E-Learning-Komponenten zur Intensivierung der Übungen in der Informatik-Lehre – ein Erfahrungsbericht

Dietmar Rösner, Mario Amelung, Michael Piotrowski

Otto-von-Guericke-Universität Magdeburg
Institut für Wissens- und Sprachverarbeitung
Postfach 4120
39016 Magdeburg
{roesner,amelung,mxp}@iws.cs.uni-magdeburg.de

Abstract: Übungen sind ein zentrales Element in der Informatiklehre. Ausgehend von didaktischen Überlegungen, wie der Übungsbetrieb durch Komponenten des E-Learning, insbesondere durch Formen des Computer-Aided Assessment, intensiviert und effizienter gestaltet werden kann, haben wir die eduComponents entwickelt. Dabei handelt es sich um eine Sammlung von Erweiterungsmodulen, die ein allgemeines CMS (Plone) um E-Learning-Funktionalität ergänzen. Seit mehreren Semestern werden diese frei verfügbaren Module sowohl in allen Lehrveranstaltungen unserer Arbeitsgruppe als auch an anderen Institutionen erfolgreich eingesetzt.

1 Einführung

Beim Studium der Informatik ist es für ein tiefergehendes Verständnis des Lernstoffs unverzichtbar, sich durch das eigenständige Lösen von Problemen mit den Theorien, Formalismen und den praktischen Fragen im Zusammenhang mit deren Implementierung auseinanderzusetzen. Dies kann jedoch nicht in Vorlesungen erfolgen, sondern nur in Übungen, die daher ein zentrales Element in der Informatiklehre darstellen.

Am Themenkomplex der Programmierung lässt sich diese allgemeine Feststellung besonders klar illustrieren: Kompetenz im Programmieren ist nicht in einer Vorlesung allein zu erwerben, sondern nur durch eigenständiges Lösen von Programmieraufgaben und durch die Umsetzung von Algorithmen in lauffähige Programme.

Die Frage, wie der Übungsbetrieb zu unseren Lehrveranstaltungen durch E-Learning-Komponenten, insbesondere durch die rechnergestützte Auswertung von Aufgaben (*Computer-Aided Assessment*, CAA), effektiver und effizienter gestaltet werden kann, hat uns motiviert, die in diesem Beitrag beschriebenen Softwarekomponenten (*eduComponents*) zu entwickeln. Erste Erfahrungen mit der elektronischen Einreichung und automatischen Überprüfung von Programmieraufgaben wurden im Wintersemester 2003/2004 gesammelt. Seitdem haben wir die Softwarekomponenten kontinuierlich eingesetzt und weiterentwickelt, wobei insbesondere didaktische Überlegungen und die beim Einsatz gewonnenen Erfahrungen und die Rückmeldungen der Studierenden eingeflossen sind.

2 Übungen in der Informatiklehre

2.1 Probleme

Vorlesungen werden typischerweise durch Übungen und Tutorien ergänzt, die den Studierenden die Möglichkeit geben, das in der Vorlesung erworbene Wissen anzuwenden und ihren Leistungsstand zu überprüfen.

Bisher war bei unseren Übungen das folgende Verfahren üblich: Zu Beginn der Übungsstunde *votierten* die Studierenden Aufgaben. Sie gaben damit an, dass sie in der Lage wären, ihre Lösung an der Tafel zu erklären. Für den Scheinerwerb mussten die Studierenden einen bestimmten Prozentsatz der Aufgaben votieren bzw. präsentieren.

Bei diesem System gab es allerdings einige Probleme. Jede Aufgabe wurde meist nur von jeweils *einem* Studierenden an der Tafel vorgetragen, so dass die Übungsleiter nur eine kleine Anzahl von Lösungen sahen. Typische Fehler und Probleme waren bei dieser nur stichprobenartigen Kontrolle oft schwer zu erkennen und eine Beurteilung des Leistungsstandes der Übungsgruppe nur unzureichend möglich.

Diese Situation war auch für die Studierenden unbefriedigend, da aufgrund der zeitlichen Beschränkung üblicherweise nicht alle Lösungen und Probleme besprochen werden konnten. Dadurch erhielt die Mehrzahl der Studierenden keine Rückmeldung zu ihren eigenen Lösungen, was sich negativ auf die Motivation auswirken kann.

Für die Lehrenden bedeutete schließlich auch die Verwaltung der papierbasierten Votierlisten einen zusätzlichen Aufwand, da für die Ausstellung der Scheine die Daten von diesen Formularen wieder in den Rechner übertragen werden mussten.

Außerdem kam es immer wieder vor, dass Studierende versuchten, den Schein zu erwerben, ohne die Übungsaufgaben tatsächlich zu bearbeiten, da man für unbearbeitete Aufgaben votieren und darauf spekulieren konnte, nicht aufgerufen zu werden. Eine schriftliche Abgabe der Lösungen wäre eine Möglichkeit, der Votierung für unbearbeitete Aufgaben entgegenzuwirken, allerdings muss dann zumindest eine stichprobenartige Durchsicht der Einreichungen erfolgen, damit diese Maßnahme tatsächlich effektiv ist. Die Durchsicht von auf Papier oder per E-Mail eingereichten Aufgaben hätte jedoch einen erheblichen Mehraufwand für die Lehrenden bedeutet und kam somit nicht in Frage.

2.2 Unterstützung durch E-Learning

Die Analyse der Probleme beim bisherigen Übungsablauf führte uns zu den folgenden wesentlichen Punkten: Übungen dienen zunächst zur Vertiefung des in der Vorlesung vermittelten Stoffs, insbesondere durch die selbständige praktische Anwendung und durch die Bearbeitung von Analyse-, Synthese- und Evaluationsaufgaben. Außerdem bieten sie die Möglichkeit zur Diskussion. Übungen dienen darüber hinaus auch der Überprüfung des Kenntnisstandes der Studierenden, wobei dies wiederum den Studierenden hilft, ihren eigenen Lernfortschritt einzuschätzen.

