Evaluation automatisierter Programmbewertung bei der Vermittlung der Sprachen Java und SQL mit den Gradern „aSQLg“ und „Graja“ aus studentischer Perspektive

Andreas Stöcker, Sebastian Becker, Robert Garmann,
Felix Heine, Carsten Kleiner, Oliver J. Bott

eLearning Servicestelle/Fakultät IV
Hochschule Hannover
Expo Plaza 12
30539 Hannover
andreas.stoecker@hs-hannover.de

Abstract: Als Bestandteil der Informatik-Lehre werden für die Programmierausbildung vermehrt Methoden der automatisierten Programmbewertung eingesetzt. Für die Programmiersprachen Java und SQL stehen hierfür an der Hochschule Hannover die Werkzeuge „Graja“ und „aSQLg“ zur Verfügung. In einer Evaluationsstudie wurde ermittelt, inwieweit diese beiden Werkzeuge Studierende und Dozenten unterstützen und wo die Grenzen der Bewertungshilfen dieser Systeme liegen. Befragt wurden 56 Studierende und die Tutoren eines 2. Semesters aus dem Studiengang Informatik im Bereich der Anwendung von Graja für einen Java-Kurs und 76 Studierende im 1. Semester des Studiengangs Medizinisches Informationsmanagement im Bereich der Anwendung von aSQLg für einen Datenbanken-Kurs mit SQL.

1 Einleitung

Automatisierte Programmbewertung in der Informatik-Lehre zielt darauf ab, die Bewertung studentischer Lösungen von Programmieraufgaben durch ein Programm („Grader“) mit entsprechender Bewertungs- und Feedbackfunktion zu unterstützen. Erwartete Vorteile liegen in der Beschleunigung und Objektivierung des Bewertungs- und Kommentierungsprozesses [Al05]. Auch kann erwartet werden, die Qualität der Programmierausbildung von Studierenden durch automatisierte Programmbewertung zu steigern [SOP11]. Für einige Programmiersprachen sind bereits Grader entwickelt worden [Al05], so z.B. JACK, ein webbasiertes System zur Überprüfung von Lösungen zu Java-Programmieraufgaben [SG11] oder XLX für SQL-Datenbankabfragen [VW04]. Ein weiteres Tool ist Praktomat, ein webbasiertes Bewertungssystem für mehrere Programmiersprachen [KSZ02]. Der Praktomat bietet zudem funktionale Tests, automatische Stilprüfung, gegenseitiges Kommentieren und die Erstellung von individuellen Aufgaben, um das Abschreiben fremder Lösungen zu erschweren.

An der Hochschule Hannover (HsH) begann 2010 die Eigenentwicklung des SQL-Graders aSQLg. Ursprünglich als Plugin für das von der Virginia Tech University, USA

stammende eAssessment Tool Web-CAT [Ed03] gedacht, wird es nun unabhängig davon in der Lehre eingesetzt. Ebenfalls in diesem Zeitraum begann man an der HsH mit der Nutzung von Web-CAT zur automatisierten Bewertung von Java-Quellcode. 2012 wurde die Arbeit mit diesen beiden Werkzeugen intensiviert und zur Praxisnutzung auf zusätzliche Kurse ausgeweitet. Zudem führte die Arbeit mit Web-CAT an der HsH zur Entwicklung des eigenständigen Graders Graja.

Die vorliegende, im Rahmen des BMBF-geförderten Projekts eCult entstandene Arbeit beschreibt neben den Systemen aSQLg und Graja die Ergebnisse einer Evaluationsstudie zur Frage, inwieweit die beiden Werkzeuge eine Eignung zur Unterstützung der Studenten und Dozenten bieten und wo die Grenzen der Bewertungshilfen dieser Systeme liegen. Zudem sollte die Akzeptanz beider Werkzeuge in Bezug auf deren Einsatz für summatives Assessment bzw. E-Prüfungen untersucht werden.

