Flexibles E-Assessment mit OLAT und ECSpooler

Jens Hoernecke, Mario Amelung, Katrin Krieger, Dietmar Rösner
Otto-von-Guericke-Universität Magdeburg
Institut für Wissens- und Sprachverarbeitung
Postfach 4120, 39016 Magdeburg
jenshoernecke@gmx.de
{amelung,kkrieger,roesner}@ovgu.de

Abstract: Der Web-Service *ECSpooler* bietet eine flexible Möglichkeit, E-Assessment-Funktionalitäten für Programmieraufgaben in bestehende Learning Management Systeme (LMS) zu integrieren. Bisher existierten als Frontends für diesen Service das Plone-Produkt *ECAutoAssessmentBox* sowie ein stand-alone Java-Client. Das LMS OLAT bietet neben üblichen Testmöglichkeiten, z. B. in Form von Multiple-Choice-Tests oder Lückentexten, bisher keine Möglichkeit, auch Lösungen zu typischen Aufgabenstellungen in der Informatikausbildung automatisch zu überprüfen. In diesem Beitrag wird gezeigt, wie OLAT um diese E-Assessment-Möglichkeit erweitert werden kann. OLAT agiert als weiteres Frontend für *ECSpooler* und ist somit in der Lage, die flexiblen Möglichkeiten des Web-Service zur automatischen Überprüfung von Programmieraufgaben zu nutzen.

1 Motivation

Bei der Nutzung von Learning Management Systemen (LMS) gibt es verschiedene Möglichkeiten, den aktuellen Lernfortschritt der einzelnen Lernenden zu verfolgen. Viele LMS stellen dazu Werkzeuge wie beispielsweise Multiple-Choice-Tests, Lückentexte oder Zuordnungsaufgaben bereit. Darüber hinaus können in den meisten LMS Lösungen zu Aufgaben in Form von Freitext eingereicht werden. Diese Einreichungen sind dann allerdings manuell von den Lehrenden durchzusehen und zu bewerten.

In der Informatikausbildung – insbesondere beim Erlernen von Programmiersprachen sowie Algorithmen und Datenstrukturen – beinhalten derartige Freitexteinreichungen in der Regel ausführbaren Programmcode. Die Besonderheit im Vergleich zu natürlichsprachlichen Texten ist, dass Programmcode mit Hilfe entsprechender Werkzeugen, z. B. Unit-Test-Frameworks, automatisch überprüft werden kann.

Aktuell existieren mehrere webbasierte Systeme, die solche Werkzeuge nutzen und damit eine automatische Überprüfung und Bewertung von Programmieraufgaben unterstützen. Zu diesen zählen u. a. JACK (Java Applet Correctness Test) [GSB08], Praktomat [KSZ02], ASB (Automatische Software Bewertung) [MOSS07], DUESIE (Das UEbungsSystem der Informatik Einführung) [HQW08], WeBWorK [GSW04], Web-CAT [EPQ08] oder Criterion [BCL03]. Diese Systeme haben allerdings gemeinsam, dass sie jeweils eigenständig und in sich geschlossen sind, so dass sich ihre Testwerkzeuge nicht in bestehende LMS

integrieren lassen. Die Nutzung dieser Systeme neben einem bereits vorhandenen LMS resultiert daher in einem Mehraufwand bei der Administration und Redundanzen in der Datenhaltung.

Ein System, das sich nur auf die automatische Überprüfung und Bewertung von Lösungen zu Programmieraufgaben konzentriert, ist der Web-Service ECSpooler (s. [AKR09] und [AKR11]). Damit ist es möglich, Programmcode mithilfe verschiedener sogenannter *Backends* für mehrere Programmiersprachen automatisch überprüfen zu lassen, ohne dass hierfür eine Verwaltung der Nutzern, Lehrveranstaltungen, Aufgaben und Lösungen vorhanden sein muss. Ein LMS kann als Client in Form eines sogenannten *Frontends* für *ECSpooler* agieren und die angebotenen Testmöglichkeiten in das eigene Testportfolio integrieren.

Das LMS OLAT ist nach eigenen Angaben weltweit ca. 150 mal installiert und wird u. a. von mehreren Bildungseinrichtungen in Deutschland und der Schweiz eingesetzt.[1] Allerdings bietet OLAT keine Möglichkeiten zur automatischen Überprüfung von Programmieraufgaben. Durch die Nutzung von *ECSpooler* könnte die Funktionalität des automatischen Testens in OLAT angeboten werden und somit einen erheblichen Mehrwert für das LMS bieten.