Unsere Überlegung war, dass der Einsatz von Komponenten des CAA im Zusammenspiel mit einem Content-Management-System (CMS) helfen könnte, den Übungsbetrieb zu verbessern. So lassen sich beispielsweise die Einreichung, Verwaltung und Bewertung von studentischen Lösungen zu Übungsaufgaben mittels Software unterstützen und teilweise sogar automatisieren. Auf diese Weise könnte der administrative Aufwand verringert werden, so dass die o. g. didaktischen Aspekte wieder in den Vordergrund rücken.

Seit dem Wintersemester 2003/2004 werden in unserer Arbeitsgruppe Softwarekomponenten zur elektronischen Unterstützung der Lehre entwickelt und eingesetzt.[1] Das Ziel ist es, die Lehrenden bei der Erstellung und Verwaltung von Übungsaufgaben und Tests und bei der Bewertung von studentischen Lösungen zu unterstützen. Gleichzeitig sollen für die Studierenden einfache Möglichkeiten zur Überprüfung ihres Lernerfolgs und zur Einreichung ihrer Lösungen bereitgestellt werden. In Zeiten steigender oder immerhin noch hoher Studierendenzahlen und gleichzeitigem Personalabbau ist es um so dringlicher, die für die Betreuung von Studierenden zur Verfügung stehende Arbeitszeit besonders effektiv zu nutzen, den Verwaltungsaufwand zu reduzieren und automatisierbare Teile des Übungsbetriebs auch tatsächlich zu automatisieren.

3 eduComponents

Im einzelnen bestehen die eduComponents aus folgenden Softwarekomponenten, die die o. g. Ziele unterstützen:

- **ECLecture**, ein Werkzeug zur Verwaltung von Lehrveranstaltungen und Teilnehmern;
- **ECQuiz**, ein Werkzeug zur Erstellung, Durchführung und Auswertung von webbasierten interaktiven Tests im Multiple-Choice-Format [PR05];
- **ECAutoAssessmentBox**, ein Werkzeug, das es erlaubt, *Programmieraufgaben* zu stellen, studentische Lösungen dazu automatisch anhand von durch den Aufgabensteller definierten Kriterien auszuwerten und den Studierenden unmittelbare Rückmeldungen dazu zu geben [RAP05, RA05];
- **ECAssignmentBox** ist als Verallgemeinerung von ECAutoAssessmentBox ein Werkzeug, mit dem Lehrende *beliebige Aufgaben* (d. h., ohne automatische Überprüfung) erstellen können. Ebenso wie bei ECAutoAssessmentBox wird die elektronische Einreichung der studentischen Lösungen, die Einhaltung von Einreichungsfristen und die Auswertung unterstützt [APR06].

Die entwickelten Komponenten[2] sind Erweiterungsmodule (sog. *Produkte*) für das quelloffene CMS Plone[3].

[1] Die Arbeiten wurden anfangs im Rahmen eines vom Land Sachsen-Anhalt geförderten Projekts (Förderkennzeichen 0047M1/0002A) unterstützt und werden mittlerweile durch die Mitarbeiter und studentische Hilfskräfte der Arbeitsgruppe fortgeführt.

[2] Die Software ist quelloffen und unter `http://wwwai.cs.uni-magdeburg.de/software` erhältlich; Tutorials und Videos sind unter `http://wwwai.cs.uni-magdeburg.de/forschung/projekte/educomponents/` verfügbar.

[3] `http://plone.org/`

Da sich Plone im Gegensatz zu reinen E-Learning-Plattformen für alle Arten von Webinhalten eignet, bietet die Integration der E-Learning-Funktionalität in das CMS wesentliche Vorteile: Studierende finden die Aufgaben und Tests an derselben Stelle wie die anderen Veranstaltungsmaterialien, mit gleichem Erscheinungsbild und gleicher Bedienung. Lehrende müssen nicht die Benutzung eines weiteren Systems erlernen, wenn sie mit der Bedienung des CMS bereits vertraut sind. Systemadministratoren schließlich müssen neben einem Webserver kein weiteres System verwalten, insbesondere müssen die Benutzer in nur einem System verwaltet werden.

Die Module können sowohl einzeln als auch gemeinsam eingesetzt werden und sind beliebig mit anderen Plone-Produkten kombinierbar. Dadurch ist die Einstiegsschwelle niedrig (man kann z. B. zunächst nur einige MC-Tests anbieten) und Anwender können sich individuelle E-Learning-Umgebungen nach ihren Bedürfnissen zusammenstellen.

Die Verwendung von quelloffener Software als Basis ist nicht nur wegen des Entfalls von Anschaffungs- und Lizenzkosten und der Unabhängigkeit von einem Hersteller vorteilhaft, sondern auch in technischer Hinsicht interessant.

Abbildung 1: Die Anzeige eines Testergebnisses in ECQuiz.

3.1 Multiple-Choice-Tests in der Informatiklehre

Multiple-Choice-Tests (kurz: *MC-Tests*) werden in einigen Fächern (z. B. Medizin oder BWL) regelmäßig eingesetzt, während sie in anderen Fächern auf Vorbehalte stoßen. Wir denken jedoch, dass insbesondere interaktive webbasierte MC-Tests auch in der Informatiklehre eine hilfreiche Funktion haben können. Wir setzen sie daher seit mehreren Semestern als *formative Tests* – also studienbegleitend – in unseren Lehrveranstaltungen ein. Sie dienen dabei als Bindeglied zwischen der Vorlesung und den Aufgaben der Übungen (vgl. Abb. 1).

