Daniel Schiffner (Hrsg.): Proceedings of DeLFI Workshops 2018
co-located with 16th e-Learning Conference of the German Computer Society (DeLFI 2018)
Frankfurt, Germany, September 10, 2018

Nicht nur Bestehen, sondern auch Verstehen: Ein Werkzeug für direktes, kontinuierliches Feedback beim Lernen von Programmieren

Automatisierte Testfälle für eine neue Organisation der Programmierausbildung

Sven Eric Panitz[1], Ralf Dörner[2]

Abstract: Experiences of a change in teaching programming are reported. We switched from weekly graded assignments to an automatic web application for the submission of solutions. We developed a web application, called »subato«, for this purpose. Programming challenges, with hidden unit tests, can be stored in the system. Subato will give continuous and direct feedback concerning code layout and correctness. Lecturers give individual comments to submissions. The system has been used for courses in the programming languages Java, C, Haskell and Kotlin.

Keywords: E-Learning, Programmierausbildung

1 Einleitung

Ähnlich wie z.B. das Schwimmen oder das Klavierspiel kann Programmieren nicht allein aus einem Buch erlernt oder über Vorlesungen vermittelt werden, sondern es bedarf der praktischen Tätigkeit. Empfehlungen der Gesellschaft für Informatik [GI16] sehen im Kompetenzfeld „Übertragen“ einer modifizierten AKT-Matrix [AK01] die Vermittlung folgender Kompetenzen in einem Bachelor-Studium der Informatik explizit vor: „Für eine spezielle Anwendungsaufgabe ein Software-System mittlerer Komplexität und angemessener Qualität in einer dafür geeigneten Programmiersprache mit angemessenen Programmiermethoden entwickeln (z.B. im Rahmen von Bachelor-Arbeit oder Software-Praktikum).“ Obwohl nicht direkt empfohlen sind üblicherweise in entsprechenden Modulen zur Programmierausbildung an Hochschulen Praktika integriert, in denen anhand von zumeist vorgegebenen Aufgaben schrittweise Erfahrungen in der Programmierung gemacht werden.

Dass Programmierausbildung gerade für Personen ohne Vorkenntnisse sich als schwierig darstellt, ist in der Literatur vielfach aufgezeigt worden [Mil02, Rob03, Jen02]. Entsprechende Vorschläge zur Verbesserung wurden diskutiert, wie z.B. Game-basierte Ansätze [Abb17] oder Ansätze, die ein individuelles Lerntempo erlauben [Off17]. Zu dieser Diskussion möchten wir beitragen, indem wir von unseren Erfahrungen mit der

[1] HS RheinMain, FB Design Informatik Medien, Unter den Eichen 5, 65195 Wiesbaden sveneric@panitz.name
[2] HS RheinMain, FB Design Informatik Medien, Unter den Eichen 5, 65195 Wiesbaden ralf.doerner@hs-rm.de

Umstellung unseres Ansatzes zu Praktika in der Programmierausbildung im ersten und zweiten Semester von Bachelor-Studiengängen im Bereich Informatik berichten. Diese Umstellung war nur durch den Einsatz eines von uns entwickelten eLearning-Systems möglich. Im nächsten Abschnitt beschreiben wir die Ausgangssituation und die Motivation für die Umstellung. Dann gehen wir auf unser neues Konzept ein und stellen dabei unsere Webplattform Subato vor. Schließlich berichten wir von unseren Erkenntnissen, bevor wir eine Zusammenfassung und einen Ausblick geben.

2 Ausgangssituation

Wir haben in unseren Informatikstudiengängen bisher einen klassischen Ansatz verfolgt, begleitend zu einer Vorlesung in Praktika wöchentliche Abgaben vorzusehen, die dann in den Praktikumsstunden von der jeweiligen Lehrkraft abgenommen und benotet wurden. Die Abnahme stellte dabei eine kleine wöchentliche Prüfung dar, da die Lehrkraft sicher gehen musste, dass die präsentierte, in Heimarbeit erstellte Lösung auch selbstständig erarbeitet und verstanden wurde. Das Praktikum zählt als Studienleistung, d.h. es kann beliebig oft wiederholt werden. Es herrscht Anwesenheitspflicht. Die Note geht zu 30% in die Modulnote ein. Entsprechend den GI-Empfehlungen [GI16] ist die Gruppengröße in den Praktika 15 Studierende. Als Zeit sind für das Praktikum wöchentlich im Semester 2 SWS Zeit (90 Minuten) vorgesehen. Für das aus Vorlesung und Praktikum bestehende Modul erhalten die Studierenden 10 ECTS Punkte.

