Prototypische Integration automatisierter Programmbewertung in das LMS Moodle

Sebastian Becker, Andreas Stöcker, Daniel Bräckelmann, Robert Garmann, Sören Grzanna, Felix Heine, Carsten Kleiner, Peter Werner, Oliver J. Bott
eLearning Servicestelle/Fakultät IV, Hochschule Hannover, Expo Plaza 12, 30539 Hannover
Kontakt: sebastian.becker@hs-hannover.de

1 Einleitung und Fragestellung

Die automatisierte Programmbewertung als ergänzendes Hilfsmittel in der Programmierausbildung ermöglicht eine zusätzliche Lernerfahrung für Studierende. Das unmittelbare Feedback dieser Systeme auf die eingereichten Aufgabenlösungen bietet den Studierenden Hilfestellung bei der erfolgreichen Bearbeitung der Aufgabe. An vielen Hochschulen existieren bereits Angebote an solchen Programmen zur automatisierten Programmbewertung (sog. Gradern), wobei sich deren Auswahl in den verschiedenen Hochschulen unterscheiden kann. An der Hochschule Hannover (HsH) wurden die Grader aSQLg für die Datenbankabfragesprache SQL und Graja für die Programmiersprache Java entwickelt [1]. Derzeitig verfügbare Grader arbeiten zumeist mit individuellen Benutzeroberflächen und müssen auch individuell konfiguriert werden. Um die Integration automatisierter Programmbewertung in die Hochschullehre zu vereinfachen, ist eine Integration der Grader in die jeweils bereits verfügbaren Lernmanagementsysteme (LMS) sinnvoll. Da standardisierte Schnittstellen in diesem Bereich bisher nicht existieren wurde an der HsH im Rahmen des Projektes eCult eine prototypische, Grader- und LMS-unabhängige Webservice-Schnittstelle mit dem Namen „Grappa“ implementiert und mit dem an der HsH eingesetzten LMS Moodle getestet. Mit der Entwicklung von Grappa und der testweisen Integration in Moodle sollen folgenden Fragen beantwortet werden:

1. Welche Schnittstellen muss eine Middleware bieten, um Grader in LMS integrieren zu können, und wie kann die Interaktion von LMS, Middleware und Grader erfolgen?
2. Wie kann der Prozess der Konfigurationseinstellung für den jeweils eingesetzten Grader und die jeweils eingesetzten Aufgaben für den Lehrenden im LMS am Beispiel Moodle möglichst aufwandsarm unterstützt werden?
3. Wie lassen sich die von Grappa gelieferten Bewertungsergebnisse der Grader innerhalb des LMS am Beispiel Moodle in dessen Bewertungsstrukturen integrieren?

2 Material und Methoden

2.1 aSQLg

aSQLg (**a**utomated **SQL** **g**rading) ist ein Grader, der zur Bewertung von Lösungen für SQL-Aufgaben dient. aSQLg untersucht dazu eine Lösung bezüglich der Kriterien Syntax, Kosten, Korrektheit und Stil. Passend zu dem Prüfergebnis werden Punkte für die Lösung vergeben. Bei Fehlern in der Lösung erzeugt der Grader detaillierte Hinweise, welche dem Studenten bzw. der Studentin die Korrektur vereinfachen sollen.

Derzeit unterstützt aSQLg SELECT-Anweisungen. Für die meisten Prüfungen verwendet aSQLg ein Datenbankmanagementsystem (DBMS). Die entsprechende Datenbank müssen die Dozenten/innen

[1]Das Projekt eCult ist gefördert vom BMBF im Rahmen des Qualitätspakt Lehre (Förderkennzeichen 01PL11066D).

vorab mit einem zu den Aufgaben passenden Schema einrichten und Beispieldaten einspielen. Zu jeder Aufgabe muss eine Musterlösung in Form eines SQL-Statements vorgegeben werden.

Zu jeder Teilprüfung vergibt aSQLg einen Punktwert zwischen 0 und 1. Um eine Gesamtpunktzahl zu vergeben, kann der Dozierende für jede Teilprüfung eine individuelle Gewichtung vorgeben. Diese kann pro Aufgabe variiert werden, so dass z.B. für Anfängergruppen auch für korrekte Syntax Punkte vergeben werden, bei fortgeschrittenen Studierenden dies aber vorausgesetzt wird und die Punkte nur noch für die weiteren Kriterien vergeben werden.