2 Material und Methoden

2.1 aSQLg

aSQLg („automated SQL grader“) prüft und bewertet studentische Abgaben zu SQL-spezifischen Aufgaben. Der Fokus liegt dabei auf Abfragen, deren Erlernen erfahrungsgemäß viel Übung erfordert. aSQLg speichert zu jeder Aufgabe eine Musterlösung für die Korrektheitsprüfung und um die Anleitungsfähigkeit (tutoring) für die Studierenden zu verbessern. Die studentische Abgabe wird in mehreren Schritten geprüft und mit der Musterlösung verglichen. Dazu nutzt aSQLg eine Datenbank, in der ein zur Aufgabe passendes Schema installiert sein muss. Zunächst prüft aSQLg die syntaktische Korrektheit, dann die Kosten, um Überlast auf der Datenbank zu verhindern. Wenn diese Prüfungen erfolgreich waren, vergleicht aSQLg schrittweise das Ergebnis der Abgabe mit dem Ergebnis der Musterlösung. Hierbei liefert aSQLg detailliertes Feedback, mit dem den Studierenden bei Bedarf eine zielgerichtete Fehlersuche ermöglicht wird. Folgendes Beispiel aus einer einführenden Datenbank-Veranstaltung erläutert den Ablauf. Die Musterlösung lautet:

```
SELECT first_name || ' ' || last_name name,
TO_NUMBER(TO_CHAR(hire_date, 'YYYY')) hired
FROM hr.employees ORDER BY hired, last_name;
```

Die erste studentische Einreichung ist folgendes Statement:

```
Select First_Name||' '||Last_Name, to_char(Hire_Date, YEAR)
from hr.employees order by Hire_Date, Last_Name;
```

Hier schlägt der Syntax-Check fehl, sodass aSQLg eine entsprechende Fehlermeldung zurückliefert, inklusive der originalen Fehlermeldung der Datenbank. Der nächste studentische Versuch korrigiert dies:

```
... to_char(Hire_Date, 'YYYY') ...
```

Diese Einreichung ist syntaktisch korrekt und besteht die Kostenprüfung. Als nächstes vergleicht aSQLg die Spaltenanzahl und die Spaltennamen des Ergebnisses mit der

Musterlösung. Die Anzahl ist richtig; aufgrund der abweichenden Spaltenbezeichnungen erzeugt der Grader eine Warnung. Der anschließende Datentyp-Vergleich für jede Spalte liefert einen Fehler, da die zweite Spalte einen String statt eines numerischen Wertes liefert. Eine entsprechende Meldung führt zu folgender studentischen Einreichung:

```
... to_number(to_char(Hire_Date, 'YYYY')) ...
```

Jetzt kann der Grader erfolgreiche Tests auf Spaltenanzahl und Datentypen zurückmelden; auch die Anzahl Ergebniszeilen und die Inhalte sind korrekt. Einzig der letzte Test auf die richtige Sortierung schlägt noch fehl, da die Abgabe nach dem ganzen Datum statt nur des Jahres sortiert. Die letzte studentische Abgabe korrigiert dann auch diesen Fehler, so dass aSQLg diese Abgabe am Ende mit der vollen Punktzahl bewertet.

Zusammenfassend zeichnet sich aSQLg durch ein detailliertes Feedback aus, verbunden mit einer dem Grad der Korrektheit angepassten Möglichkeit der Punktevergabe. Für die Zukunft arbeiten wir an einer verbesserten Stil-Prüfung der Abgabe sowie an einer Möglichkeit, auch weitere Statement-Typen wie DDL und DML zu bewerten.

2.2 Graja

Graja ("**Gra**ding **Ja**va programs") verifiziert einfache studentische Java-Programme funktional und hinsichtlich des Ressourcenverbrauchs. Graja entstand aufgrund von Vorerfahrungen mit dem System Web-CAT [Ed03], welches auf testgetriebene Entwicklung setzt. Web-CAT und Graja setzen JUnit[1]-Tests ein, um beobachtbares Verhalten studentischer Programme zu bewerten. Graja nutzt dabei die „student"-Bibliothek von Web-CAT, die den Dozenten bei der Erstellung von Testtreibern unterstützt. Gründe für die Entwicklung eines eigenen Graders auf Basis von Web-CAT liegen in den spezifischen Bedürfnissen der zu unterstützenden Lehrveranstaltungen sowie an partiellen Problemen mit der Stabilität von Web-CAT. Maßgeblichen Einfluss auf die Neuentwicklung von Graja hatte der Wunsch, den Grader in andere Lernmanagementsysteme einbinden zu können, was Web-CAT nicht unterstützt. Derzeitiges Einsatzszenario von Graja sind Anfänger-Lehrveranstaltungen zur Java-Programmierung.

