Kundenverwaltung, Tarifsystem und Abrechnungskomponente. Dieses System ist fachlich überschaubar und zugleich genügend komplex. Technisch betrachten wir nacheinander Architektur und Implementierung des Anwendungskerns, die Persistenz der verwalteten Daten und die grafische Benutzeroberfläche.

Eine geeignete Basis bildet die Programmiersprache Java, erweitert um eine Reihe technischer Komponenten und Entwicklungswerkzeuge, die zusammen eine professionell einsetzbare Entwicklungsumgebung bilden. Grundkenntnisse über Java werden vorausgesetzt oder ggf. vorher in einem eintägigen Crash-Kurs vermittelt. Die notwendige Hardware besteht aus einem Laptop für jeden Teilnehmer, die über

ein lokales Netzwerk mit dem Schulungsserver und über ein Modem mit dem Internet verbunden sind.

Die School beginnt mit der praktischen Umsetzung höherer objektorientierter Konzepte, insbesondere zur strikten Trennung von Zuständigkeiten im Programmcode. Ziel dieses ersten Schulungstages ist gutes Programmieren und Designen im Kleinen. Neben einem seichten Einstieg in die Entwicklungsumgebung lernen die Teilnehmer in Übungen, wie man Schnittstellen und Muster einsetzt, um Aufgaben klar zu trennen und notwendige Abhängigkeiten darzulegen.

Der zweite Tag konzentriert sich auf die Implementierung eines Anwendungskerns, also der Fachlichkeit eines betrieblichen Informationssystems, unter Einhaltung der Quasar-Standardarchitektur. Zwischen und nach den Vorlesungen vervollständigen die Teilnehmer einen lückenhaften Anwendungskern des Telekom-Abrechnungssystems.

Anders als in den ersten Übungen arbeiten die Teilnehmer hier zu zweit oder zu dritt zusammen. Um den Einstieg in die komponentenbasierte Softwarearchitektur zu erleichtern, finden diese Übungen noch ohne Anbindung von Datenbank und grafi- scher Benutzeroberfläche statt. Auskodierte Testfälle und eine Dummy-Implementierung der Datenhaltung geben den Teilnehmern schnell Rückmeldung über Erfolg oder Misserfolg ihrer Anstrengungen.

Der dritte Tag widmet sich der Verwaltung persistenter Objekte auf Basis einer relationalen Datenbank. Nach einer Einführung in den von sd&m entwickelten Persistenzmanager besteht die Aufgabe für die Teilnehmer darin, die Beschreibung persistenter Klassen für den Persistenzmanager zu vervollständigen und die Dummy-Implementierung der Datenhaltung durch eine auf den Persistenzmanager gestützte, voll funktionsfähige eigene Implementierung zu ersetzen. Dabei wird die Trennung von Datenbankanbindung und fachlichem Code im Anwendungskern mit Hilfe eines Persistenzmanagers vermittelt.

Gleich zwei Tage sind zum Entwickeln und Anbinden einer grafischen Benutzeroberfläche geplant. Dies entspricht durchaus der Verteilung des Implementierungsaufwands realer Projekte. Als Basis dient hier eine Erweiterung des Web-GUI-Frameworks Struts [5], womit als zusätzliche Sprachen HTML bzw. JSP und als zusätzliche Infrastruktur ein Web-Server einzuführen sind.

Im Anschluss an die letzte Übung hat jede Gruppe Gelegenheit, einen interessanten Aspekt ihrer Arbeit im Plenum zu präsentieren. An den vorherigen Tagen erfolgt das Feedback durch die vorgegebenen oder selbst zu kodierenden Testfälle und durch intensive Betreuung durch die drei Tutoren.

Kritik an der Programmer School richtet sich besonders an die Übungen. In vielen Übungen werden Lücken im Code gefüllt. Diese Übungen werden oft durch Kopieren von Code-Segmenten gelöst, ohne dass dabei der Zusammenhang verstanden wird. Für die Übungen planen wir deshalb eine Umstellung von Code-Gerüsten zum kompletten Kodieren einfacher Anwendungsteile. Eine ausführliche Schritt-für-Schritt-Anleitung soll die notwendigen Zusammenhänge vermitteln. Als Ausgangspunkt verwenden wir die Referenzimplementierung einer Anwendungskomponente.

Die Programmier-Übungen zu Beginn der School werden durch Kodierarbeiten am Konfigurationscode der Anwendung ersetzt, so dass insgesamt mehr Zeit für die Anwendungsprogrammierung zur Verfügung steht. Durch Hinweise auf mögliche Erweiterung werden wir Teilnehmern mit höherem Einstiegsniveau gerecht.

In der School verwenden wir eine reale, komplexe Entwicklungsumgebung:

- SUN Java JDK SE 1.3.1 oder 1.4.1 [4]
- IntelliJ IDEA 2.6 als grafisches Entwicklungswerkzeug [5]
- CVS Versionsverwaltung [6]
- JUnit Test Framework [7]
- Ant Build-Werkzeug [8]
- MySQL Datenbank [9]
- sd&m Quasar Persistence + Modeler's Workbench
- Jakarta Tomcat Web-Server [10]
- Jakarta Struts Web Application Framework [11]
- Mozilla HTML Editor

Bis auf IntelliJ ist die eingesetzte Software frei erhältlich. Alle Komponenten werden mit Ausnahme der Datenbank in professionellen Projekten eingesetzt, so dass die Teilnehmer auch in dieser Hinsicht auf zukünftige Projekte vorbereitet werden.

Die resultierende Vielzahl eingesetzter Technologien ist natürlich nicht ohne Nachteile. Die Vorbereitung einer solchen School birgt einen hohen, weil fehlerträchtigen Konfigurationsaufwand sowie erheblichen Aufwand für die Softwareverteilung. Durch die zahlreichen neuen Werkzeuge in kurzer Zeit sind viele Teilnehmer überfordert. Eine Besserung erhoffen wir uns durch ein Umschwenken in den Übungen von der komplexen Web- auf eine reine Java-Benutzeroberfläche.