Die *Schnittstelle* zwischen aSQLg und Grappa arbeitet wie folgt. Als Eingabe schickt Grappa eine studentische Lösung sowie eine Musterlösung als Textdateien an aSQLg. Weiterhin werden Konfigurationsdetails als Textdatei übergeben. Dazu gehören z.B. Parameter der Datenbankverbindung, die aSQLg zur Prüfung verwenden soll, sowie die Regeln für die Stilprüfung. Weiterhin enthält die Konfiguration Einstellungen zur Punktevergabe durch aSQLg. Als Ergebnis der Bewertung einer als Textdatei an aSQLg übergebenen studentischen Aufgabenlösung liefert aSQLg ein Dokument, welches alle Details zur Prüfung und Punktevergabe enthält. Dieses Dokument kann von aSQLg in verschiedenen Formaten, wie z.B. XML, HTML oder Text geliefert werden.

2.2 Graja

Graja (**Gra**der for **Ja**va programs) vergibt Punkte für von Studierenden eingereichte Lösungen zu Programmieraufgaben. Graja verwendet JUnit-Tests zur dynamischen Analyse der Lösung und basiert partiell auf der „student“-Bibliothek von Web-CAT [4]. Graja erwartet zur automatischen Bewertung einer Aufgabe an der Eingabestelle eine sog. Request-Datei und liefert an der Ausgabeschnittstelle eine sog. Result-Datei. Die Inhalte dieser beiden Dateien sind in Tabelle 1 und 2 überblicksartig dargestellt.

Tabelle 1: Informationen an der Graja-Eingabeschnittstelle (Request-Datei)

Nr.	Information	Quelle	Zweck
1	Verweis auf JAR-Datei	Lehrkraft	Parametrierung von Graja für die Bewertung eines definierten Programmieraufgabenpakets
2	Verweis auf Policy-Datei		
3	Einstellungsparameter	Lehrkraft oder vom Front-end[1] selbst generiert	
4	Max. Punktzahl je Aufgabe	Lehrkraft	Gewichtung von Teilaufgaben
5	Verweis auf ZIP-Datei	Studierende	Einreichung

Tabelle 2: Informationen an der Graja-Ausgabeschnittstelle (Result-Datei)

Nr.	Information	Adressat/-in	Zweck
6	Punktzahl	Lehrkraft und Studierende	Bewertung
7	Kommentar (HTML / Text)	Studierende	Erläuterung und Tipps zur Verbesserung der Lösung
8	Kommentar (HTML / Text)	Lehrkraft	Bewertungsprotokoll

Wir beschreiben den Prozess der Erstellung der Informationen an der Graja-Eingabeschnittstelle und verweisen dabei jeweils auf die Zeilennummer in Tabelle 1. Die Lehrkraft erstellt im einfachsten Fall

[1] Frontend bezieht sich auf das Graja beliefernde System (z.B. ein LMS wie moodle oder eine Vermittlerplattform wie Grappa).

eine Musterlösung sowie eine „Grader“-Klasse mit JUnit-Test-Methoden, die die Ausgabe der studentischen Lösung und die der Musterlösung für verschiedene Eingaben unter Einsatz der Graja-Bibliothek miteinander vergleichen. Die Lehrkraft übersetzt Musterlösung und „Grader“-Klasse in eine *JAR-Datei* (1). Wenn die studentische Lösung und/oder die Grader-Klasse besondere Ausführungsrechte benötigen, erstellt die Lehrkraft zusätzlich eine *Policy-Datei* (2). Das Format ist an das Standard-Policy-Dateiformat[2] angelehnt.