Graja ist in Java geschrieben und bewertet mehrere Teilaufgaben in einem Zug. Der Dozent erstellt eine jar-Datei mit je einer Testtreiberklasse (Subklasse einer vorgegebenen Graja-Basisklasse) pro Teilaufgabe, wobei er ggf. mehrere Methoden im JUnit-Stil erstellt, die verschiedene Aspekte des studentischen Programms testen. Ist ein JUnit-Test erfolgreich, schreibt Graja die vom Dozenten per Java-Annotation programmierten Punkte gut. Schlägt ein Test fehl, kann der Dozent Hinweistexte programmieren. In Graja gibt es interne Hinweistexte für den Dozenten und für Tutoren und externe Hinweistexte für den Studenten. Graja wird hauptsächlich für den intelligenten Vergleich von Ausgaben studentischer Programme mit erwarteten Ausgaben genutzt. Graja besitzt Parameter zur Beschränkung der genutzten Ressourcen (Rechenzeit, persistenter Speicher und RAM) und zur Definition einer „security policy".

[1] http://junit.org/ (letzter Zugriff 27.03.2013)

Studentische Programme müssen sich an Namensvorgaben des Dozenten halten (Packages, Klassen, Methoden), damit der Dozenten-Testtreiber den studentischen Code via Java-Reflection orten kann. Graja erwartet vom Studenten eine ZIP-Datei mit vordefiniertem Aufbau (je Teilaufgabe ein Unterordner), entpackt die Datei, übersetzt Quelltexte, lädt den resultierenden Bytecode mit einer vom Dozenten konfigurierten Sicherheitsrichtlinie („ProtectionDomain“) und führt schließlich den Testtreiber aus. Als Rückgabe sieht Graja u. a. die erreichten Punkte und eventuelle Hinweistexte vor. Nicht geeignet ist Grajas JUnit-Ansatz für die Bewertung des Programmierstils (Programmaufbau, gewählte Bezeichner, verwendete Entwurfsmuster, etc.). Tutoren bewerten diese Aspekte und passen die von Graja vergebenen Punkte u. U. händisch an. Graja ist konzipiert für die Beobachtung des studentischen Programmverhaltens an den Schnittstellen Console, Datei-Ein-/Ausgabe sowie an internen Programmschnittstellen. Derzeit nicht Gegenstand ist die Nutzung von Mauseingabe und GUI-Ausgabe.

2.3 Evaluation

Um den Einsatz der Grader an der HsH untersuchen zu können, wurde eine anonyme Umfrage unter Studenten und den an der Auswertung der studentischen Lösungen beteiligten Personen (Tutoren) durchgeführt und ausgewertet. Sowohl die Evaluation zu aSQLg als auch zu Graja stützte sich dabei auf folgende Fragestellungen:

- Wie hoch ist die derzeitige Leistungsfähigkeit der Bewertung von Programmquellcode im Bereich Java durch „Graja“ und im Bereich SQL durch „aSQLg“ im Vergleich zu den Bewertungsmöglichkeiten eines menschlichen Tutors?
- Welche technischen Probleme lassen sich bei der Nutzung beider Grader durch größere Studentenzahlen im Bereich der Stabilität und Usability identifizieren?
- Wie beurteilen die Studierenden die Einsetzbarkeit beider Grader für E-Prüfungen?

Als Basis beider Umfragen diente ein Fragenkatalog, der an der Universität Duisburg-Essen entwickelt und zur Evaluation des dortigen Graders „JACK“ [SG11] eingesetzt wurde[2]. Die Befragung zum Einsatz der Grader „aSQLg“ und „Graja“ wurde im WS 2012/13 zwischen Dezember 2012 und Januar 2013 durchgeführt. Die Umfrage zu Graja fand in einem Java-Kurs mit 56 befragten Studenten und den zugehörigen Tutoren eines 1. Semester im Studiengang „Angewandte Informatik“ an der Fakultät IV „Wirtschaft und Informatik“ der HsH statt. Die Befragung zu aSQLg wurde unter 76 Studenten eines 1. Semesters im Studiengang „Medizinisches Informationsmanagement“ an der Fakultät III „Medien, Information und Design“ durchgeführt. Die Einführung in SQL als Abfragesprache für Datenbanken nahm innerhalb des Kurses ein Zeitfenster von ca. 4 Wochen ein, in dem die Studenten zur Lösung ihrer Aufgaben mit „aSQLg“ arbeiteten. Die erhobenen Befragungsdaten wurden im Februar/März 2013 ausgewertet.