4.5 Test School

Die Test School umfasst in vier Schulungstagen den ganzen Bereich des Testens von der Testplanung und -konzeption anhand der Ergebnisse früher Phasen bis zur Durchführung und Analyse der Ergebnisse (Fehler). Lehrziele sind die Schulung und Verbesserung der Mitarbeiter in Testplanung, Testverfahren, Teststufen, systematischer Testfallermittlung und Handhabung von Testwerkzeugen.

Der Ablauf orientiert sich dabei an einem Projekt. Am Anfang steht die Planung des Tests, am Ende der School der Systemtest und das Fehlermanagement. Der letzte Tag ist dem eigenen Projekt gewidmet, in dem das Gelernte direkt auf die Wirklichkeit angewendet und überprüft werden kann.

Die Test School richtet sich an alle Software-Ingenieure, (Chef-)Designer und Projektleiter, denn alle müssen sich im Rahmen der Projektdurchführung oder Angebotserstellung mit dem Thema Testen auseinander setzen.

5 Reihenfolge der Schools

Wann welche School belegt wird, ist hauptsächlich von den Erfahrungen eines Mitarbeiters und dessen Bedarf abhängig. Für die Specification School ist die Erfahrung in der Spezifikation in mindestens einem Projekt Voraussetzung. Sie soll belegt werden, wenn in näherer Zukunft Spezifikationsaufgaben für diesen Mitarbeiter anstehen. Analoges gilt für die Designer School. Für die TCD School müssen die Teilnehmer wie oben beschrieben sehr viel mehr praktische Erfahrung mitbringen und die Designer School bereits besucht haben. Die Programmer School richtet sich dagegen eher an Einsteiger mit wenig Programmiererfahrung. Wie erwähnt ist die Test School für Mitarbeiter mit unterschiedlichstem Vorwissen interessant.

Die Teilnahme an den Schulungen folgt somit keiner festen Reihenfolge, sondern wird individuell gestaltet. Gewöhnlich wird die Programmer School als erste School besucht. Danach erfolgt die Auswahl aus Designer, Specification und Test School nach Weiterbildungswunsch und -bedarf des Mitarbeiter. In der Regeln nehmen Mitarbeiter alle 1-2 Jahre an einer School teil.

6 Allgemeine Erfahrungen mit den Schulungen

Am Ende jeder School stellen sich die Veranstalter der Kritik aller Teilnehmer. Hier erhalten wir in der Regel viel Lob von zufriedenen Mitarbeitern. Natürlich gibt es daneben aber auch immer Kritik und Anregungen zur weiteren Verbesserung der School, die hier zusammengefasst werden.

6.1 Erfahrung mit dem Ablauf

Positive Resonanz erhält die Gestaltung aus Vorlesungen und Übungen und die Gruppenarbeit.

Die Übungen werden von Teilnehmern und Trainern als Mittel zur Vermittlung praktischen Wissens sehr geschätzt. Problematisch ist die Bemessung der Übungszeit. Oft können Übungsaufgaben aus Zeitmangel nicht zur Zufriedenheit der Teilnehmer fertiggestellt werden. Auch wünschen sich die Teilnehmer Zeit, um nach einem Feedback ihr Ergebnis korrigieren zu können. In der Specification School haben wir bereits mit einer Übung in zwei Teilen gute Erfahrungen gesammelt. Dabei erhalten die Gruppen nach einem ersten Teil der Übung ein Feedback und haben danach im zweiten Teil der Übung Gelegenheit, dieses Feedback einzuarbeiten.

Der Umgang mit den teilweise sehr komplexen Übungsaufgaben ist unterschiedlich. Während ein Großteil der Teilnehmer die Herausforderung dieser schweren Aufgaben schätzt, fühlen sich andere Teilnehmer überfordert und demotiviert, wenn sie die Aufgabe nicht in der gegebenen Zeit lösen können.

Neben den Übungen werden die Projektanalysen, in denen die Schulungsinhalte an realen Projekten reflektiert werden, als besonders wertvoll empfunden. Je nach

Thema der School wird hier die Art der Spezifikation, das Design oder die Teststrategie unter die Lupe genommen. Bei einer solchen Analyse werden den Teilnehmern nicht nur anhand eines Beispiels die Inhalte der School nochmals vor Augen geführt. Insbesondere trainieren sie dabei auch die Beurteilung von Projekten anhand weniger Unterlagen in kurzer Zeit – eine Fähigkeit, die in der weiteren Arbeit bei Reviews und in der Angebotserstellung von hohem Wert ist.

6.2 Erfahrungen mit der Gruppenarbeit

Die Arbeit in Gruppen ist ein wichtiger Erfolgsfaktor der Schools. Durch die eher kleinen Gruppen trägt jeder merklich zum Ergebnis bei und feiert seine privaten Erfolge. Die Ergebnisse werden abwechselnd präsentiert, so dass jede Gruppe mindestens einmal ihr Ergebnis vor den kritischen Augen der Kollegen verteidigen muss. Teilnehmer wie Trainer sind sich einig, dass sie in diesen Feedback-Runden am meisten lernen, weil die eigenen Lösungen anhand der vorgestellten Lösung hinterfragt werden.

Für die Zufriedenheit mit der Gruppenarbeit ist besonders die Gruppengröße und die Betreuung wichtig. Wir haben auch mit einer School von über 60 Teilnehmern Erfahrung gesammelt. Hier wurden 15 Gruppen gebildet, die von 9 Trainern betreut wurden. So stand stets ein Ansprechpartner zur Verfügung. Entscheidend für den Erfolg war hier, dass für das Vorstellen der Ergebnisse immer drei Gruppen zusammengenommen wurden, so dass jede Gruppe ihr Ergebnis vorstellen und diskutieren konnte.