Graja ermöglicht die Einreichung einer studentischen Lösung zu mehreren Programmieraufgaben in einem Durchgang. Die Einreichung erfolgt als *ZIP-Datei* (5), deren interner Aufbau sich in einen Wurzelordner und mehrere Aufgaben-Unterordner gliedert. Darunter werden die Quelltexte der einzelnen Aufgabenlösungen erwartet. Graja verarbeitet an der Eingabeschnittstelle eine Request-Datei im XML- oder JSON-Format. Die Request-Datei referenziert u.a. die oben mit Nummern gekennzeichneten Dateien (1), (2) und (5). Weiterhin enthalten sind die *maximal erreichbare Punktzahl* (4), sowie *Einstellungsparameter* (3) zur detaillierteren Steuerung des Gradingprozesses.

An der Ausgabeschnittstelle stellt Graja eine sog. Result-Datei (XML oder JSON) bereit, die für jede eingereichte Aufgabenlösung eine *Punktzahl* (6) und *Bewertungskommentare* (7,8) enthält. Bewertungskommentare sind in HTML oder einfachem vorformatiertem Text verfügbar. Graja kann separate Bewertungskommentare für die Lehrkraft bzw. für den/die Studenten/-in liefern und diese jeweils in verschiedenem Detailgrad (SEVERE bis FINEST, s. java.util.logging.Level[3]) anzeigen. Es kann etwa sinnvoll sein, dass die Lehrkraft für sich selbst ein detailliertes Bewertungsprotokoll mit Debug-Informationen anfordert, um eine auf den ersten Blick unerklärliche Graja-Bewertung zu verstehen. Studierende können zeitgleich einen Kommentar normalen Detailgrads mit Verbesserungstipps erhalten, weil diese an den Debug-Informationen nicht interessiert sind.

2.3 Moodle

Moodle ist ein Open-Source LMS, das international an Hochschulen, Universitäten und Schulen eingesetzt wird mit derzeit mehr als 70 Millionen Nutzern weltweit. Es arbeitet kursbasiert, skaliert für Anwendungen mit Tausenden Lernenden und eignet sich für reine Online-Lehre ebenso, wie für Blended-Learning. Moodle fußt auf dem kursbasierten Angebot von sog. *Aktivitäten* (wie z.B. Foren, Wikis, Datenbanken etc.) und *Materialien* (wie z.B. Dateien, Links, Texte, Datenverzeichnisse etc.), und fokussiert dabei auf kooperative Lerngruppen. Moodle ist als typische Web-Anwendung in PHP programmiert und benötigt ein Datenbankmanagementsystem wie z.B. MySQL. [2]

Spezielle Aktivitäten im Kontext des formativen und summativen E-Assessments mit Moodle sind sogenannte *Aufgaben* oder *Tests*. Erstere fragen ein Arbeitsergebnis beliebiger Art als Datei ab, das die Lehrkraft dann freitextlich kommentieren und mit Punkten bewerten kann. Letztere können als Sammlung einzelner Fragen der Typen Multiple- & Single-Choice, Zuordnungsfrage, Wahr/Falsch, Freitext, Lückentext etc. für veranstaltungsbegleitende Leistungsabfragen oder E-Prüfungen verwendet werden [3]. Hierbei arbeitet die Testperson die vorgesehenen Fragen ab. Nach Abschluss des Tests korrigiert der Prüfer etwaige Freitextaufgaben durch Kommentierung und Punktvergabe. Auch die automatisch bewerteten Fragen wie Multiple-Choice, Zuordnungsfragen oder Lückentextfragen können vom Prüfer nochmals durchgesehen werden, um fehlerhafte Bewertung z.B. bei Lückentextfragen (Schreibfehler) auszugleichen. Auch hierbei ist die Kommentierung der Lösungen bzw. Bewertungen möglich. Ob der Prüfer einen Test vor Bekanntgabe des Ergebnisses

[2] http://docs.oracle.com/javase/7/docs/technotes/guides/security/PolicyFiles.html
[3] http://docs.oracle.com/javase/7/docs/api/java/util/logging/Level.html

noch einmal manuell durchsehen und korrigieren möchte oder das automatisch ermittelte Ergebnis der Bewertung dem Studierenden direkt nach Abschluss des Tests zugeht, ist in Moodle konfigurierbar. Letzteres schließt Freitextfragen aus.