[2] Die einzelnen Fragen können aus Platzgründen an dieser Stelle nicht behandelt werden. Beide Fragebögen und die detaillierten Evaluationsergebnisse sendet der Erstautor auf Anfrage zu.

3 Evaluationsergebnisse

Die Ergebnisse der Umfrage zu Graja zeigte in der Frage der Usability insgesamt ein positives Bild des Graders. Das von Graja produzierte Feedback musste 41% der Studierenden nie, 54% selten oder manchmal und 6% oft oder immer von anderen Personen erläutert werden. Die Intuitivität der Webschnittstelle wurde von 62% als „sehr gut“ bis „gut“ und von 35% als „befriedigend“ und von 4% als „mangelhaft“ bewertet. Die Stabilität von Graja beurteilten die Benutzer wie folgt: Interne Fehler, die dazu führten, das Graja keine Lösungen überprüfen konnte, konnten 75% der Befragten „gar nicht“ ausmachen, 17% der Befragten „selten“. Insgesamt wurde Graja im Bereich seiner Bewertungsmöglichkeiten als überwiegend gut bezeichnet: 73% der befragten Personen gaben an, dass Graja nie Lösungen akzeptiert hat, die falsch waren, jeweils 12% gaben an, dass dieses „selten“ bzw. „manchmal“ der Fall war. 58% der Studenten waren der Meinung, dass Graja bei der Lösung der Aufgaben hilfreich war. Graja animierte 79% der Probanden zudem dazu, die eigenen Fehler genauer zu analysieren. Das Graja ein guter Trainingspartner sei, finden 28% der befragten Personen. Die Hälfte aller befragten Studenten sehen in Graja eine Möglichkeit zum selbstständigen und unabhängigen Arbeiten, 37% sehen in Graja ein sinnvolles und modernes E-Learning-Werkzeug. Zusammenfassend empfinden 83% der befragten Studierenden Graja als hilfreich für die Programmierübungen, 78% würden Graja weiter im Studium einsetzen wollen, 5% nicht. Die Tutorenbefragung attestierte Graja insgesamt eine gute Trefferquote bei der automatisierten Bewertung.

Die Befragungsergebnisse zu aSQLg zeigten ein ambivalentes Gesamtbild. Usability und Stabilität von aSQLg wurden insgesamt nicht so positiv bewertet wie bei Graja. 59% der Befragten gaben an, dass aSQLg selten oder manchmal Fehler anzeigte, die keine waren, 18% waren der Meinung, dass dieses sogar oft oder immer der Fall wäre. Nur 23% konnten dies nicht beobachten. Dass aSQLg falsche Lösungen akzeptierte, konnten 48% nicht feststellen, 41% bemerkten dies selten oder manchmal. Interne Fehler, die auftraten und dazu führten, dass aSQLg keine Lösungen überprüfen konnte, konnten 34% nie beobachten, 51% manchmal oder selten, 15% beobachteten dies oft. Die Bewertungsmöglichkeiten von aSQLg wurden positiver eingeschätzt: aSQLg animierte 75% dazu, die eigenen Fehler genauer zu analysieren und 65% der befragten Studenten gaben an, dass dieser Grader ein sinnvolles Hilfsmittel war, die Aufgaben zu lösen. 63% würden aSQLg weiter im Studium einsetzen wollen, für 18% kommt dies nicht in Frage. Bei beiden Gradern waren die Antworten hinsichtlich der Fragen bezüglich des Einsatzes in Prüfungsumgebungen eindeutig: Dass der Grader gerechter sei als menschliche Überprüfungen, lehnten 71% bei Graja und 65% bei aSQLg ab, 76% konnten sich bei aSQLg und 73% bei Graja nicht vorstellen, eine Klausur mit diesen Gradern zu schreiben.