6.3 Niveau und Projektnutzen

Wie am Feedback zu Vorlesungen und Übungen deutlich wird, ist es schwierig, das richtige Niveau für die Schulungen zu treffen. Das Niveau der Einstiegsvorlesungen wird in allen Schools von vielen Teilnehmern als zu niedrig empfunden. Dies liegt zum Teil an einem sehr unterschiedlichen Erfahrungshintergrund der Teilnehmer. Hierin äußert sich auch ein Problem im Anmeldungsprozess für die Schulungen. Einerseits sind die Lehrinhalte einer School den Teilnehmern bei der Anmeldung nicht transparent genug. Andererseits werden auch die geforderten Voraussetzungen nicht klar genug dargestellt. Hinzu kommt, dass Anmeldungen selten abgelehnt werden. Hier fehlt es an einem Auswahlprozess, der Teilnehmer nicht nur nach deren Schulungswünschen und freien Plätzen, sondern auch nach den Erfahrungen der Teilnehmer den Schulungen zuordnet.

Ein weiterer Ansatz zur Verbesserung besteht in einer Vergrößerung des Angebots mit differenzierten Inhalten. So wurde die TCD School erst im Jahr 2002 in das Programm aufgenommen, um erfahrenen Mitarbeitern eine angemessene Fortbildung zu bieten. In 2003 wird dieses Angebot durch die Designer Refresher School erweitert, die zwischen Designer und TCD School einzuordnen ist.

Der zukünftige Nutzen der Schulungsinhalte in Projekten wird von den Teilnehmern allgemein als gut eingestuft. Dies hängt jedoch wieder stark von der Art der

Projekte ab, in der die Teilnehmer mitarbeiten. Die Inhalte von Specification School und Test School sind in jedem Projekt anzuwenden, das die jeweilige Phase durchläuft. Die Inhalte der Programmer, Designer und TCD School sind dagegen stark auf objektorientiertes Vorgehen ausgerichtet. Die Übertragung in beispielsweise Hostprojekte fällt einigen Mitarbeitern schwer.

Während wir bisher versucht haben, die Schulungen unabhängig von einer bestimmten Technologie zu halten, erkennen wir zunehmenden Bedarf nach Schulung in konkreten Technologien, um Mitarbeiter auf die Projekte besser vorzubereiten.

7 Fazit

Mit den Schools bietet sd&m seinen Mitarbeitern eine Reihe interessanter Schulungen an, die alle Phasen der Software-Entwicklung abdecken. Die wichtigsten Erfolgsfaktoren dieser Schulungen sind ein Wechsel von Vorlesungen und Übungen, das Arbeiten in kleinen Gruppen, die Behandlung realistischer Beispiele und die Analyse eigener Projekte. Die Schools sind ein erfolgreiches Modell der Schulung, in dem auch der ausgebildete Software-Ingenieur Nutzen für seine Arbeit erlangt.

Um die Schulungen interessant zu halten, werden die Schulungsinhalte jährlich entsprechend der sich weiterentwickelnden Standardarchitektur und -methodik angepasst. Mit einer zunehmend technischen Ausrichtung auf Komponenten und Werkzeuge begegnet sd&m dem wachsenden Marktdruck und den komplexer werdenden Technologien. Um das Verständnis größerer Zusammenhänge zu verbessern, werden die Aufgaben weiter in Richtung Design bzw. Implementierung kompletter Teilsysteme umgestaltet. Durch zusätzliche Schools wird das Angebot inhaltlich stärker auf den Bedarf der Teilnehmer ausgerichtet.

Literatur

1. Siedersleben, J. (Hrsg.): Softwaretechnik. Hanser, München, 2002, 2. Auflage
2. Siedersleben, J. (Hrsg.): Quasar: Die sd&m Standard-Architektur. sd&m Research, 2002
3. Stutz, C.; Siedersleben, J.; Kretschmer, D.; Krug, W.: Analysis beyond UML. Proceedings of IEEE Requirements Engineering Conference 2002
4. http://java.sun.com Java
5. http://www.intellij.com IntelliJ IDEA Entwicklungsumgebung
6. http://www.cvshome.org CVS Versionsverwaltungssystem
7. http://www.junit.org A regression testing framework for Java
8. http://jakarta.apache.org/ant Java-basiertes Build-Werkzeug
9. http://www.mysql.de Relationales DBMS
10. http://jakarta.apache.org/tomcat/index.html Web-Server bzw. Servlet-Container
11. http://jakarta.apache.org/struts/index.html A framework for building web applications

Räumlich verteilte Software-Entwicklung unter experimenteller Betrachtung verteilter Inspektionen – Ein Erfahrungsbericht

Claudia Schlumpberger
Abt. Software-Prozessgestaltung
DaimlerChrysler Forschungszentrum Ulm
Postfach 23 60
89081 Ulm
claudia.schlumpberger@daimlerchrysler.com

Dietmar Ernst
Abteilung Programmiermethodik und Compilerbau
Fakultät für Informatik, Universität Ulm
89069 Ulm
dernst@uni-ulm.de

Zusammenfassung

Im Rahmen einer bereits langjährig bestehenden Kooperation zwischen dem DaimlerChrysler Forschungszentrum Ulm und der Universität Ulm wird in jedem Semester interessierten Studierenden des Informatik-Hauptstudiums ein Praktikum zum Thema "Experimentelles Software Engineering für eingebettete Systeme" angeboten. Gestaltet wird dieses von einem Mitarbeiter des DaimlerChrysler Forschungszentrums und einem Mitarbeiter der Universität Ulm, Fakultät für Informatik. Jedes Praktikum fokussiert ein anderes Thema aus dem Bereich des Software Engineering mit entsprechend unterschiedlichen Inhalten und Fragestellungen.