3 Ergebnisse

3.1 Grappa

Um Grader über eine Grader-unabhängige Schnittstelle in LMS verfügbar zu machen wurde eine Webservice-Middleware namens „Grappa“[4] entworfen und in der Programmiersprache Java implementiert. Grappa als Serverprozess und Ansprechpartner für ein oder mehrere LMS wird über Grader-spezifische Plug-Ins mit den eigentlichen Grader-Serverprozessen wie aSQLg oder Graja verknüpft. Diese Architektur ermöglicht zum einen die flexible Integration weiterer Grader über den Webservice in ein LMS sowie zum anderen die Ansteuerung von Grappa durch andere LMS. Das derzeitige Konzept sieht vor, dass pro Grader eine Grappa-Instanz gestartet wird und unabhängig von ggf. vorhandenen weiteren Grappa-Grader-Kombinationen arbeitet.

Die Kommunikation eines LMS mit Grappa erfolgt in zwei Stufen:

1. Einrichtung der Aufgabe über die sogenannte Setup-Schnittstelle: Zunächst legt die Lehrkraft über das LMS für die Verwendung eines Graders etwaig erforderliche Konfigurationsinformationen ab (im Folgenden *aufgabenübergreifende Graderkonfiguration* (*AÜGK*) bezeichnet). Im Falle von aSQLg enthält die AÜGK die benötigte Datenbankverbindung, im Falle von Graja werden diverse aufgabenübergreifende Einstellungsparameter (maximaler Ressourcenverbrauch, erwartete Ordnernamen im eingereichten Submission-Zip-Archiv, etc.) abgelegt. Dann wird pro Aufgabe ebenfalls eine Grader-spezifische Konfigurationsdatei an Grappa übergeben (im Folgenden *aufgabenspezifische Graderkonfiguration* (*ASGK*) bezeichnet). Im Falle von aSQLg ist dies die Musterlösung. Beispiele für von Graja erwartete ASGK sind eine JAR-Datei, welche kompilierten, JUnit-basierten Bewertungsprogrammcode enthält, und eine Policy-Datei, welche Sicherheitseinstellungen bei der Ausführung des Bewertungs- und des studentischen Programmcodes enthält. Zuletzt wird über die Setup-Schnittstelle unter Angabe einer ID ein *Problem* angelegt, das die Aufgabe unter Bezugnahme auf die erforderlichen AÜGK und ASGK abschließend spezifiziert. Nach Übergabe der Konfigurationsdateien und des Problems kann die Bewertung erfolgen. Hierbei wird vorausgesetzt, dass etwaig benötigte externe Systeme wie z.B. die für die Bewertung mit aSQLg benötigte Datenbank bereits eingerichtet sind.
2. Abgabe und Bewertung der Lösung über die sogenannte *RunGrader*-Schnittstelle: Der Studierende löst die Aufgabe durch Abgabe einer Datei (alternativ (z.B. für aSQLg) eine Texteingabe), die über das LMS und unter Bezugnahme auf o. g. Problem an Grappa übergeben und über das Grappa-Plug-In an den entsprechenden Grader weitergereicht wird. Grappa übernimmt die vom Grader zurückgelieferten Bewertungsergebnisse und leitet das Bewertungsergebnis inklusive der erreichten Punkte sowie Ergebniskommentare zur Erläuterung als mittels XML strukturiere Information an das LMS zurück. Bewertungsergebnisse werden in Grappa nicht gespeichert, dies wird dem LMS überlassen.

[4] Der Name „Grappa“ entstand als phonetisch kaum von „Grapper“ zu unterscheidendes Wort – mit bei den Projektmitgliedern aus naheliegenden Gründen angenehmeren Assoziationen. „Grapper“ ist ein Kofferwort für **Gra**der Wra**pper**.

Der Prozess der Bewertung und Ergebnisrückmeldung ist asynchron realisiert, um zu verhindern, dass das LMS während der Wartezeit auf das Bewertungsergebnis blockiert wird.

In der Kommunikation des LMS mit Grappa-Serverinstanzen identifiziert sich das LMS eindeutig über eine LMS-ID. Diese muss zuvor in der Grappa-Serverinstanz registriert sein. Die Kommunikation des LMS mit der Grappa-Serverinstanz erfolgt dann gesichert über ein Passwort. Auch die Konfigurationsdateien sowie die Aufgaben werden in der Grappa-Serverinstanz über eindeutige IDs identifizierbar abgelegt.