Ähnlich wie kleine Aufgaben und Verständnisfragen in guten Lehrbüchern können MC-Tests die Studierenden zur Auseinandersetzung mit dem Stoff animieren und so zum besseren Verständnis von Definitionen und Begriffen, aber auch von Formalismen und Verfahren beitragen. MC-Tests können dabei nicht nur *Wissen* (im Sinne der Bloomschen Taxonomie von Lernzielen [BEFH76]) abfragen, sondern, wenn sie entsprechend konzipiert sind, durchaus auch höhere kognitive Leistungen wie *Analyse* oder *Evaluation* von den Studierenden abfordern.

Formative MC-Tests unterstützen die Kontrolle des Lernfortschritts im Sinne eines kontinuierlichen Monitoring. Lehrende erhalten Übersichten zum Abschneiden aller Studierenden bei den einzelnen Fragen. Wird dadurch erkennbar, dass bei bestimmten Fragestellungen oder Themen Probleme auftreten, so kann in der Vorlesung oder in den Übungen vertieft auf die jeweilige Problematik eingegangen werden.

Das eduComponents-Modul ECQuiz unterstützt die genannten Aspekte: Lehrende können bequem Tests mit Einfach-, Mehrfach- und Textantwortfragen erstellen und auswerten. Durch die Integration in Plone können Tests wie andere Inhaltstypen verwaltet werden.

3.2 Automatische Überprüfung von Programmieraufgaben

Bereits seit den 1960er Jahren werden Systeme zur automatischen Überprüfung und Benotung von studentischen Lösungen zu Programmieraufgaben entwickelt [FW65]. Die Motivation für diese Systeme sind die meist großen Studierendenzahlen in den Anfängerveranstaltungen (siehe z. B. [GVN02, MKS02, DW04]). Unserer Ansicht nach besteht ein weiterer großer Nutzen bei der automatischen Überprüfung von Programmieraufgaben darin, mit relativ geringem Aufwand eine große Zahl von Programmieraufgaben stellen zu können und den Studierenden somit zu mehr Programmierpraxis zu verhelfen.

Das eduComponents-Modul ECAutoAssessmentBox stellt die dafür notwendige Infrastruktur zur Verfügung. Die Begutachtung von studentischen Lösungen kann als Prozess gesehen werden, der mit der Einreichung beginnt und mit der Benotung endet. Im CMS Plone ist jedem Inhaltstyp ein Workflow zugeordnet. Wir nutzen diese Möglichkeit, um einen spezialisierten Workflow für die Einreichung und Bewertung von Übungsaufgaben zu definieren. Die grundlegende Idee ist, dass Studierende ihre Lösungen zu Übungsaufgaben einreichen und diese dann mehrere Workflowzustände durchlaufen, üblicherweise »eingereicht«, »akzeptiert« und »benotet«. Lehrende können sich die Einreichungen ansehen, Noten zuweisen

und Kommentare anfügen. Eine Einreichung wird typischerweise dann akzeptiert, wenn das Programm lauffähig ist und die Tests erfolgreich durchlaufen hat. Die Vorteile liegen in der zentralen Verfügbarkeit von Aufgaben und Einreichungen sowie den klar definierten Abläufen und Strukturen.

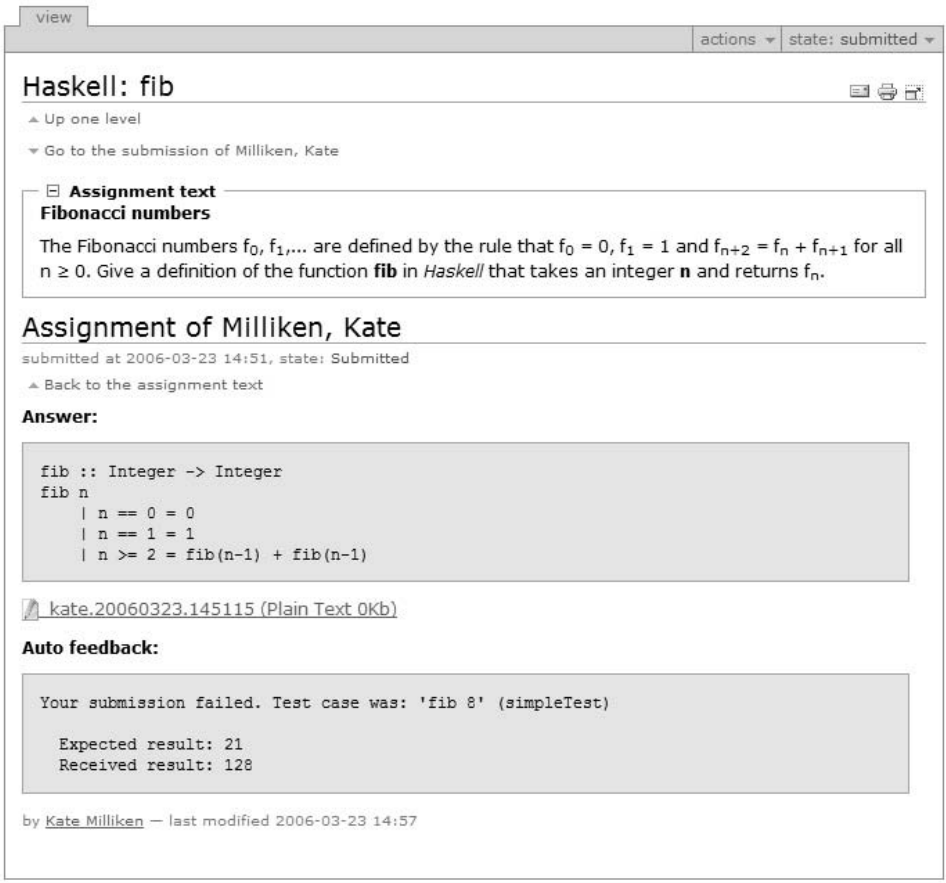

Abbildung 2: ECAutoAssessmentBox überprüft Einreichungen zu Programmieraufgaben automatisch und liefert sofort eine Rückmeldung (studentische Sicht).