Im Sommersemester 2002 lag der Schwerpunkt des Praktikums auf der Betrachtung räumlich verteilter Entwicklungen und im Rahmen von entwicklungsbegleitenden Qualitätssicherungsmaßnahmen in der Durchführung von räumlich verteilten Inspektionen unter Zuhilfenahme eines Videokonferenzsystems.

Der Leser erhält einen kurzen Überblick über Ziel und Aufbau dieses Praktikums, dessen Vorbereitung und Durchführung sowie einen Einblick in die hierbei gewonnenen Erfahrungen.

1 Einleitung

Industrielle Forschungseinrichtungen wie das DaimlerChrysler Forschungszentrum Ulm sind hauptsächlich auf die industrienahe Forschung ausgerichtet und bringen daher sehr viele Fragestellungen aus den einzelnen Entwicklungsbereichen mit sich.

Im Rahmen von Kooperationen mit universitären Einrichtungen sollen diese Fragestellungen aufgenommen und anhand empirischer Untersuchungen in Form von angebotenen Praktika untersucht werden.

1.1 Motivation und Zielsetzung

Der Einsatz von Software in Form von eingebetteten Systemen – vor allem im verkehrstechnischen Bereich – spielt eine immer größere Rolle. Auf diese Herausforderung muss auch der Hochschul-Nachwuchs entsprechend vorbereitet werden. So organisieren je ein Mitarbeiter des DaimlerChrysler Forschungszentrums Ulm, Abteilung „Software Prozessgestaltung“ und der Universität Ulm, Abteilung „Programmiermethodik und Compilerbau“, seit 1996 jedes Semester ein Praktikum zum übergeordneten Thema „Experimentelles Software Engineering für eingebettete Systeme“. Dabei werden in Anlehnung an aktuelle industrielle Forschungsfragen wechselnde Aktivitäten mit interessierten Informatikstudierenden aus dem Hauptstudium durchgeführt. So wurden in den vergangenen Praktika u.a. folgende Themen behandelt:

- Vergleich statischer und dynamischer Softwareprüfungen in eingebetteten Systemen
- Textuelle Spezifikation versus modellbasierte Spezifikation
- Partielle Prüfung umfangreicher Systeme
- Design und Implementierung mittels strukturierter und objektorientierter Methoden im Hinblick auf eingebettete Systeme

Ziel ist es, die hierbei gewonnenen Erkenntnisse in die industrielle Praxis rückzuspiegeln, um einen Beitrag zu deren Verbesserung zu leisten. Aus Sicht der Lehre sollen die am Praktikum beteiligten Studierenden für Themen aus der Praxis sensibilisiert werden und neue Lösungsansätze kennen lernen.

1.2 Einbettung in das Curriculum

Während des Informatik-Grundstudiums an der Universität Ulm besuchen alle Studierenden im 3. Fachsemester das „Softwaregrundpraktikum“ [1]. Als Lehrziele stehen dabei die Vermittlung erster Kenntnisse und Erfahrungen bei der methodischen Entwicklung großer Software-Systeme im Team im Vordergrund.

Im Hauptstudium bildet die Vorlesung „Softwaretechnik“ die Basis für das Vertiefungsgebiet „Programmiermethodik“. Weitere, darauf aufbauende Vorlesungen in diesem Gebiet sind „Requirements Engineering“, „Software Qualität“ und „Management von Softwareprojekten“. Parallel zu diesen Vorlesungen wird empfohlen, eines der angebotenen Praktika, u.a. das „Experimentelle Software Engineering für eingebettete Systeme“ zu besuchen, um die in den Vorlesungen erlangten Kenntnisse praktisch zu erproben und zu vertiefen.

2 Vorbereitungsphase

Dieses Kapitel beschreibt die Themenfindung und Planung des Praktikums im Sommersemester 2002. In diesem Zusammenhang wird auch die Verknüpfung zu einem aktuellen Thema aus der industriellen Praxis gezogen.

2.1 Themenfindung

In einem multinationalen Konzern wie DaimlerChrysler findet die Produktentwicklung nicht nur lokal begrenzt, sondern weiträumig verteilt statt. So müssen die Entwickler über weite Strecken und auch Kontinente miteinander kommunizieren.

Ebenso wie der Hardware-Entwicklungsprozess ist auch der Software-Entwicklungsprozess als ein komplexes Gefüge aus verschiedenen internen Entwicklungsmannschaften und externen Zulieferern zu betrachten. Diese wiederum nehmen oft gleichzeitig Auftraggeber- und Auftragnehmerrolle ein, wobei auch diese Rollen häufig wechseln.

So werden einzelne Subsysteme verteilt an den diversen Standorten entwickelt und sind anschließend zu einem funktionsfähigen Gesamtsystem zu integrieren. In der Regel treten dabei folgende Probleme auf:

- Unklarheiten bei der Spezifikation, Realisierung und Integration
- Hohe Aufwände für Kommunikation und viele Missverständnisse

Hohe Organisations- und Zeitaufwände für Reviews/Inspektionen

Alle Beteiligten haben jedoch das gemeinsame Ziel, ein qualitativ hochwertiges Software-Produkt im geplanten Zeit- und Kostenrahmen zu entwickeln. Es ist jedoch offensichtlich, dass oben genannte Faktoren die Erreichung dieses Ziels erschweren.

Aus diesen Überlegungen heraus entstand die Idee, im Rahmen des Software-Praktikums “Experimentelles Software Engineering für eingebettete Systeme” zu beobachten, welche Hindernisse, Probleme oder weitere Fragestellungen sich bei einer räumlich verteilten Entwicklung und der Durchführung von räumlich verteilten Inspektionen ergeben.