In diesem Beitrag stellen wir zunächst kurz die Projekte *ECSpooler* sowie OLAT vor und zeigen, wie die Funktionalität des automatischen Testens von Programmieraufgaben in OLAT integriert werden kann. Im Anschluss daran gehen wir auf die konkrete Implementierung ein und fassen abschließend die erreichten Ergebnisse zusammen.

2 *ECSpooler*

Die *eduComponents* sind Softwarekomponenten, die das allgemeine Content Management System *Plone*[2] um E-Learning-Funktionalitäten erweitern (vgl. [APR06], [RPA07]) und damit den Einsatz als LMS ermöglichen.

Als eine Komponente der *eduComponents* bietet der *ECSpooler* in Kombination mit verschiedenen Backends das automatische Überprüfen von Programmieraufgaben und anderen formalen Notationen an. Als webbasierter Dienst stellt *ECSpooler* diese Funktionalität plattformunabhängig über eine einheitliche Schnittstelle zur Verfügung. *ECSpooler* übernimmt dabei die programmiersprachenunabhängigen Funktionen wie das Verwalten der Backends, die Entgegennahme der Einreichungen, das Anstoßen der Tests und die Weiterleitung der Ergebnisse.

Die tatsächliche Überprüfung der Einreichungen übernimmt jeweils ein spezifisches Backend, das die Testfunktionalität für eine bestimmte Programmiersprache und Testmethode kapselt. So lassen sich beispielsweise Java-Programme mittels des JUnit-Backends

[1] Das Bildundsportal Sachsen (http://bildungsportal.sachsen.de/) bietet beispielsweise OPAL – eine auf OLAT basierende Variante des LMS – allen Bildungseinrichtungen in Sachsens an. Die Universität Hamburg hat zum April 2010 die bis dahin eingesetze Lernplattform Blackboard komplett durch OLAT abgelöst (vgl. http://www.michel.uni-hamburg.de/olat.php).

[2] http://plone.org/

hinsichtlich entsprechender Unit-Tests überprüfen. Weitere Backends existieren für die Programmiersprachen Haskell, Scheme, Erlang, Prolog und Python sowie für XML.

Beliebige Frontends (insbesondere LMS, aber auch andere Systeme) können *ECSpooler* und die Backends nutzen, um Einreichungen bezüglich bestimmter Kriterien überprüfen zu lassen. Die Verwaltung von Nutzern, Aufgaben und Einreichungen (z. B. Zugriffsrechte, Einreichungsfristen oder Auswertungen) obliegt dabei weiterhin dem Frontend.

Mit *ECAutoAssessmentBox* – ebenfalls Teil der *eduComponents* – existiert ein Frontend für *ECSpooler*, das seit mehreren Jahren eingesetzt, evaluiert und weiterentwickelt wird. Lernende reichen ihre Lösung mittels der *ECAutoAssessmentBox* ein, die die Einreichung zusammen mit den hinterlegten Testdaten und dem Namen des ausgewählten Backends an den *ECSpooler* schickt. Der Spooler verwaltet sämtliche Einreichungen in einer Warteschlange und gibt die Einreichung an das entsprechende Backend weiter. Das Backend überprüft die Einreichung und liefert Statusmeldungen sowie eventuelle Fehlerbeschreibungen zurück. Das Ergebnis der Überprüfung wird dann durch *ECAutoAssessmentBox* als Rückmeldung zu einer Einreichung dem Lernenden, aber auch dem Lehrenden angezeigt.

Durch die Implementierung als Web-Service kann *ECSpooler* prinzipiell mit beliebigen Frontends verknüpft werden. Ebenso können eigene Backends implementiert und an den Spooler gebunden werden (s. Abbildung 1).