4 Fazit und Ausblick

Die Arbeit mit den Gradern befindet sich noch im Anfangsstadium. Die Evaluation liefert jedoch wichtige Hinweise zu deren Möglichkeiten und zur Akzeptanz unter den Studierenden. Die insgesamt positive Bewertung der Grader als Werkzeuge mit

gleichzeitig großer Ablehnung der graderunterstützten Notenvergabe zeigt auf, dass diese Tools als Lernhilfe verstanden werden, die bei relativ niedrigen Komplexitätsstufen ausreichen, um Programmieraufgaben lösen zu können. Die Entwicklung solcher Tools an der HsH führt derzeitig in die Richtung, die Qualität des Feedbacks bzw. der Lösungshilfen zu erhöhen, z.B. durch Fortschrittsanzeigen sowie konkreter und differenzierter gefasste Bestätigungs- und Hinweismeldungen. Lösungshinweise für den Studierenden könnten auch durch hinterlegte, generelle Lösungsschemata erzielt werden. Zudem wäre zu prüfen, ob sich in verschiedenen Sprachen wiederholende Grundmuster fehlerhaften Programmierens (z.B. fehlende Variablendeklarationen) in den Grader integrieren lassen.

Um automatisierte Programmbewertung in der Lehre zu etablieren ist es erforderlich, den Aufwand der Aufgabenstellung für die Grader zu reduzieren. Insbesondere in Bezug auf die automatisierte Bewertung von Java-Programmen mit Graja ist der zusätzliche Aufwand für die Gestaltung von Aufgaben vergleichsweise hoch. aSQLg-Aufgaben zu SQL können dagegen mit relativ geringem Zusatzaufwand erstellt werden. Effizienzverbesserungen bietet der Austausch von Programmieraufgaben mit anderen Lehrenden (vgl. [Al05]), weshalb im eCULT-Verbund an einem Austauschformat gearbeitet wird. Es soll ermöglichen, Aufgaben und Tests zwischen verschiedenen eAssessment-Tools auszutauschen und langfristig ein Aufgaben-Repository aufzubauen, das Lehrenden eine Arbeitserleichterung und den Studierenden durch die Aufgabenvielfalt eine qualitative Verbesserung der eigenen Lernerfahrung bietet. Um die Grader aSQLg und Graja sowie weitere Grader flexibel an das an der HsH eingesetzte Lernmanagementsystem Moodle bzw. weitere LMS wie z.B. LON-CAPA oder Stud.IP anbinden zu können, wird derzeit an der Entwicklung eines Grader-unabhängigen Webservices „Grappa“ gearbeitet. Hierdurch erwarten wir die Beschleunigung des Rollouts der Ansätze zur automatisierten Programmbewertung.

Literaturverzeichnis

[Al05] Ala-Mutka, K.: A Survey of Automated Assessment Approaches for Programming Assignments. Computer science education, volume 15, number 2, pages 93-102, 2005.

[Ed03] 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.

[KSZ02] Krinke J, Störzer M, Zeller A, Web-basierte Programmierpraktika mit Praktomat, Softwaretechnik-Trends, Vol. 22, (3), October 2002.

[PJR12] Priss, U.; Jensen, N.; Rod, O.: Software for E-Assessment of Programming Exercises. In Goltz et al. (Hrsg.) Informatik 2012, GI LNI, P-208, S. 1786-1791.

[SG11] Striewe M, Goedicke M: Studentische Interaktion mit automatischen Prüfungssystemen, in: Proceedings of "DeLFI 2011 - Die 9. E-Learning Fachtagung Informatik", Dresden, pages 209-219, 2011

[SOP11] Strickroth, S.; Olivier, H.; Pinkwart, N.: Das GATE-System: Qualitätssteigerung durch Selbsttests für Studenten bei der Onlineabgabe von Übungsaufgaben? In LNI Proc. DeLFI, 2011, S. 115 - 126.

[VW04] Vossen G, Westerkamp P: XLX and L2P - Platforms for Blended Learning. EMISA Forum 24(2): 18-20 (2004)