2.2 Planung des Praktikums

Folgende Punkte wurden in der Planungsphase zum Praktikum festgelegt:

Zeitlicher Rahmen. Das Praktikum wird mit einer Gesamtdauer von 12 Wochen und einem Umfang von 4 Semesterwochenstunden (SWS) angelegt. Da der Gesamtarbeitsumfang ungefähr doppelt so hoch wie die SWS sein sollte, wird für das Praktikum eine wöchentliche Arbeitsbelastung für die Studierenden von 8 Stunden veranschlagt.

Maßnahmen zur Qualitätssicherung. Die Teilnehmer sollen im Rahmen des Praktikums für das Thema „Software-Qualitätssicherung“ sensibilisiert werden. Im Zeitplan sind hierfür Slots zur Vermittlung theoretischer Grundlagen

zu reservieren. Weiterhin finden nach jeder wichtigen Entwicklungsphase gegenseitige Reviews [2] der einzelnen Subsysteme statt. Auch hier ist ausreichend Zeit für die Vor- und Nachbereitung der Reviews einzuplanen. Für die technische Umsetzung räumlich verteilter Reviews wird entweder ein Videokonferenzsystem oder Netmeeting in Erwägung gezogen.

Festlegen einer Spezifikation für das zu entwickelnde System. Im Rahmen früherer Praktika betrachtete man eingebettete Systeme aus sehr unterschiedlichen Anwendungsgebieten, wie z.B. eine Mikrowelle, einen Anrufbeantworter, einen Kopierer, eine Parkhausschranke oder die Innenlichtsteuerung eines PKWs. In diesem Praktikum wurde jedoch ein etwas umfangreicheres System als die bisher verwendeten benötigt. Deshalb erweiterte man die früher verwendete „Parkhausschranke“ zu einer kompletten Parkhaussteuerung und erstellte eine entsprechende textuelle Systembeschreibung [3]. Dieses System lässt sich auch sehr gut in verschiedene Subsysteme unterschiedlicher Größe zerlegen.

Auswahl einer geeigneten Entwicklungsumgebung. Die Erfahrungen der früheren Praktika haben gezeigt, dass die Studentengruppen meist sehr heterogen in Bezug auf Erfahrungen mit Programmiersprachen und Entwicklungsumgebungen sind. Um ein einheitliches Basislevel für alle Teilnehmer zu schaffen, fällt die Auswahl auf Statemate, ein Werkzeug, das es erlaubt, ein simulationsfähiges Modell mittels Statecharts [4] zu erstellen. Dieses Werkzeug kommt im Bereich der Abteilung „Programmiermethodik und Compilerbau“ vielfach zur Anwendung. Damit ist auch sichergestellt, dass die Grundlagen den Teilnehmern nach kurzer Einarbeitungszeit vermittelbar sind.

Festlegen der Entwicklungsphasen. In der zur Verfügung stehenden Zeit nach Punkt 1 ist eine Komplettentwicklung eines umfangreicheren Systems gemäß Punkt 3 zu aufwändig. Daher beschränkte man sich auf die Analyse- und Entwurfsphase mit dem Ziel eines ablauffähigen und testbaren Modells.

Praktikumsziele. Die verteilte Entwicklung soll anhand folgender Punkte bewertet werden:

- Zielerreichung im vorgegebenen Zeitrahmen und mit erwarteter Qualität
- Entwicklungs- und Testaufwände in späteren Entwicklungsphasen
- Detaillierungsgrad und Angemessenheit der gegebenen Informationen
- Motivation und Engagement der Beteiligten bei der Entwicklung der Subsysteme und im Rahmen der durchgeführen Reviewsitzungen
- Als Vergleichspunkte sollen u.a. Erfahrungswerte aus früheren Praktika dienen.

Das Ergebnis der Planung ist aus der Abbildung 1 ersichtlich.

Eine Schwierigkeit in der Planung und Vorbereitung des Praktikums besteht darin, dass die Teilnehmerzahl nur schwer vorherzusagen ist, da sich Studierende oft erst zu Beginn des Praktikums anmelden. Daher wurde das Praktikum zwar mit zwei Gruppen geplant, aber auch eine dritte Gruppe wäre problemlos möglich gewesen.

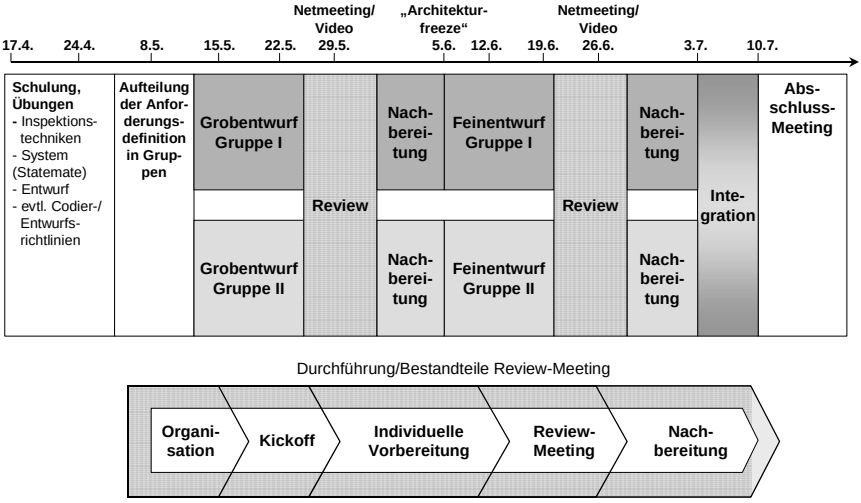


Abb. 1: Konzept des geplanten Ablaufs

3 Durchführung des Praktikums

3.1 Teilnehmende Studierende

Letztendlich haben 5 Studentinnen an dem Praktikum teilgenommen. Vier von ihnen waren im 6. Semester, also eher zu Beginn des Hauptstudiums, und haben parallel zum Praktikum die Vorlesung „Softwaretechnik“ gehört. Eine Studentin war im 8. Semester.