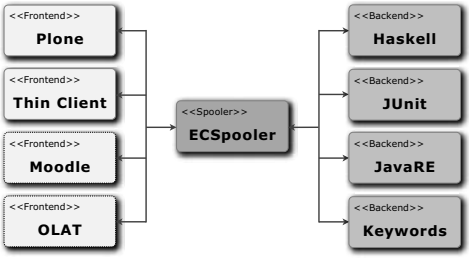

Abbildung 1: Beispiel für die Nutzung des *ECSpooler* ([AKR11])

Die einzelnen Komponenten – *ECSpooler*, die Backends und Frontends – können auf verschiedenen Servern verteilt werden. Dadurch ist es möglich, dass mehrere Frontends übergreifend den selben Spooler inklusive Backends nutzen. Sowohl Lehrende als auch Lernende nutzen weiterhin das ihnen bekannte Frontend, profitieren jedoch von den zusätzlichen Testfunktionen des Spoolers und der Backends.

3 Das LMS OLAT

OLAT steht für „Online Learning And Training“ [3] und ist ein in Java (J2EE) [4] implementiertes Open-Source Projekt, das 1999 von der Universität Zürich ins Leben gerufen wurde. Die

[3] http://www.olat.org/
[4] http://www.oracle.com/technetwork/java/javaee/overview/

ersten Versionen von OLAT waren zunächst in PHP umgesetzt. Im Jahr 2004 entschieden sich die Entwickler zu einer Neuimplementierung in Java, da PHP nicht performant und skalierbar genug erschien (vgl. [Gna01]).

OLAT bietet u. a. ein flexibles Kurssystem, die Verwaltung von Lernressourcen, Kommunikationswerkzeuge sowie Editorwerkzeuge für Kurse und Tests. Weiterhin werden die E-Learning-Standards SCORM [5], IMS QTI[6] und IMS Content Packaging[7] unterstützt.

Ein Kurs wird in OLAT mittels sogenannter *Kursbausteine* aufgebaut. Bei der Erstellung eines neuen Kurses ist bereits der Baustein *„Struktur“* enthalten. Dieser dient dazu, den Kurs zu gliedern – d. h., in jeder Struktur sind die jeweiligen anderen Bausteine enthalten, die zu diesem Abschnitt gehören.

Um Aufgaben stellen und diese bewerten zu können, gibt es die Bausteine „Bewertung“, „Aufgabe“ und „Test“. Der Kursbaustein „Bewertung“ eignet sich, um Leistungen zu bewerten, welche nicht elektronisch abgegeben werden. Eine „Aufgabe“ wird hingegen elektronisch eingereicht, aber manuell bewertet. Ein „Test“ dient zur Leistungsüberprüfung im Kurs, die Ergebnisse werden dabei personalisiert gespeichert.

OLAT stellt im Kursbaustein „Test“ verschiedene Fragetypen mit automatischer Überprüfung von Antworten zur Verfügung. Dazu zählen die Single- und Multiple-Choice-Fragen, der *Kprim* und der Lückentext. Beim Kprim stehen jeweils genau vier Antworten zur Verfügung und der Lernende muss entscheiden, welche dieser Antworten zutreffend sind. Im Lückentext wird das gesuchte Wort durch einen ausfüllbaren Block ersetzt, wobei die Anwortmöglichkeiten im Editor eingestellt werden (vgl. [ASI+11]).

Zur Erweiterung von OLAT um eigene, bisher noch nicht vorhandene Funktionalitäten stehen drei Wege zur Verfügung:

- Einfügen der Funktionalität in den bereits bestehenden Quellcode,
- Nutzung der von OLAT angebotenen *Extension Points* und
- Einbinden mittels Spring XML-Dateien[8].

Die Möglichkeiten 2 und 3 werden durch Spring-Beans realisiert, wobei die Konfiguration mittels *Dependency Injection* erfolgt[9]. Dadurch können Erweiterungen hinzugefügt werden, ohne in den Quellcode der Basisimplementierung von OLAT eingreifen zu müssen.

4 Anbindung von *ECSpooler* an OLAT

Aufgrund der positiven Erfahrungen und der Evaluation beim Einsatz der *eduComponents* und insbesondere der *ECAutoAssessmentBox* (vgl. u. a. [RAP06] und [AFR08]) wurden in

[5] http://scorm.com/ und http://www.adlnet.gov/technologies/scorm/
[6] http://www.imsglobal.org/question/qtiv1p2/
[7] http://www.imsglobal.org/content/packaging/
[8] Spring ist ein J2EE-Entwicklungsframework; s. http://www.springsource.org/
[9] vgl. http://static.springsource.org/spring/docs/2.5.x/reference/beans.html

Anlehnung an die *ECAutoAssessmentBox* folgende Erweiterungen implementiert, die die Funktionalität des automatischen Testens durch *ECSpooler* in OLAT integrieren:

- EC-Struktur als Strukturknoten (Spring-Bean),
- EC-Aufgabe für die Einreichungen (Spring-Bean),
- EC-Statistiken zur Auswertung und Analyse (Extension Point) und
- EC-Admin-Extension zur Administration (Extension Point).