Jede Aufgabenstellung besteht im wesentlichen aus einem Titel und dem Aufgabentext. Zusätzlich können Aufgabensteller festlegen, in welchem Zeitraum Einreichungen erlaubt sind und ob sie bei neuen Einreichungen per E-Mail benachrichtigt werden möchten.

Darüber hinaus kann eine Antwortvorlage hinterlegt werden. Antwortvorlagen können bei der Lehre zu Programmierung und Programmiersprachen zum Beispiel dann eingesetzt werden, wenn die Fähigkeit zum Verstehen und Verändern vorgegebener Programme geübt werden soll. Dazu zählt auch das Debugging vorgegebener fehlerhafter Programme. Antwortvorlagen können auch zur deutlichen Reduzierung der Komplexität dienen, indem ein Programm nicht in allen Aspekten neu geschrieben werden muss, sondern eben in einem

vorgegebenen Rahmen nur überschaubare und punktuelle Änderungen oder Erweiterungen vorzunehmen sind, das Ergebnis aber ein lauffähiges und testbares Programmm darstellt. Für den Anfängerunterricht in Java ist das besonders vorteilhaft.

Unser Ansatz unterscheidet sich von Systemen wie beispielsweise TRAKLA [LSKM04], SchemeRobo [SMK01] oder CourseMaster [HST02] durch die konsequente Trennung der verschiedenen Funktionen. Die Einreichung und Anzeige erfolgt durch ECAutoAssessmentBox (siehe Abb. 2). Die konkrete Durchführung der Überprüfung von Einreichungen ist stark abhängig von der Programmiersprache und dem verwendeten Interpreter bzw. Compiler und erfolgt in sog. *Backends*. Frontends und Backends werden durch ECSpooler verknüpft, ein von Plone unabhängiger Dienst, der eine Warteschlange für Einreichungen und die verfügbaren Backends verwaltet.

Bezüglich der Einreichung und automatischen Überprüfung von Programmieraufgaben sind auch WebAssign [BHSV99] mit dem AT(x)-Framework [BKW05] und Praktomat [KSZ02] vergleichbare Systeme. Tatsächlich bieten diese System teilweise Funktionen, die in ECAutoAssessmentBox bislang nicht verfügbar sind; der entscheidende Unterschied ist jedoch, dass WebAssign und Praktomat spezialisierte Stand-Alone-Systeme sind, während ECAutoAssessmentBox ein Modul für ein allgemeines CMS ist – mit den o. g. Vorteilen.

Bei der Erstellung werden Programmieraufgaben mit bestimmten Backends assoziiert (z. Z. für Haskell, Erlang, Scheme, Prolog, Python und Java). Wenn Studierende dann ihre Programme einreichen, werden diese an die entsprechenden Backends zur Überprüfung weitergeleitet.

Die konkrete Durchführung der Tests ist stark abhängig von der Programmiersprache und dem verwendeten Interpreter bzw. Compiler. Diese programmiersprachenspezifischen Unterschiede werden in den Backends gekapselt.

Ebenso lassen sich in den Backends unterschiedliche Testmethoden realisieren, ohne dass Anpassungen des Gesamtsystems nötig wären. Wir haben beispielsweise für Haskell zwei unterschiedliche Backends implementiert. Das eine vergleicht die Ergebnisse der eingereichten Lösung mit den Ergebnissen einer Musterlösung bezogen auf eine Menge von Testdaten. Das andere verwendet QuickCheck [CH00], um mittels zufällig generierter Testdaten zu überprüfen, ob die studentische Lösung bestimmte formal spezifizierte Eigenschaften erfüllt.

Sobald die Ergebnisse der durch das Backend durchgeführten Prüfungen vorliegen, werden diese dem Einreicher angezeigt. Ähnlich wie bei MC-Tests führt die unmittelbare Rückmeldung erfahrungsgemäß zu einer höheren Motivation der Studierenden. Wenn erwünscht, kann die unmittelbare Rückmeldung in Kombination mit der Möglichkeit zur Mehrfacheinreichung auch das »Herantasten« an eine korrekte Lösung [WW05] ermöglichen.

3.3 Verallgemeinerung: Elektronische Einreichung beliebiger Aufgaben

Die Architektur von ECAutoAssessmentBox ermöglicht es, nicht nur Programmieraufgaben zu überprüfen, sondern auch andere formale Notationen oder sogar Freitext zu analysieren