Die Studentinnen hatten zwar Grundkenntnisse über Zustandsautomaten aus der Vorlesung “Theoretische Informatik”, aber keine hatte bisher mit Statemate gearbeitet. So wurden die Teilnehmerinnen an den ersten Terminen anhand von praktischen Übungen mit der Entwicklungsumgebung und den Besonderheiten von Statemate vertraut gemacht. Die Zeit zwischen den Treffen mussten die Teilnehmerinnen dazu nutzen, eigenständig kleinere Aufgaben zu lösen. In den Treffen selbst wurden dann aufgetretene Probleme und mögliche Lösungen aufgezeigt.

Generell war gewährleistet, dass Rücksprachen mit den Betreuern auch außerhalb der festgelegten Termine möglich waren. Weiterhin wurde eine gemeinsame E-Mail-Plattform eingerichtet, unter der während des gesamten Praktikums organisatorische Änderungen, Fragen oder Probleme eingestellt werden konnten.

3.2 Einblick in die Systemspezifikation

Die Systemspezifikation zum System “Parkhausschranke” beschreibt in Anlehnung an Parkhäuser, wie sie in Städten oder größeren Einkaufszentren zu finden sind, ein

fiktives Parkleitsystem. Kunde bzw. Auftraggeber ist das Kaufhaus “Easybuy”, welches für seine Kunden ein neues Parkhaus erbaut hat. Die Zufahrt erfolgt über ein Schrankensystem mit Chipausgabe, die Ausfahrt erfolgt erst nach Bezahlung der Parkgebühr an einem Kassenautomaten und Entgegennahme des Chips an der Ausfahrtsschranke. Erschwerend kommt hier allerdings hinzu, dass es aus baurechtlichen Gründen nur eine gemeinsame Ein- und Ausfahrt gibt, die über ein System aus Induktionsschleifen und Ampeln geregelt werden muss.

Im Parkhaus selbst muss es auch eine Zentrale (Operator) geben, die den ganzen Ablauf der Zufahrt, Bezahlung und Ausfahrt kontrolliert und in Notfällen auch einen manuellen Eingriff in das System zulässt.

3.3 Gruppeneinteilung (Aufteilung in Subsysteme)

Nach der erfolgreichen Einarbeitung in die Entwicklungsumgebung erhielten die Teilnehmerinnen die Anforderungsspezifikation zum System „Parkhausschranke“ mit der Aufgabe, sich ein Konzept zum Zerlegen des Systems in Subsysteme und einer möglichen Aufteilung in Entwicklergruppen zu machen.

Die Teilnehmerinnen formierten sich eigenständig in eine Zweiergruppe, die für das System “Bediencontrol” (Kassenautomat und Operator) verantwortlich war, und in eine Dreiergruppe, die die Entwicklung des Systems “Schrankencontrol” (Ein- und Ausfahrt, Ampelregelung und Induktionsschleifen) übernahm.

Zusätzlich gab es noch das Subsystem „Chipverwaltung“, das von den Betreuern entwickelt wurde, da dieser Teil einerseits detailliertere Statemate-Kenntnisse erforderte und andererseits die zu entwickelten Subsysteme sonst zu umfangreich gewesen wären. Auch die komplette Hardwareumgebung, die für eine spätere Simulation nötig war, wurde von Seiten der Betreuer modelliert.

3.4 Verteilte Entwicklung der Subsysteme

Beide Gruppen entwickelten ihre jeweiligen Subsysteme völlig autark und ohne gegenseitige Nebenabsprachen. Vor allem in der Anfangsphase der Entwicklung waren aber die wöchentlichen Treffen wichtig, um die Schnittstellen zwischen den Subsystemen genauer zu definieren. Die Treffen dienten hauptsächlich dem Erfahrungsaustausch bezüglich Problemen mit der Entwicklungsumgebung, der Vorbereitung auf Reviews und Terminabsprachen, aber auch dem gegenseitigen Kennenlernen.

Ein Termin wurde dazu genutzt, die Gruppen mit der Erstellung eines Qualitätsmodells vertraut zu machen. Hier musste jede Entwicklergruppe für ihr eigenes System jeweils aus Auftraggeber- und Auftragnehmersicht Qualitätsmerkmale priorisieren. Anhand dieser Priorisierung wurden von den Betreuern Checklisten erstellt, die den Gruppen als Hilfsmittel zur Inspektion des jeweils anderen Subsystems im Rahmen der Reviewvorbereitung zur Verfügung gestellt wurde.

3.5 Organisation der verteilten Inspektionen/Reviews

Bevor das jeweils andere Subsystem inspiziert wurde, hatten die Entwicklergruppen die Aufgabe, gemeinsam einen Termin zum „Modell-Freeze“ festzulegen, um sicherzustellen, dass vor dem Review keine Änderungen mehr in die Modelle einfließen. Allen Beteiligten wurden vorab Blanko-Protokolle in Form von Einzelprüfberichten ausgehändigt, auf denen sie die bei der Inspektion gefundenen Mängel notieren sollten. Im Review selbst wurden diese dann von einem Protokollanten, dessen Rolle einer der beiden Betreuer übernahm, in einem gemeinsamen Reviewprotokoll auf einem separaten Notebook zusammengeführt. Die Moderation übernahm jeweils der andere Betreuer.

Bei der Durchführung der räumlich verteilten Reviewsitzungen kamen die umfangreichen technischen Mittel des Universitätsrechenzentrums Ulm zur Anwendung. Hier bestand die Möglichkeit, ein Videokonferenzsystem, bestehend aus Videokameras und -monitoren – wahlweise steuerbar über ISDN oder Internet (TCP/IP) – zu nutzen.