Die Erweiterung EC-Aufgabe repräsentiert dabei das eigentliche Frontend für die automatische Überprüfung durch *ECSpooler* in OLAT. Ausgehend von den in [AKR11] definierten Anforderungen an Frontends übernimmt EC-Aufgabe folgende Funktionen:

- Erstellung von Aufgaben (Lehrende)
  - Aufgabentext, Beispieldaten und Antwortvorlage
  - Einreichungsfristen und Anzahl der Versuche
  - Musterlösung, Testdaten, etc.
  - Test der Einstellungen
- Einreichung von Lösungen (Lernende)
  - Entgegennahme der Einreichung (manuelle Eingabe oder Upload)
  - Senden der Lösung und der Testdaten an *ECSpooler*
  - Anzeige der Ergebnisse der automatischen Überprüfung
- Bewertung von Einreichungen (Lehrende)
  - Auflistung der Einreichungen aller betreuten Lernenden
  - Anzeige der Details einer Einreichung inkl. des automatischen Feedbacks
  - endgültige Bewertung durch den Lehrenden und Speicherung in OLAT

Erstellt ein Lehrender eine neue Aufgabe, werden zunächst die Namen und Kurzbeschreibungen aller verfügbaren Backends vom *ECSpooler* geladen. Wählt der Anwender ein bestimmtes Backend aus, werden die durch das Backend definierten Eingabefelder beim *ECSpooler* in Form von *DTOs*[10] abgefragt und daraus ein Eingabeformular generiert. In dieses Formular sind nun durch den Lehrenden alle für die automatische Überprüfung notwendigen Informationen einzutragen (je nach Backend z. B. Hilfsfunktionen, Musterlösung und Testdaten oder Unit-Tests) und zu speichern. Der Lehrende kann seine Einstellungen testen und die Aufgabe aktivieren.

[10] DTO – Data Transfer Object, „ein Objekt, dass Daten zwischen Prozessen überträgt, um die Anzahl der Methodenaufrufe zu verringern“ [Fow03].

Die Lernenden können innerhalb der Einreichungsfrist ihre Lösungen als Text eingeben oder als Datei hochladen. Die Einreichuung wird in OLAT gespeichert und in Form eines *TestDTOs* an den *ECSpooler* gesendet. Das TestDTO enthält neben der Einreichung nur die für die Tests notwendigen Daten.

Als Rückgabewert liefert *ECSpooler* eine sogenannte *JobID*, mit der die Ergebnisse der Überprüfung abgefragt werden können. Das Ergebnis besteht dabei aus einem *ResultDTO*, dass Informationen darüber beinhaltet, ob alles Tests die erwarteten Ergebnisse geliefert haben oder falls nicht, welche Fehler aufgetreten sind.

Die Einstellungen zur Kommunikation mit *ECSpooler* werden mit der Erweiterung EC-Admin-Extension verwaltet. Hier werden der URL des Servers, auf der Spooler läuft, sowie die Zugangsdaten eingetragen. Darüber hinaus kann aus einer Liste ausgewählt werden, welche der vorhandenen Backends den Kursautoren in OLAT zur Verfügung stehen sollen.

Durch die Erweiterung EC-Aufgabe ist es somit möglich, die speziell in der Informatikausbildung wichtigen praktischen Programmierübungen – inkl. der zeitnahen Rückmeldung an die Lernenden – nun auch in OLAT zu unterstützen.

5 Einsatz und Nutzung der Erweiterungen

Mit den in Abschnitt 4 vorgestellten, neu entwickelten Komponenten für OLAT wird das LMS um Funktionen zur automatischen Überprüfung von Programmieraufgaben und zur Analyse der erreichten Ergebnisse erweitert. Im folgenden Abschnitt wird erläutert, wie der praktische Einsatz dieser Komponenten konkret erfolgen kann.

