Webbasierte Dienste für das E-Assessment

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

Abstract: Zum Erwerb von Programmierfähigkeiten ist neben einem theoretischen Verständnis vor allem praktische Übung notwendig. Die automatische Überprüfung von Programmieraufgaben hilft, Studierenden mehr Übungsmöglichkeiten mit schnellerer Rückmeldung zur Verfügung zu stellen und Lehrende gleichzeitig zu entlasten, so dass sie sich auf inhaltliche und didaktische Fragen konzentrieren können. Wir stellen eine schlanke, dienstebasierte Architektur und ihre Implementierung für die automatische Überprüfung von studentischen Einreichungen vor und berichten über praktische Erfahrungen beim Einsatz dieser Software in unseren Lehrveranstaltungen.

1 Motivation

Für das Verständnis des Lernstoffs einer Vorlesung ist es unverzichtbar, das erworbene Wissen anzuwenden und sich durch das selbständige Lösen von Problemen mit den Theorien und Formalismen auseinanderzusetzen. So lässt sich beispielsweise Kompetenz im Programmieren nicht in einer Vorlesung allein erwerben, sondern nur durch eigenständiges Lösen von Programmieraufgaben und durch die Umsetzung von Algorithmen in lauffähige Programme.

Es ist daher ein an vielen Hochschulen verfolgtes Ziel, studentische Lösungen zu Programmieraufgaben automatisch zu überprüfen und den Studierenden somit mehr Übungsmöglichkeiten zu bieten und ihnen zu mehr Programmierpraxis zu verhelfen, ohne gleichzeitig den Betreuungsaufwand seitens der Lehrenden zu erhöhen. Die Ergänzung der Präsenzlehre durch Elemente des E-Learning, insbesondere des Computer-Aided Assessment (CAA), kann helfen, die universitäre Ausbildung zu intensivieren und effizienter zu gestalten. Hierbei stehen die Förderung der aktiven Beteiligung der Studierenden und die Reduzierung des administrativen Aufwands für die Lehrenden im Mittelpunkt. Um die aktive Beteiligung der Studierenden zu fördern, ist insbesondere die schnelle Verfügbarkeit von Feedback wichtig, um Motivation und Interesse zu erhalten. Durch die webbasierte Einreichung und die automatische Überprüfung von Übungsaufgaben wird auch das ort- und zeitunabhängige Selbststudium im Sinne einer *pervasive university* unterstützt.

2 Stand der Technik

Bereits seit den 1960er Jahren werden Systeme zur automatischen Überprüfung und Benotung von studentischen Lösungen zu Programmieraufgaben entwickelt [FW65]. Sowohl die Motivation für den Einsatz, als auch die Fragestellungen beim Einsatz dieser Systeme (z. B. Art der Bewertung, Kontrolle der Laufzeit, Sicherheit) haben sich seither nicht grundsätzlich geändert, wohl aber die Architektur und die Implementierung: Praktisch alle modernen Systeme verwenden eine Client-Server-Architektur; häufig erfolgt der Zugriff wahlweise per Web-Interface oder mit einem speziellen Client.

Es existiert heute eine Vielzahl von Systemen zur automatischen Überprüfung von Programmierlösungen. Häufig werden Einreichungs- und Verwaltungsfunktionen mit der eigentlichen Überprüfung integriert, die sich auf eine bestimmte Programmiersprache oder bestimmte Aufgabentypen beschränkt. Einige Systeme wie CourseMarker [HGST05], BOSS [JGB05] oder das AT(x)-Framework [BKW03] können jedoch prinzipiell für beliebige Sprachen und Aufgaben verwendet werden, da die eigentliche Überprüfung durch Module implementiert wird.

3 Anforderungen

Unsere E-Learning-Umgebung basiert auf den *eduComponents*, einer Sammlung von Modulen, die das Content-Management-System (CMS) *Plone*[1] um E-Learning-Funktionen erweitern. Die eduComponents umfassen Module für die Verwaltung von Lehrveranstaltungen (ECLecture), für Multiple-Choice-Tests (ECQuiz) und für die Einreichung von manuell bewerteten Übungsaufgaben (ECAssignmentBox), vgl. [APR06].

Zusätzlich zu manuell bewerteten Aufgaben wollten wir aus den in Abschnitt 1 genannten Gründen studentische Einreichungen (insbesondere zu Programmieraufgaben) automatisch überprüfen. An die Realisierung stellten wir drei Hauptforderungen: Zum einen sollte sich die automatische Überprüfung nahtlos in unsere E-Learning-Umgebung einfügen, so dass insbesondere Benutzerverwaltung und Datenhaltung nicht dupliziert werden. Zum anderen sollte das System modular erweiterbar sein, so dass es für Aufgaben in verschiedenen Programmiersprachen, aber auch für andere Aufgabentypen eingesetzt werden kann. Schließlich sollte für die Einreichung von Programmen den Benutzern möglichst die gleiche Oberfläche zur Verfügung stehen wie für die Einreichung anderer Aufgaben.