3.2 Moodle-Integration

Zur Integration der automatisierten Programmbewertung in das LMS Moodle werden Moodle-Erweiterungen in Form zweier neuer Aufgabentypen (im speziellen ein Aufgabentyp und ein spezieller Test-Fragentyp „Programmieraufgabe“) entwickelt. Im Folgenden wird die Implementierung und Nutzung des Aufgabentyps „Programmieraufgabe“ beschrieben, die Implementierung des Test-Fragentyps erfolgt weitestgehend analog.

Eine Aufgabe ist in Moodle eine spezielle Aktivität. Sie erlaubt dem Lehrenden, eine Aufgabe zu formulieren, die der Studierende durch Abgabe einer Datei bearbeitet. Der Lehrende kann diese dann manuell bewerten inkl. Formulierung eines textuellen Feedbacks oder eines Feedbacks in Form einer Datei. Der Studierende erhält das Feedback dann in Moodle und ggf. (nach entsprechender Konfiguration durch den Lehrenden) eine E-Mail zur Information über die erfolgte Bewertung. Wahlweise kann ein textuelles Feedback und/oder eine Datei als Feedback zurückgegeben werden.

Der neue Aktivitätstyp „Programmieraufgabe“ wurde aus der Aktivität Aufgabe abgeleitet und erweitert. Zusätzlich wurde zur Abbildung der für die Konfiguration der Grader benötigten Konfigurationsdateien das Arbeitsmaterial „Graderkonfiguration“ ergänzt. Vorbereitend für die Nutzung von Grappa bzw. den damit verknüpften Gradern muss Moodle durch Registrierung der verfügbaren Grappa-Instanzen vom Moodle-Administrator entsprechend konfiguriert werden. Die Einrichtung einer Programmieraufgabe durch die Lehrkraft erfolgt in Moodle dann wie folgt:

1. Einrichtung der Graderkonfiguration (*AÜGK*) als Arbeitsmaterial „Graderkonfiguration“. Pro Kurs ist wenigstens eine AÜGK erforderlich. Wesentliche Einträge der Graderkonfiguration sind ein Bezeichner, die für die eindeutige Kommunikation mit Grappa erforderliche ID der Konfiguration, die Bezugnahme auf eine im LMS registrierten Grappa-Instanz (z.B. aSQLg) sowie die Graderkonfigurationsdatei. Daneben können für Materialien in Moodle übliche Einstellungen wie Verfügbarkeitszeiträume und Sichtbarkeit vorgenommen werden. Für Graderkonfigurationen ist die Sichtbarkeit von vornherein auf die Rolle „Lehrender“ beschränkt, um eine versehentliche Veröffentlichung dieser Informationen zu verhindern.
2. Einrichtung der Aufgabenkonfiguration (*ASGK*) als Arbeitsmaterial „Graderkonfiguration“. Pro Aufgabe werden eine oder mehrere Graderkonfigurationen angelegt, welche die für die Bewertung der Aufgabe erforderlichen Grader-spezifischen Informationen enthalten (z.B. die Musterlösungsdatei für aSQLg oder die JAR-Datei für Graja: s.o.). Bei der Hinterlegung der ASGK werden die gleichen Angaben wie unter 1. beschrieben vorgenommen.
3. Einrichtung der Aktivität „Programmieraufgabe“ (*Problem*). Analog zur Aktivität „Aufgabe“ werden nun für die Studierenden benötigte Informationen zur Aufgabenstellung als HTML-Text hinterlegt sowie Parameter wie Abgabezeitraum, Abgabe in Gruppen etc. Zusätzliche spezifische Parameter der Aktivität Programmieraufgabe sind die Grader-Settings inklusive Bezugnahme auf die in den Schritten 1-2 angegebenen Grader-Konfigurationsdateien und

Angaben zur maximalen Punktzahl. Besteht die Aufgabe aus Teilaufgaben können pro Teilaufgabe erreichbare Punkte angegeben werden. Weiterhin kann spezifiziert werden, ob die Abgabe der Lösung als Datei oder als in Moodle eingebbare Textdatei erfolgen soll, ob mehrere Dateien abgegeben werden dürfen, ob es eine maximale Dateigröße gibt und ob Kommentare zur Abgabe erlaubt sind. Auch kann angegeben werden, ob die Bewertung der Abgabe direkt nach der automatisierten Bewertung durch den Grader an den Studierenden gesendet wird oder erst nach Einsichtnahme und ggf. ergänzende Korrektur der Lehrkraft.