5.1 Erstellung eines Aufgabenblattes

Die bereits in Abschnitt 4 erwähnte Erweiterung EC-Struktur erfüllt insbesondere folgende Aufgaben:

- Übersicht über alle studentischen Einreichungen zu einer Aufgabe sowie Zusammenfassung mehrerer Aufgaben (Lehrende),
- Definition von Bedingungen, die nötig sind, um einen Kurs zu bestehen (Lehrende),
- Übersicht über alle zu bearbeitenden Aufgaben und deren Status (Lernende) sowie
- Anzeige des automatischen Feedbacks (Lehrende und Lernende).

Naheliegend wäre die Verwendung des in OLAT bereits vorhandenen Kursbausteins „Struktur“ gewesen. Allerdings kann mit diesem Baustein nicht zwischen der Ansicht eines Lehrenden und eines Lernenden unterschieden werden. Die Ansicht der studentischen Leistungen ist in OLAT stattdessen im Unterpunkt „Bewertung“ zu finden, der jedoch nur für Autoren oder Betreuer verfügbar ist. Weiterhin ist die Definition von Bedingungen

(beispielsweise ab welcher Anzahl bearbeiteter Aufgaben bzw. Punktzahl ein Kurs als bestanden gilt) oder die Punktevergabe mit dem Kursbaustein „Struktur“ nur eingeschränkt bzw. umständlich möglich.

Daher wurde mit EC-Struktur eine spezialisierte Ableitung dieses Strukturknotens entwickelt. Neu sind die beiden Reiter „Anzeigeoptionen“ und „Bedingungen“. Der Reiter „Anzeigeoptionen“ dient dazu, die Informationen festzulegen, die dem Lernenden bzw. dem Lehrenden angezeigt werden sollen, sobald der Kursbaustein aktiv ist. Der Reiter „Bedingungen“ bietet die Möglichkeit, genaue Festlegungen zu treffen, wann ein Kurs bzw. ein Arbeitsblatt als bestanden gilt. Zudem ist es hier möglich, verschiedene Bedingungen zu kombinieren – z. B., dass ein Kurs als bestanden gilt, wenn von den ausgewählten Bausteinen mindestens 66 % mit bestanden bewertet wurden und zusätzlich ein Vortrag gehalten wurde (s. Abbildung 2).

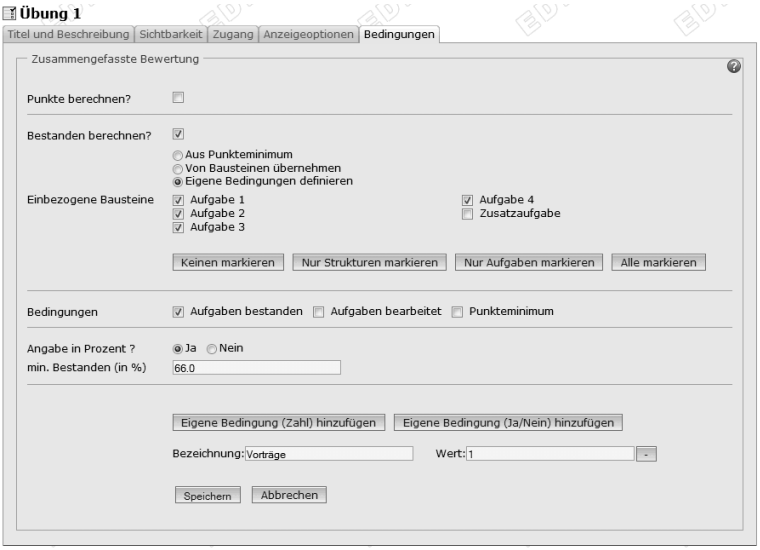

Abbildung 2: Beispiel für Bedingungen zum Bestehen eines Kurses/Arbeitsblattes

5.2 Erstellen einer Aufgabe

Die Erstellung einer Aufgabe erfolgt über den neuen Kursbaustein EC-Aufgabe. Wie in Kapitel 4 bereits aufgeführt, werden hier alle Einstellungen für eine Aufgabe getätigt, deren Einreichungen durch *ECSpooler*/Backends überprüft werden sollen. Dabei wird zwischen allgemeinen und speziellen Aufgabeneinstellungen unterschieden. Ebenfalls ist es möglich, diese Einstellungen auf dem Reiter „Aufgabenvorschau“ zu testen.