CourseMarker und BOSS sind Stand-alone-Systeme, die über eine eigene Benutzerverwaltung, Datenhaltung und Benutzerschnittstelle für Lehrende und Studierende verfügen. Das AT(x)-Framework ist dagegen zur Verwendung in Verbindung mit WebAssign [BHSV99] konzipiert, wobei WebAssign die Basisfunktionalität (Authentifizierung und Einreichung) bereitstellt, so dass sich AT(x)-Instanzen auf das eigentliche Testen der Einreichung beschränken. Das AT(x)-Framework entspricht somit am ehesten den o. g. Anforderungen.

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

Auch wenn eine Einbindung von AT(x) in unsere E-Learning-Umgebung prinzipiell möglich gewesen wäre, sprachen jedoch aus unserer Sicht zwei Punkte dagegen: Zum einen die Tatsache, dass jede AT(x)-Instanz eine eigene Datenbank mit Testdaten und Musterlösungen verwaltet – u. E. sollten diese Daten jedoch zusammen mit der Aufgabenstellung im CMS verwaltet werden – zum anderen, dass jede AT(x)-Instanz direkt mit dem Frontend kommuniziert. Wir entschieden uns daher für eine eigene Lösung, die im folgenden Abschnitt beschrieben wird.

4 ECAutoAssessmentBox und ECSpooler

Unsere Architektur (vgl. Abb. 1) besteht aus drei Schichten: Die erste Schicht ist ein Plone-Modul (ECAutoAssessmentBox), das die Einreichung entgegennimmt und an die zweite Schicht (ECSpooler) übergibt. ECSpooler ist ein webbasierter Dienst, der einerseits eine Warteschlange für Einreichungen und andererseits verschiedene *Backends* verwaltet. Die Backends – wiederum eigenständige webbasierte Dienste – bilden die dritte Schicht und führen die eigentliche Überprüfung der Einreichungen durch.

Alle Komponenten – ECAutoAssessmentBox, ECSpooler und jedes einzelne Backend – können auf verschiedene Rechner verteilt sein; die Kommunikation erfolgt mittels XML-RPC. Die Datenhaltung (Benutzer, Aufgaben, Testdaten, Einreichungen usw.) wird vollständig von Plone übernommen; die anderen Schichten erhalten lediglich die für die Verarbeitung eines Jobs nötigen Informationen.

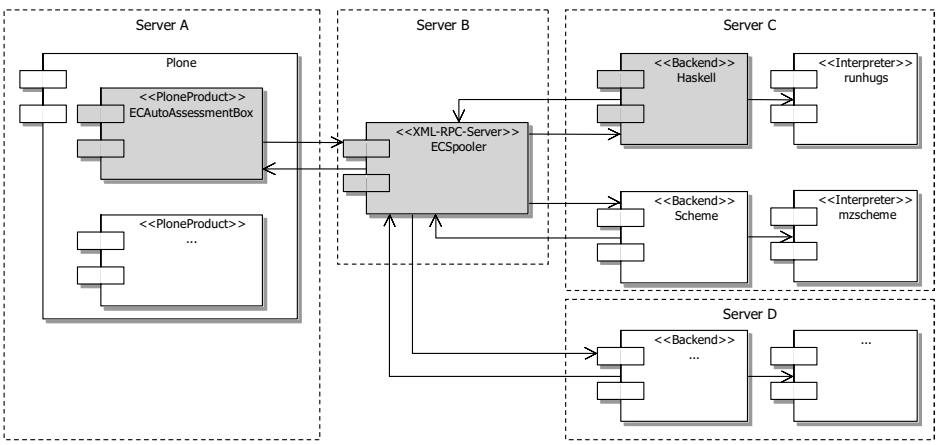

Abbildung 1: UML-Komponentendiagramm der Systemarchitektur

ECAutoAssessmentBox erbt von ECAssignmentBox (s. Abschnitt 3) und funktioniert daher prinzipiell gleich. Wenn Aufgaben in ECAutoAssessmentBox erstellt werden, werden sie vom Aufgabensteller mit einem bestimmten Backend assoziiert. Jedes Backend verfügt über ein sog. Schema, in dem beschrieben ist, welche Informationen es für das Testen

benötigt. ECAutoAssessmentBox generiert daraus dynamisch Eingabefelder für den Aufgabensteller. Zusätzlich können, wie bei ECAssignmentBox, verschiedene Parameter festgelegt werden, z. B. Einreichungsfrist und Anzahl der Versuche. Von Studierenden eingereichte Programme werden dann über den ECSpooler zum angegebenen Backend zur Überprüfung weitergeleitet.

Die konkrete Durchführung der Tests ist stark abhängig von der Programmiersprache, dem verwendeten Interpreter bzw. Compiler und der Testmethode (z. B. Spezifikation von Programmeigenschaften oder Vergleich mit Musterlösung). Die sich daraus ergebenden Besonderheiten werden in den Backends gekapselt. Zur Zeit sind Backends für Haskell, Scheme, Erlang, Prolog, Python, Java und reguläre Ausdrücke implementiert und im Einsatz. Darüber hinaus haben wir bereits mit Backends für die Überprüfung natürlichsprachlicher Texte experimentiert. Durch die Bereitstellung von Basisklassen für Backends ist es einfach, Backends für weitere Sprachen und unterschiedliche Testverfahren zu implementieren.