Damit beide Gruppen die zu besprechenden Teile des Modells gleichzeitig erfassen können, wurde je ein Notebook verwendet und das Programm VNC (Virtual Network Computing) [5] von den AT&T Research Labs Cambridge eingesetzt. Dieses kostenlose Programm ermöglicht es, einen Server-Desktop auf mehrere Client-Rechner zu projizieren, so dass der Desktop und die darauf ablaufenden Programme von jedem der Clients gesteuert werden können. Während der Reviews war es also möglich, dass beide Gruppen von ihrem Notebook das Werkzeug Statemate bedienten und dass die jeweils andere Gruppe dies mitverfolgen konnte.

Um das sich gerade im Review befindende Modell für alle Beteiligten gut sichtbar zu machen, wurden an jeden Notebook zwei Beamer angeschlossen, die das Modell an einer Leinwand vergrößert darstellten. Dabei platzierte man die Leinwand direkt neben der Kamera bzw. dem Monitor. Somit war gewährleistet, dass alle Beteiligten – einschließlich Protokollant und Moderator – immer frontal zur Kamera gerichtet waren, sich also niemals umdrehen oder umsetzen mussten. Die Abbildung 2 stellt den Aufbau des Systems sowie das benötigte Equipement grafisch dar.

4 Erfahrungen

Rückblickend konnten viele Zweifel, die vorab bezüglich verteilter Entwicklung und des Einsatzes eines Videokonferenzsystems vorhanden waren, entkräftet werden. Nachfolgende Kapitel listen die wichtigsten Erfahrungen aus dem Praktikum auf.

4.1 Qualität der verteilt entwickelten Software

Oft herrschen Zweifel über die Güte verteilt entwickelter Software im Vergleich zu nichtverteilter Software. Bei entsprechenden Rahmenbedingungen wie beispielsweise die Durchführung regelmäßiger Reviews – in diesem Zusammenhang auch strikt festgelegte Termine für Softwarefreezes – sowie genügend Zeit zur Vorbereitung eines Reviews ist die Qualität verteilt entwickelter Software nicht schlechter.

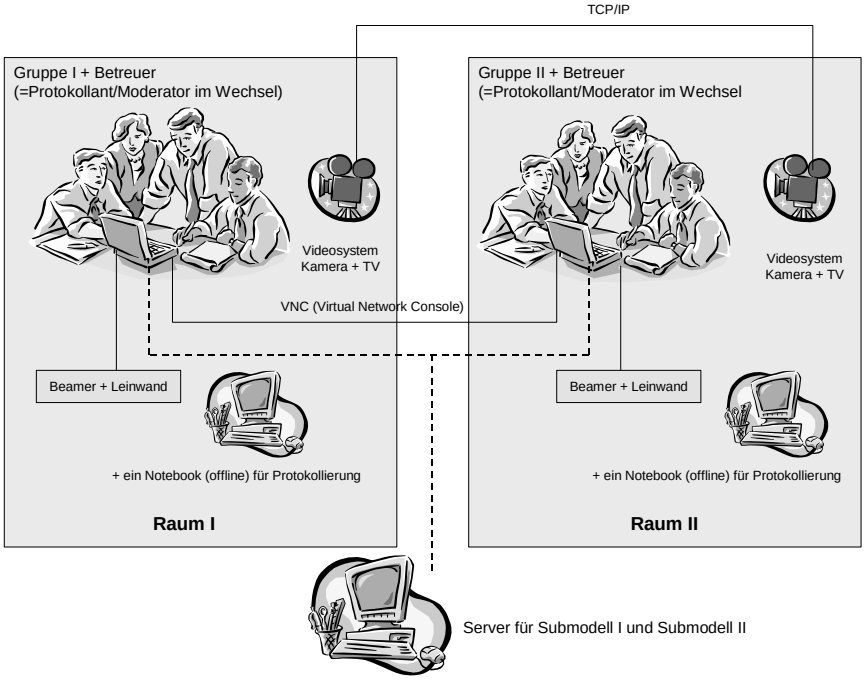


Abb. 2: Aufbau der Videokonferenz

Das Verständnis für das Gesamtsystem bleibt ebenso aufrechterhalten, denn die Einarbeitung in das jeweils andere Subsystem bei der Vorbereitung auf die Reviews und in diesem Rahmen die Bestimmung von Qualitätsmerkmalen (auch für das Fremdsystem) stärken das Gesamtsystemverständnis in beiden Entwicklergruppen. Dieser Umstand wirkt sich positiv auf die Qualität des Endproduktes aus.

4.2 Entwicklungs-, Integrations- und Testaufwand

Vor allem in der ersten Review-Sitzung, bei der die Entwicklung schon angelaufen war, wurden rückwirkend einige Unstimmigkeiten in der Spezifikation gefunden. Daraus resultierten wiederum Mehraufwände bei der Implementierung. Langjährige Untersuchungen aus dem Bereich des Software-Engineering bestätigen diesen Umstand.

Die Vermutung liegt daher auch hier nahe, dass ein zusätzliches Review der Spezifikation den Mehraufwand bei der laufenden Entwicklung gemindert hätte. Gerade bei einer verteilten Entwicklung ist eine vollständige und konsistente Spezifikation ein wesentlicher Bestandteil zur Absicherung der Qualität. Daher sollte gerade hier die Durchführung von Spezifikationsreviews (unter Umständen auch mehrere) verpflichtend sein.

Die Teilnehmerinnen entwickelten ihre “eingebetteten Subsysteme” völlig autark. Außer Event- und Variablenlisten erfolgte keinerlei Dokumentenaustausch. Negativ wurde angemerkt, dass die Bedienoberfläche, die für eine einfache und übersichtliche Simulation notwendig ist, erst spät von den Betreuern zu Verfügung gestellt wurde und deshalb die Simulation und ein Test vorher nur schwer möglich war. Schätzungen ergaben, dass sich die Testphase deshalb um den Faktor 2 verlängerte.