Unter „Allgemeine Einstellungen“ können – neben den aus OLAT bereits bekannten Einstellungen – weitere Festlegungen getroffen werden. So kann eine Begrenzung der Lösungs-

versuche eingestellt werden oder ob das Ergebnis der automatischen Überprüfung direkt als Bewertung der Aufgabe übernommen wird. Vorgaben für den Lernenden, wie z. B. Anwortvorlagen oder Programmskelette, können vom Lehrenden in das Feld „Programmvorgaben“ eingetragen werden.

Abbildung 3: Beispiel einer Aufgabenstellung mit dem Python-Backend

Die für die automatische Überprüfung notwendigen Daten werden auf dem Reiter „Aufgabeneinstellungen“ erfasst. Abbildung 3 zeigt das aus den Informationen des Python-Backends generierte Eingabeformular. Bei diesem Backend werden die eingereichten Lösungen auf Basis einer Musterlösung überprüft, d. h., beide Programme werden auf denselben Testdaten ausgeführt und die Ergebnisse miteinander verglichen. Der Lehrende muss daher die Musterlösung und die Testdaten festlegen sowie optional eventuell notwendige Hilfsfunktionen.

Die Darstellung variiert von Backend zu Backend – bietet dem Lehrenden aber je nach verfügbaren Backends ein hohes Maß an Flexibilität.

5.3 Einreichen von Lösungen

Für die Lernenden liegt das Hauptaugenmerk auf der Einreichung von Lösungen und dem automatischen Feedback, mit dessen Hilfe sich fehlerhafte Lösung gegebenenfalls verbessern lassen.

In Abbildung 4 ist das Beispiel einer Aufgabe zu sehen. Unterhalb von Name und Text der Aufgabe ist das Feld „Einreichung“ zu sehen, in das die Lösung eingetragen wird. Hier wird, falls vorhanden, auch die vom Lehrenden erstellte „Programmvorgabe“ angezeigt.

Abbildung 4: Beispiel einer Aufgabenstellung mit Antwortvorlage

Im Feld „Automatisches Feedback“ erscheint nach der Überprüfung einer eingereichten Lösung die automatische Rückmeldung des eingestellten Backends. Sollte eine maximale Versuchsanzahl festgelegt worden sein, wird unter diesem Feld die Anzahl der bereits verbrauchten bzw. noch zur Verfügung stehenden Versuche angezeigt. Darunter wird das aufbereitete Ergebnis der automatischen Überprüfung („Ergebnis des letzten Tests“) sowie die aktuelle manuelle Bewertung durch einen Lehrenden („Bewertung“) dargestellt.

5.4 Bewertung von Lösungen

Dem Lehrenden werden bei Aktivierung eines Kursbausteins alle von ihm betreuten Lernenden und deren Einreichungen sowie die bisherigen Bewertungen angezeigt. Diese Übersicht lässt sich beliebig sortieren. Durch Auswahl einer bestimmten Lerngruppe kann der Lehrende in die Detailansicht wechseln.

Der Lehrende kann sich einzelne Einreichungen näher ansehen, diese bewerten (bzw. die automatische Bewertung durch das Backend übernehmen oder ändern) und eventuell eingestellte Zusatzwerte eintragen (wie beispielsweise Vortragspunkte). Weiterhin bietet EC-Aufgabe die Möglichkeit, die Einreichung eines Lernenden herunterzuladen und zu speichern.

5.5 Auswertung der Ergebnisse

Die Auswertung der Ergebnisse zur Erfassung und Analyse des Lernfortschritts erfolgt entweder mittels der Zusammenfassung der Leistungen in den Aufgaben bzw. Aufgabenblättern oder aber mittels der Erweiterung EC-Statistiken (s. Abbildung 5).