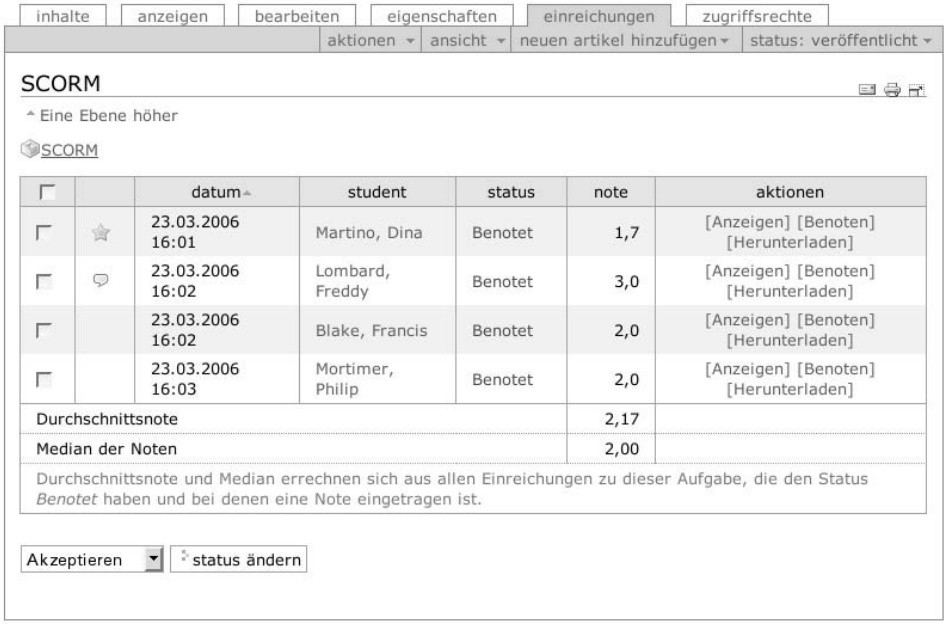

Abbildung 3: Übersicht über Einreichungen zu einer Aufgabe in ECAssignmentBox(für Lehrende).

[Feu06]. Ebenso lässt sich die Infrastruktur auch ganz ohne automatische Überprüfung nutzen, um beispielsweise Aufgaben zu stellen, bei denen textuelle (z. B. aufsatzartige) Antworten gefordert werden. Für die in diesem Fall manuelle Bewertung stehen dieselben Verwaltungs- und Statistikfunktionen wie für Programmieraufgaben zur Verfügung (vgl. Abb. 3). Diese Variante wird von dem Produkt ECAssignmentBox bereitgestellt.

4 Erfahrungen

Seit dem Wintersemester 2004/2005 werden unsere Vorlesungen durch MC-Tests ergänzt. Im Wintersemester 2005/2006 wurde in allen Übungen unserer Arbeitsgruppe das bisher übliche Verfahren zur Votierung durch die elektronische Einreichung von Übungsaufgaben mittels ECAssignmentBox ersetzt. Im Sommersemester 2006 wurde dann zusätzlich das bisher verwendete Modul zur automatischen Überprüfung von Programmierlösungen ([RAP05]) durch ECAutoAssessmentBox abgelöst. Somit werden jetzt alle in 2.2 erwähnten Ziele von Übungen durch die eduComponents unterstützt.

Seit dem Wintersemester 2005/2006 wurden die eduComponents in den folgenden Vorlesungen eingesetzt: *Dokumentverarbeitung*, *Funktionale Programmierung*, *Informationsextraktion*, *KI-Programmierung und Wissensrepräsentation*, *Lehr- und Lernsysteme*, *Natural Language Systems I*, *Natural Language Systems II* und *Programmierkonzepte und*

Modellierung sowie in Seminaren. Im laufenden Semester (Wintersemester 2006/2007) werden die Module von über 200 Studierenden genutzt.

Der Ablauf der Übungen ist nun grundsätzlich wie folgt: Die Studierenden müssen bis zu einer Einreichungsfrist – typischerweise einige Stunden vor dem Übungstermin – ihre Lösungen in ECAssignmentBox bzw. ECAutoAssessmentBox einreichen. Die Lehrenden haben dann die Möglichkeit, die Einreichungen durchzusehen und wiederkehrende Probleme oder auch besonders gute Lösungen zu finden, um auf sie in der Übung besonders eingehen zu können. Mit Hilfe der Statistikfunktionen, die den aktuellen Status aller Übungsteilnehmer zeigen, wird meist schon in dieser Phase festgelegt, welche Studierende welche Aufgaben in der Übung vortragen sollen.

In der Übung selbst werden die Aufgabenstellungen und die vorzutragenden Einreichungen direkt an die Wand projiziert. Dadurch entfällt das insbesondere bei Programmieraufgaben problematische Anschreiben an die Tafel. Die meist vorher festgelegten Studierenden werden dann aufgefordert, ihre Lösung zu kommentieren. Die Studierenden sollen ihre Lösung dann nicht einfach vorlesen, sondern stattdessen auf die der Antwort zugrundeliegenden Überlegungen und Konzepte eingehen. Insbesondere bei Programmieraufgaben hat sich gezeigt, dass durch die Tatsache, dass die Antwort vollständig und gut lesbar vorliegt, andere Studierende zu Kommentaren ermuntert werden; bei Bedarf kann dann auch schnell ein anderer Ansatz eines anderen Studierenden gezeigt werden. Wurde die Aufgabe zufriedenstellend vorgetragen, kann die Einreichung des Studierenden gleich in den entsprechenden Workflowzustand versetzt werden, so dass sich der neue Stand dann sofort in der Statistik widerspiegelt.