Die beiden Reviews der Modelle hingegen trugen wesentlich dazu bei, dass nach der Integration der Subsysteme nur noch einzelne, überschaubare Änderungen vorgenommen werden mussten.

4.3 Einsatz eines Videokonferenzsystems in Reviewsitzungen

Die vorab gehegten Zweifel, ein Videokonferenzsystem schüre die Technologieangst und mindere die Diskussionsfreudigkeit, konnte nicht bestätigt werden. Bereits beim ersten Review trat nach kürzester Zeit ein gewisser Gewöhnungseffekt ein. Besonders hervorzuheben ist, dass hier eine wesentlich diszipliniertere Diskussionskultur als bei einem On-Place-Meeting gepflegt wurde. Die Teilnehmerinnen fielen sich nicht gegenseitig ins Wort und beschränkten sich auf das Wesentliche.

In diesem Zusammenhang stellt sich auch die Frage, ob die Akzeptanz eines derartigen Systems immer noch so hoch ist, wenn es in Situationen genutzt wird, in denen eine gewisse “Medienflexibilität” gefordert ist, z.B. in Brainstorming-Situationen, in denen vieles auf Papier oder anderen Medien skizziert wird.

Obwohl der Einsatz des in diesem Praktikum eingesetzten Systems mit einem gewissen Mehraufwand bezüglich Vorbereitung, Schulung, Durchführung und vor allem technischer Mittel verbunden ist, erfüllte das System seinen Zweck optimal.

4.4 Die Entwicklungsumgebung

Statemate hat sich als Entwicklungsumgebung in diesem Praktikum hauptsächlich zur Entwicklung des Subsystems “Schrankencontrol” (da dies als ein wirkliches “eingebettetes System” bezeichnet werden kann) bewährt.

Der Lerneffekt bei den Teilnehmerinnen war hoch und sie waren in der Lage, in der doch begrenzten Zeit ein anspruchsvolles System zu modellieren. Jedoch ist der Einsatz nur bei Studierenden aus dem Hauptstudium sinnvoll, da sie erst hier Kenntnisse für die Umsetzung einer Spezifikation in Zustandsautomaten mitbringen. Dennoch ist die 3-wöchige Einlernphase notwendig. Weiterhin ist wichtig, dass während der gesamten Entwicklungszeit immer ein fachlich versierter Ansprechpartner für kurzfristige Fragen zur Verfügung steht.

5 Fazit und Ausblick

Die Erkenntnisse aus dem Praktikum zum Thema “Räumlich verteilte Entwicklung unter experimenteller Betrachtung räumlich verteilter Inspektionen“ bestätigen einerseits bereits bekannte Probleme aus dem Bereich des Software Engineering, andererseits konnte man neue Erkenntnisse im Umgang mit Entwicklungsumgebung und neuen Technologien gewinnen.

Eine Befragung der Teilnehmerinnen ergab, dass der Aufbau des Praktikums in der hier beschriebenen Form für gut befunden wurde. Als ein sehr positiver Aspekt wird hervorgehoben, dass der gesamte Entwicklungsprozess zu jedem Zeitpunkt transparent gehalten und somit kontrollierbar war. Negativ hingegen wurde bewertet, dass die Bedienoberfläche erst sehr spät geliefert wurde und sich so der Testaufwand des Gesamtsystems erhöhte. Schon die Ausgabe einer Skizze der Oberfläche vorab wäre für die Testvorbereitung hilfreich gewesen.

Zwar wurde der Einsatz von Statemate ebenso positiv bewertet, allerdings gibt es auf Seiten der Teilnehmerinnen Bedenken bezüglich des Gebrauchswertes im weiteren Verlauf des Studiums und Berufes.

Interessant ist auch die Erkenntnis der Studentinnen, dass die wöchentlichen Treffen den Revieweffekt etwas mindern, da es hier unbemerkt immer wieder zu gemeinsamen Absprachen kam. Offen bleibt dabei die Frage, wie ein solches Experiment mit strikt getrennten Gruppen, z.B. im Rahmen einer Zusammenarbeit mit anderen Universitäten, verlaufen würde.

Allgemein wird man selten die Gelegenheit haben, mit einer ausschließlich weiblichen Entwicklergruppe zu arbeiten. Die Studentinnen arbeiten normalerweise in der Mehrzahl mit männlichen Kommilitonen zusammen und empfanden in diesem Praktikum den Druck des “Sich-Beweisen-Müssens” als wesentlich weniger stark. Weiterhin wurde das Arbeiten mit den anderen Gruppenpartnerinnen als organisierter und teamorientierter empfunden.

Literatur

1. Schwarz, M.; Schulte, W.: Realistische Aufgabenstellungen für das Softwaregrundpraktikum, Workshop des German Chapter of the ACM und der Gesellschaft für Informatik: "Software Engineering im Unterricht der Hochschulen SEUH'97", Rostock, 27-28. Februar, Berichte des German Chapter of the ACM, Band 48, B.G.Teubner Stuttgart, 1997, S.94-104.
2. Frühauf, K., Ludewig J., Sandmayr H.: Software-Prüfung: Eine Anleitung zum Test und zur Inspektion, B.G.Teubner Stuttgart, 1995.
3. Houdek, F.: Softwareanforderungen und Systemspezifikation „Parkhausschranke“, DaimlerChrysler Forschungszentrum Ulm, 2002.
4. Harel, D.: Statecharts: A visual formalism for complex systems, Science of Computer Programming, Band 8, 1987, S. 231-274.
5. Richardson T.; Stafford-Fraser, Q.; Wood, K.R.; Hopper, A.: Virtual Network Computing, IEEE Internet Computing,, Band 2, Nr. 1, Jan/Feb 1998, S. 33-38.