Ist die Programmieraufgabe online geschaltet kann der Studierende die Aufgabe je nach in Schritt 3 gewählter Einstellung als Datei oder als Text abgeben und bewerten lassen. Je nach in 3. gewählter Einstellung erreicht den Studierenden das Bewertungsergebnis dann direkt nach Abschluss der Bewertung durch den Grader oder nach ergänzender Korrektur durch den Lehrenden. Die erreichten Punkte werden analog zu den bereits vorhandenen Assessment-Aktivitäten an die Bewertungsverwaltung von Moodle übergeben und stehen für eine Gesamtbewertung der Leistungen des Studierenden zur Verfügung.

4 Diskussion und Ausblick

Erste Tests belegen die generelle Funktionsfähigkeit der Grappa-Implementierung sowie der LMS-Integration in Moodle. Aktuell zu optimierende Systemaspekte betreffen die Rückmeldung bei fehlerhaften Graderkonfigurationen an das LMS respektive Moodle und deren Handhabung durch die Lehrkraft. Generell erfordert die Einrichtung der Graderkonfigurationen LMS-seitig noch sehr viel Grader-spezifisches Know-How mit negativer Auswirkung auf Usability und Fehlerrobustheit. In einem nächsten Entwurfsschritt wäre daher zu prüfen, wie Grader-spezifisch erforderliche Konfigurationseinstellungen (speziell die AÜGK) seitens des LMS über Grappa abgefragt und der Lehrkraft benutzerfreundlich zur Eingabe angeboten werden können. Derzeit (WiSe 2013/14) werden die Implementierung von Grappa und die neu geschaffene Funktionserweiterung von Moodle zunächst mit aSQLg im Echteinsatz evaluiert. Eine Evaluation mit dem Grader „Graja“ ist für einen späteren Zeitpunkt vorgesehen. Die Arbeit mit Gradern und speziell die Aufgabenerstellung erfordert auf Dozentenseite ein hohes Maß an Arbeitszeit und Systemkenntnis. Um Wiederverwendung von Aufgaben zu ermöglichen, wird innerhalb des eCult-Projekts an Konzepten für Aufgaben-Repositories gearbeitet, um Aufgaben für andere Dozenten nutzbar zu machen. Hierfür wurde im eCult-Verbundprojekt ein XML-basiertes Austauschformat entwickelt. Parallel zur o.g. Evaluation sollen Funktionen zum Aufgabenexport und -import auf Basis dieses Austauschformats in Moodle und Grappa integriert werden.

Literatur

[1] Stöcker A et al. Evaluation automatisierter Programmbewertung bei der Vermittlung der Sprachen Java und SQL mit den Gradern „aSQLg“ und „Graja“ aus studentischer Perspektive. Tagungsband der DeLFI-Tagung vom 2.-11.09.2013

[2] www.moodle.org, Community-Website (letzter Zugriff: 29.07.2013).

[3] Stöcker A, Chukhlova T, Tjettmers S, Becker S, Bott OJ. E-Prüfungen mit dem LMS Moodle: Ergebnisse einer Pilotstudie. In: Goltz U, Magnor M, Appelrath H-J, Matthies H, Balke W-T, Wolf L (Hrsg.) INFORMATICS 2012. GI-Edition Lecture Notes in Informatics; Proceedings 208; 2012 Sep 16-21: 1808-1821

[4] Edwards, S.: Using Test-Driven Development in the Classroom: Providing Students with Automatic, Concrete Feedback on Performance. In Proc. Int’l Conf. Education and Information Systems: Technologies and Applications (EISTA ’03), Aug. 2003.