Um Feedback von den Studierenden zu erhalten, wurden die Übungsteilnehmer zu ihren Erfahrungen mit dem neuen System befragt. Dazu wurde ein Fragebogen erstellt, der 20 Aussagen enthält, zu denen jeweils die Zustimmung auf einer Skala von 1 (»trifft voll zu«) bis 6 (»trifft überhaupt nicht zu«) erfragt wird. Die Fragen lassen sich drei Themenbereichen zuordnen: Zunächst sollte die elektronische Einreichung im Vergleich zum bisherigen Verfahren beurteilt werden; ein Frage war zum Beispiel *Die Online-Einreichung ist meiner Meinung nach ein sehr guter Ersatz für das bisherige Votiersystem.* Dann wurde nach der Bewertung der konkreten Implementierung gefragt, z. B. *Das System bietet einen guten Überblick über die Anzahl und den Status meiner bisherigen Einreichungen*. Schließlich wurden Fragen zum Einfluss der elektronischen Einreichung auf die persönliche Arbeitsweise gestellt, ein Beispiel für eine Frage in diesem Bereich ist *Meine Lösungen sind ausführlicher als bisher*. Der Fragebogen wurde bislang von über 50 Studierenden aus unterschiedlichen Lehrveranstaltungen beantwortet. [4]

Die Studierenden empfanden es insbesondere als sehr hilfreich, dass alle Übungsaufgaben und ihre Lösungen online und an zentraler Stelle verfügbar waren. Die meisten von ihnen würden sich daher eine elektronische Einreichung auch bei anderen Veranstaltungen wünschen. Es wurde von den Studierenden nicht als Problem betrachtet, dass sie die Aufgaben jetzt bereits einige Stunden *vor* der Übung einreichen müssen. Die eduComponents und ihr Einsatz in unseren Übungen wurde von den Studierenden insgesamt mit »gut« bis »sehr gut« bewertet.

[4] Alle Fragen und die Auswertung finden sich unter http://wwwai.cs.uni-magdeburg.de/studium-und-lehre/lehrveranstaltungen/auswertung-e-learning/.

Die persönliche Arbeitsweise wird durch die Online-Einreichung offenbar positiv beeinflusst: Die Studierenden gaben mehrheitlich an, dass sie die Aufgaben gewissenhafter lösen, da alle eingereichten Lösungen vom Übungsleiter durchgesehen werden können.

Die Befragung der Studierenden und unsere Erfahrung haben gezeigt, dass elektronisch eingereichte, schriftlich ausformulierte Lösungen zu Aufgaben Vorteile für Lehrende und Lernende bieten. Dies wird besonders deutlich beim Vergleich mit der bisherigen Praxis der Übungen.

Über die oben beschriebenenen Vorteile hinaus halten wir die folgenden Punkte für wichtig: Werden Lösungen schriftlich ausformuliert verlangt, so müssen sich die Studierenden intensiver mit dem Material auseinandersetzen. Die schriftliche Aufzeichnung erlaubt auch eine intensivere Durchsicht; während bei einer rein mündlichen Präsentation Wissenslücken oft durch vage Formulierungen kaschiert werden können, ist dies schriftlich nicht so einfach.

Wir haben in den letzten Jahren vermehrt beobachtet, dass einige unserer Studierenden damit Probleme haben, klar gegliederte, sprachlich korrekte und stilistisch ansprechende fachliche Texte zu formulieren. Wird dies nicht bereits laufend während des Studiums geübt und praktiziert, so führt dies bei der Studien- oder Diplomarbeit zu Problemen, selbst wenn die inhaltliche Leistung der Studierenden (z. B. bei Modellierung und Implementierung) gut oder gar sehr gut ist. Schriftlich ausformulierte Lösungen bieten eine gute Möglichkeit, die Schlüsselkompetenz »Kommunikationsfähigkeit über fachliche Inhalte« zu entwickeln.

Aus Sicht der Lehrenden hat sich das System ebenfalls bewährt. Die Verwaltung und Bewertung der studentischen Lösungen wird deutlich erleichtert: Das Führen und die manuelle Erfassung von Votierlisten entfällt; die Anzahl der Einreichungen und Vorträge ist immer aktuell online verfügbar. Nur bei der erstmaligen Nutzung von eduComponents müssen sich die Studierenden registrieren. Bei weiteren Lehrveranstaltungen sind die persönlichen Daten dann bereits verfügbar. Die Daten über erzielte Leistungen (Anzahl Einreichungen und Vorträge) werden am Semesterende dann für die automatische Erstellung von Scheinen exportiert.

Ein weiterer Vorteil ist, dass an zentraler Stelle eine Sammlung von Tests, Aufgaben und Lösungen entsteht: Dies wird dadurch begünstigt, dass jede Aufgabe ein eigenständiges Objekt ist; das bisher übliche Aufgabenblatt wird durch die Kombination von Aufgaben in einem Containerobjekt ersetzt, das sich bei Bedarf aber natürlich auch ausdrucken lässt. Die eingereichten Lösungen sind der jeweiligen Aufgabe zugeordnet und erlauben auch noch später eine Analyse, um z. B. den relativen Schwierigkeitsgrad der Aufgabe zu bestimmen. Die Aufgaben lassen sich einzeln wiederverwenden und können beliebig zu neuen Übungsblättern oder Aufgabensammlungen kombininert werden. Aufgaben und Lösungen können als Objekte des CMS mit Metadaten versehen werden. Bei Aufgaben ermöglicht dies z. B. die Suche und Auswahl nach thematischen Kritierien. Bei Lösungen kann man diesen Mechanismus nutzen, um besonders interessante Lösungen leicht wiederauffindbar zu machen.

Der Wegfall der Votierlisten, die Möglichkeit, sich vor der Übung *gezielt* vorbereiten zu können, die Zeitersparnis durch den Wegfall der Tafelanschriebe und bei Programmieraufgaben die automatische Überprüfung bedeutet eine beträchtliche Arbeitserleichterung, so dass bei gleichem Aufwand mehr Aufgaben gestellt und besprochen werden.

5 Zusammenfassung und zukünftige Arbeiten

Wir haben in diesem Beitrag die Unterstützung von Übungen in der Informatiklehre durch die eduComponents vorgestellt.

Rechnerunterstützung für Multiple-Choice-Tests und die Einreichung und Verwaltung von Übungsaufgaben ist nicht neu (vgl. z. B. [Car96], [BHSV99]), die eduComponents unterscheiden sich jedoch von anderen Ansätzen durch die Integration in ein allgemeines CMS und die Anwendung des CMS-Workflowkonzeptes auf Tests und Übungsaufgaben. Die üblichen Abläufe im Übungsbetrieb können durch die Einsatz der eduComponents effizienter gestaltet werden.