Bei der Ausführung von nicht vertrauenswürdigem Programmcode wie den studentischen Einreichungen müssen außerdem verschiedene Sicherheitsaspekte beachtet werden. Die Backends lassen sich entsprechend den Anforderungen an die Sicherheit und in Abhängigkeit von Programmiersprache und Plattform anpassen. Beispielsweise kann ein eingeschränkter Interpreter benutzt werden, die Ausführung innerhalb einer Sandbox-Umgebung erfolgen oder – wie in unserer Konfiguration – Systrace [Pro03] verwendet werden.

5 Einsatz und 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 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 die in diesem Beitrag beschriebene Kombination von ECAutoAssessmentBox und ECSpooler abgelöst. Im Wintersemester 2006/2007 wurde das System von über 200 Studierenden genutzt. Seit seiner Einführung bis Ende Juni 2007 wurden über 12 000 studentische Einreichungen automatisch überprüft. Das System hat sich dabei als sehr stabil erwiesen. Die Module werden auch an anderen Institutionen erfolgreich eingesetzt. [2]

Am Ende jedes Semesters befragen wir unsere Studierenden mit einem Fragebogen nach ihren Erfahrungen mit unserer E-Learning-Umgebung. Die Ergebnisse der Befragungen sind durchgängig sehr positiv, obwohl insbesondere für Programmieraufgaben die Anforderungen deutlich höher sind als früher. Die sofortige Rückmeldung bei der automatischen Überprüfung stellt nach Aussage der Studierenden eine wichtige Motivation dar, obwohl sich die Rückmeldung auf die Testergebnisse beschränkt und keinen tutoriellen Charakter hat. Die automatische Überprüfung entlastet die Lehrenden von Routineaufgaben und bietet ihnen einen genauen Überblick über den Leistungsstand der Studierenden, so dass sie Probleme besser erkennen und in den Übungen gezielt ansprechen können.

[2] `http://www.ovgu.de/PM_162_2006.html`

6 Zusammenfassung und Ausblick

Wir haben eine verteilte, webbasierte Architektur für die automatische Überprüfung von studentischen Einreichungen und ihre Implementierung in ECAutoAssessmentBox und ECSpooler vorgestellt. Seit mehreren Semestern hat sich das System beim praktischen Einsatz in unseren Lehrveranstaltungen bewährt. Alle Komponenten der eduComponents sind freie Software und unter der GPL veröffentlicht. [3]

Zur Zeit laufen Arbeiten, um die Möglichkeiten, die sich durch die elektronische Einreichung und die automatische Überprüfung ergeben, noch umfassender zu nutzen, etwa für studentische Peer-Reviews von Programmieraufgaben oder die Erkennung von möglichen Plagiaten. Neben der Entwicklung weiterer Backends (z. B. für die Überprüfung von XSLT-Stylesheets), untersuchen wir die Einsatzmöglichkeiten der automatischen Überprüfung von Aufgaben in anderen Fächern.

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, New York, 2006. ACM Press.

[BHSV99] Jörg Brunsmann, Andreas Homrighausen, Hans-Werner Six und Josef Voss. Assignments in a Virtual University – The WebAssign-System. In *Proc. 19th World Conference on Open Learning and Distance Education, Vienna, Austria*, June 1999.

[BKW03] Christoph Beierle, Marija Kulaš und Manfred Widera. Automatic Analysis of Programming Assignments. In Arndt Bode, Jörg Desel, Sabine Ratmayer und Martin Wessner, Hrsg., *DeLFI 2003. Proceedings der 1. e-Learning Fachtagung Informatik*, Jgg. P-37 of *Lecture Notes in Informatics*, Seiten 144–153, Bonn, 2003. GI.

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

[HGST05] Colin A. Higgins, Geoffrey Gray, Pavlos Symeonidis und Athanasios Tsintsifas. Automated Assessment and Experiences of Teaching Programming. *J. Educ. Resour. Comput.*, 5(3):5, 2005.

[JGB05] Mike Joy, Nathan Griffiths und Russell Boyatt. The BOSS Online Submission and Assessment System. *J. Educ. Resour. Comput.*, 5(3):2, 2005.

[Pro03] Niels Provos. Improving Host Security with System Call Policies. In *Proceeding of the 12th USENIX Security Symposium, Washington, DC*, August 2003.

[RAP05] Dietmar Rösner, Mario Amelung und Michael Piotrowski. LlsChecker – ein CAA-System für die Lehre im Bereich Programmiersprachen. In Jörg M. Haake, Ulrike Lucke und Djamshid Tavangarian, Hrsg., *DeLFI 2005: 3. Deutsche e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.*, Jgg. P-66 of *Lecture Notes in Informatics*, Seiten 307–318, Bonn, 2005. GI.

[3] Die Komponenten sind unter `http://wdok.cs.uni-magdeburg.de/software/` erhältlich. Weiterhin steht ein Demonstrationssystem unter `http://wdok.cs.uni-magdeburg.de/demo/` zur Verfügung.