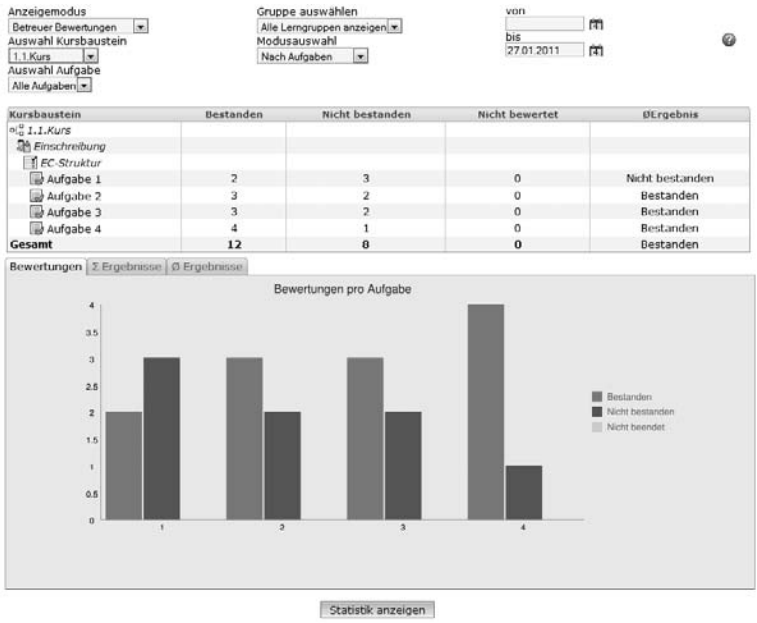

Abbildung 5: Beispielstatistik zur Bewertung von Lernenden

Die aggregierten Daten werden dabei anonym verarbeitet und ausgegeben. Es können drei verschiedene Arten von Statistiken in mehreren verschieden Kategorien dargestellt werden. Die Darstellung der Statistiken umfasst dabei die Einreichungen an sich, das Ergebnis der automatischen Bewertung und die Bewertung der Lösungen durch den Lehrenden. Als Kategorien und Einheiten stehen Aufgaben, Tage, Stunden, Wochentage und Versuche zur Auswahl.

Darüber hinaus kann die Bewertung aus der automatischen Überprüfung durch ein Backend mit der Bewertung durch den Lehrenden verglichen werden, um so Unterschiede in der

Bewertung zu finden (z. B. wenn eine Einreichung automatisch als *bestanden* markiert, diese aber nachträglich durch den Lehrenden als *abgelehnt* eingestuft wurde).

6 Zusammenfassung und Ausblick

Moderne Learning Management Systeme bieten zahlreiche Möglichkeiten zur Überprüfung des Lernfortschritts in Form verschiedenartiger Übungsaufgaben und Tests. In der Informatikausbildung haben solche Übungen oft die Gestalt von Programmieraufgaben, für die allerdings bisher nur wenige LMS entsprechende Werkzeuge zur Überprüfung anbieten. Mit *ECSpooler* exisitiert ein Web-Service, mit dessen Hilfe sich solche Übungsmöglichkeiten in bestehende LMS flexibel integrieren lassen.

In diesem Beitrag wurde gezeigt, wie das LMS OLAT dahingehend erweitert werden kann, dass es die Funktionalität von *ECSpooler* zur automatischen Überprüfung von Programmieraufgaben nutzt. Dafür wurden die speziellen Kursbausteine EC-Aufgabe und EC-Struktur entwickelt. Mit der Komponente EC-Statistiken können zahlreiche Möglichkeiten zur Analyse des Einreichverhaltens und des Lernfortschritts genutzt werden.

Die Entwicklung der neuen Bausteine für OLAT gestaltete sich schnell und effizient, da lediglich Client-Funktionalitäten implementiert werden mussten. *ECSpooler* bzw. die verschiedenen Backends kapseln die Funktionen zum automatischen Testen von Einreichungen zu Programmieraufgaben. Die in [AKR11] aufgestellte Behauptung „In conjunction with the spooler, backends offer a flexible and portable alternative to extend a learning management system or other elearning environments with functionality for automatic testing of programming assignments.“ lässt sich somit bestätigen.

Die Entwicklung von Komponenten für OLAT, die die Anbindung von Funktionalitäten zur automatischen Überprüfung von Programmieraufgaben ermöglicht, kann als Machbarkeitsstudie betrachtet werden. Die Verfeinerung der prototypischen Implementierung sollte ebenso wie die Evaluation ihres Einsatzes in der Lehre in zukünftigen Arbeiten behandelt werden. Hierfür werden aktuell Interessenten an verschiedenen Bildungseinrichtungen sowie in der OLAT-Community gesucht.

Weiterhin ist die Entwicklung von Clients bzw. Frontends zur Einbindung von *ECSpooler* für weitere LMS wie beispielsweise Moodle[11] wünschenswert und teilweise bereits in Planung.