Die Grundfunktionalität der eduComponents hat sich im praktischen Einsatz bewährt. Für die Weiterentwicklung sind daher vor allem Verbesserungen im Bereich der Benutzeroberfläche und zusätzliche Funktionen geplant. Diese Arbeiten werden z. Z. durch Mittel aus dem »Innovationsfonds zur Unterstützung von Maßnahmen zur Verbesserung von Studium und Lehre« der Otto-von-Guericke-Universität gefördert.

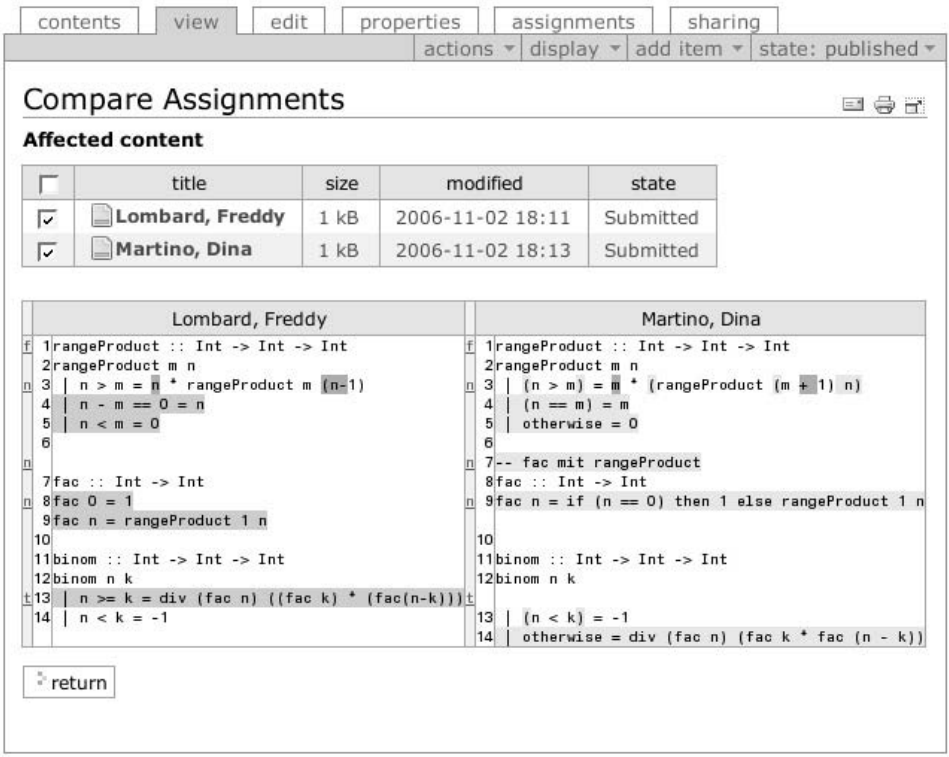

Abbildung 4: Durch die elektronische Einreichung ist ein Vergleich verschiedener Einreichungen leicht möglich (Bildschirmabzug aus einer Vorversion).

Weiterhin wurde mit der Integration von Werkzeugen für zusätzliche statistische Auswertungen begonnen. Ebenso untersuchen und testen wir Ansätze für den Umgang mit möglichen Plagiaten in ECAssignmentBox und ECAutoAssessmentBox. Dabei ist es nicht unser Ziel, vollautomatisch Einreichungen auszusondern, sondern den Lehrenden Hinweise auf mögliche Plagiate zu geben und ihnen Werkzeuge zur Verfügung zu stellen, mit denen verdächtige Einreichungen gezielt analysiert und beurteilt werden können (vgl. Abb. 4). Diese Werkzeuge könnten dann auch für andere Zwecke eingesetzt werden, z. B. zum Vergleich verschiedener Revisionen einer Einreichung.

Wir werden weiterhin die eduComponents in unseren Lehrveranstaltungen einsetzen und die Evaluation weiterführen. Aus unserer Sicht und – wie die Befragung gezeigt hat – auch aus der Sicht der Studierenden wäre ein Einsatz in anderen Veranstaltungen der Fakultät und darüber hinaus wünschenswert.

Die eduComponents sind auch für andere Fächer geeignet und werden dort auch bereits eingesetzt oder sollen es in naher Zukunft werden (z.B. im Fach Maschinenbau auch für Prüfungen bei großen Teilnehmerzahlen, wo elektronische Einreichungen gegenüber handschriftlichen Klausurbearbeitungen den Vorteil haben, dass die Korrektur und Bewertung nicht durch mangelnde Lesbarkeit zusätzlich erschwert wird).

Unsere Fakultät hat im Wintersemester 2006/2007 ihre Studiengänge auf die Abschlüsse Bachelor und Master umgestellt. Die mit eduComponents erzielbare Intensivierung der Übungen in der Informatiklehre ist gerade auch für die neuen Studiengänge von Relevanz, sollen doch diejenigen Studierenden, die als Bachelor bereits die Hochschule verlassen, zur Ausübung einer anspruchsvollen informatischen Tätigkeit dann auch wirklich befähigt sein. In diesen Studienformaten mit ihren – vor allem im Bachelor-Studium – strafferen Zeitvorgaben werden regelmäßige studienbegleitende schriftliche Aufgaben unterschiedlichen Umfangs eine größere Rolle als in den bisherigen Diplomstudiengängen spielen. Die eduComponents sind somit auch als ein Beitrag zu sehen, der die Umsetzung des Bologna-Prozesses in den universitären Lehralltag unterstützen kann.

Danksagung Wir bedanken uns bei den anonymen Gutachtern für ihre hilfreichen Hinweise und Anregungen.

Literatur

[APR06] Mario Amelung, Michael Piotrowski und Dietmar Rösner. EduComponents: Experiences in E-Assessment in Computer Science Education. In *ITiCSE '06: Proceedings of the 11th annual conference on Innovation and technology in computer science education*, Seiten 88–92. ACM Press, 2006.

[BEFH76] Benjamin S. Bloom, Max D. Engelhart, Edward J. Furst und Walker H. Hill. *Taxonomie von Lernzielen im kognitiven Bereich*. Beltz, Weinheim, 5. Auflage, 1976.

[BHSV99] J. Brunsmann, A. Homrighausen, H.-W. Six und J. Voss. Assignments in a Virtual University - The WebAssign-System. In *Proc. 19th World Conference on Open Learning and Distance Education, Vienna, Austria*, June 1999.

[BKW05] C. Beierle, M. Kulaš und M. Widera. A Pragmatic Approach to Pre-Testing Prolog Programs. In D. Seipel, M. Hanus, U. Geske und O. Breitenstein, Hrsg., *Applications of Declarative Programming and Knowledge Management. 15th International Conference on Applications of Declarative Programming and Knowledge Management, INAP 2004, and 18th Workshop on Logic Programming, WLP 2004, Potsdam, Germany, March 4-6, 2004, Revised Selected Papers*, Jgg. 3392 of *Lecture Notes in Computer Science*, Seiten 294–308. Springer, 2005.

[Car96] Richard Carver. Computer-Assisted Instruction for a First Course in Computer Science. In *Electronic Proceedings for FIE 1996 Conference*. IEEE, 1996.

[CH00] Koen Claessen und John Hughes. QuickCheck: a lightweight tool for random testing of Haskell programs. In *ICFP '00: Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, Seiten 268–279, New York, NY, USA, 2000. ACM Press.

[DW04] Charlie Daly und John Waldron. Assessing the Assessment of Programming Ability. In Dan Joyce, Deborah Knox, Wanda Dann und Thomas L. Naps, Hrsg., *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2004, Norfolk, Virginia, USA, March 3-7, 2004*, Seiten 210–213. ACM, 2004.

[Feu06] Thomas Feustel. Analyse von Texteingaben in einem CAA-Werkzeug zur elektronischen Einreichung und Auswertung von Aufgaben. Master's thesis, Fakultät für Informatik, Otto-von-Guericke-Universität, Magdeburg, 2006.

[FW65] George E. Forsythe und Niklaus Wirth. Automatic grading programs. *Commun. ACM*, 8(5):275–278, 1965.

[GVN02] M. Ghosh, B. Verma und A. Nguyen. An Automatic Assessment Marking and Plagiarism Detection. In *International Conference on Information Technology and Applications, IT in Engineering: AI, Signal/Image Processing, ICITA02*, Seiten 274–279, 2002.

[HST02] Colin Higgins, Pavlos Symeonidis und Athanasios Tsintsifas. The marking system for CourseMaster. In *ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education*, Seiten 46–50. ACM Press, 2002.

[KSZ02] Jens Krinke, Maximilian Störzer und Andreas Zeller. Web-basierte Programmierpraktika mit Praktomat. In *Proceedings des Workshop Neue Medien in der Informatik-Lehre*, Dortmund, Oktober 2002.

[LSKM04] Mikko Laakso, Tapio Salakoski, Ari Korhonen und Lauri Malmi. Automatic Assessment of Exercises for Algorithms and Data Structures – A Case Study with TRAKLA2. In *Proceedings of the 4th Finnish/Baltic Sea Conference on Computer Science Education, October 1-3, 2004, Koli, Finland*, Seiten 28–36, 2004.

[MKS02] Lauri Malmi, Ari Korhonen und Riku Saikkonen. Experiences in Automatic Assessment on Mass Courses and Issues for Designing Virtual Courses. In *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education*, Seiten 55–59. ACM, 2002.

[PR05] Michael Piotrowski und Dietmar Rösner. Integration von E-Assessment und Content-Management. In Djamshid Tavangarian Jörg M. Haake, Ulrike Lucke, Hrsg., *DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.*, Lecture Notes in Informatics (LNI) - Proceedings, Seiten 129–140, Bonn, 2005. GI-Verlag.

[RA05] Dietmar Rösner und Mario Amelung. A Web-Based Environment to Support Teaching of Programming Paradigms. In *Proceedings of the 4th IASTED International Conference on Web-based Education (WBE 2005), February 21–23, 2005, Grindelwald, Switzerland*, Seiten 655–660. IASTED, ACTA Press, 2005.

[RAP05] Dietmar Rösner, Mario Amelung und Michael Piotrowski. LlsChecker – ein CAA-System für die Lehre im Bereich Programmiersprachen. In Djamshid Tavangarian Jörg M. Haake, Ulrike Lucke, Hrsg., *DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.*, Lecture Notes in Informatics (LNI) - Proceedings, Seiten 307–318, Bonn, 2005. GI-Verlag.

[SMK01] Riku Saikkonen, Lauri Malmi und Ari Korhonen. Fully automatic assessment of programming exercises. In *ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education*, Seiten 133–136. ACM Press, 2001.

[WW05] Nicole Weicker und Karsten Weicker. Didaktische Anmerkungen zur Unterstützung der Programmierlehre durch E-Learning. In Jörg M. Haake, Ulrike Lucke und Djamshid Tavangarian, Hrsg., *DeLFI2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.*, Seiten 435–446, Bonn, 2005. GI-Verlag.