Literatur

[AFR08] Mario Amelung, Peter Forbrig und Dietmar Rösner. Towards Generic and Flexible Web Services for E-Assessment. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, Seiten 219–224, New York, NY, USA, 2008. ACM.

[11] http://moodle.org

[AKR09] Mario Amelung, Katrin Krieger und Dietmar Rösner. Flexibles E-Assessment auf Basis einer Service-orientierten Architektur. In Andreas Schwill und Nicolas Apostolopoulos, Hrsg., *Lernen im Digitalen Zeitalter - DeLFI 2009: 7. E-Learning Fachtagung Informatik*, LNI, Seiten 247–258, Bonn, 2009. Gesellschaft für Informatik. ISBN 978-3-88579-247-5 / ISSN 1617-5468.

[AKR11] Mario Amelung, Katrin Krieger und Dietmar Rösner. E-Assessment as a Service. *IEEE Transactions on Learning Technologies*, 4:162–174, 2011.

[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.

[ASI+11] Sandra Arnold, Renata Sevcikova, Kristina Isacson, Joël Fisler, Sandra Hübner, Christian Meier und Sven Morgner. OLAT 7.0 Benutzerhandbuch. http://www.olat.org/website/en/download/help/OLAT_7_0_Manual_DE_online.pdf, 2011.

[BCL03] Jill Burstein, Martin Chodorow und Claudia Leacock. Criterion Online Essay Evaluation: An application for automated evaluation of student essays. In *Proceedings of the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence*, 2003.

[EPQ08] Stephen Edwards und Manuel A. Perez-Quinones. Web-CAT: Automatically Grading Programming Assignments. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, Seiten 328 – 338. ACM, New York, 2008.

[Fow03] Martin Fowler. *Patterns für Enterprise Application-Architekturen*. MITP-Verlag, 2003.

[Gna01] Florian Gnaegi. Architektur eines webbasierten Lernsystems. Diplomarbeit, Department of Computer Science, University of Zurich, Zurich, 2001.

[GSB08] Michael Goedicke, Michael Striewe und Moritz Balz. Computer Aided Assessments and Programming Exercises with JACK. Technical Report 28, ICB, University of Duisburg-Essen, 2008.

[GSW04] Olly Gotel, Christelle Scharff und Andy Wildenberg. Teaching software quality assurance by encouraging student contributions to an open source web-based system for the assessment of programming assignments. In *ITiCSE '08 Proceedings of the 13th annual conference on Innovation and technology in computer science education*. ACM, New York, 2004.

[HQW08] Andreas Hoffmann, Alexander Quast und Roland Wismüller. Online-Übungssystem für die Programmierausbildung zur Einführung in die Informatik. In *DeLFI 2008 - Die 6. E-Learning Fachtagung der Gesellschaft für Informatik e.V*, Seiten 173–184. Köllen Druck+Verlag GmbH, Bonn, 2008.

[KSZ02] Jens Krinke, Maximilian Störzer und Andreas Zeller. Web-basierte Programmierpraktika mit Praktomat. In *Softwaretechnik-Trends, Band 22, Heft 3*. GI-Verlag, Bonn, 2002.

[MOSS07] Thiemo Morth, Rainer Oechsle, Hermann Schloss und Markus Schwinn. Automatische Bewertung studentischer Software. In *Workshop "Rechnerunterstütztes Selbststudium in der Informatik"*. Universität Siegen, 2007.

[RAP06] Dietmar Rösner, Mario Amelung und Michael Piotrowski. E-Learning-Komponenten zur Intensivierung der Übungen in der Informatik-Lehre – ein Erfahrungsbericht. In Peter Forbrig, Günter Siegel und Markus Schneider, Hrsg., *2. GI-Fachtagung Hochschuldidaktik der Informatik*, Jgg. P-100 of *Lecture Notes in Informatics (LNI) – Proceedings*, Seiten 89–102, Bonn, 2006. GI-Verlag.

[RPA07] Dietmar Rösner, Michael Piotrowski und Mario Amelung. A Sustainable Learning Environment based on an Open Source Content Management Systems. In Wilhelm Bühler, Hrsg., *Proceedings of the German e-Science Conference (GES 2007)*. Max-Planck-Gesellschaft, 2007.