Vorteil dieses Verfahrens ist, dass jeder Studierende persönliches Feedback durch eine Lehrkraft erhält. Daneben kann sich die Lehrkraft ein Bild vom aktuellen Leistungsstand der Studierenden machen. Allerdings gibt es auch eine ganze Reihe von Nachteilen. (1) Kursteilnehmer erhalten nur einmal die Woche Feedback zu ihren Lösungen und keine kontinuierliche Rückmeldung. (2) Lehrkräfte schaffen es kaum in einer Doppelstunde mit jedem Kursteilnehmer ausführlich über die individuelle Lösung zu schauen. Pro Person sind im Schnitt 6 Minuten verfügbar, wenn die Gesamtzeit nur auf die Abnahme verwendet wird. (3) Jede Woche findet ein kleines Prüfungsgespräch statt, um die benotete Eigenleistung sicher zu stellen. Die Studierenden befinden sich über das Semester in einer ständigen Prüfungssituation. Dies begünstigt, dass der Fokus im Lernziel nicht auf die Aneignung von Wissen, sondern auf das Bestehen von Prüfungen gelegt wird. (4) Lehrkräfte erklären in den Individualgesprächen oft immer wieder dasselbe. Dies führt zu ineffizientem Einsatz der teuren Lehrressourcen. (5) Aufgrund der geringen verfügbaren Zeit kann nicht sichergestellt werden, dass Aufgaben eigenständig erarbeitet wurden. Auch ob ein tiefgehendes Verständnis seitens der Studierenden vorliegt, kann nur oberflächlich beurteilt werden.

Die Fülle der in Abschnitt 2 beschriebenen Nachteile bewog uns die Lehre von praktischen Programmierkenntnissen komplett umzustellen.

3 Konzept und dessen technische Unterstützung

3.1 Elimination von Abnahmen

Die finanziellen Rahmenbedingungen an der Hochschule erlaubten nicht, alternativ mehr Lehrkräfte pro Praktikumsgruppe vorzusehen, um mehr Zeit pro Studierenden im bisherigen Ansatz investieren zu können. Stattdessen wurde ein neuer eLearning-Ansatz entwickelt, bei dem zu von Studierenden erstellten Programmen automatisch Feedback unter Verwendung einer Webplattform erzeugt wird. Auf diese Webplattform können die Kursteilnehmer jederzeit Lösungen und Teillösungen zu Programmieraufgaben hochladen und erhalten direkte Rückmeldung zur Kompilierbarkeit, zu einem Style-Check und zur korrekten Arbeitsweise ihres Programms basierend auf hinterlegte Testfälle. Dies ersetzt das Feedback durch die Abnahmen der Lehrkräfte. Um Abnahmen vollständig eliminieren zu können, musste auch eine Lösung für den Notenfindungsaspekt der Abnahme gefunden werden. Dazu wurde die Prüfungsordnung so geändert, dass die praktische Tätigkeit nicht mehr benotet wird, sondern nur ein »mit Erfolg bestanden« erhält.

Mit dem neuen Konzept wurden folgende positiven Änderungen erwartet: (1) Lehrkräfte sollten von dem Zeitdruck in den Abnahmen befreit werden und ihre Zeit effizienter einsetzen können, z.B. um typische Probleme in der ganzen Gruppe besprechen zu können statt 15 mal mit jedem Individuum. (2) Jeder Studierende erhält viel mehr Feedback zu den selbst erstellten Programmen. Dieses Feedback ist nicht auf einen kurzen Zeitraum in einer Praktikumsstunde beschränkt, sondern kann kontinuierlich, auch außerhalb der Hochschule genutzt werden. Dies soll auch die Problematik vermeiden, dass Studierende sich überschätzen [Lah05] (3) In der Rolle der Lehrkraft tritt der Aspekt der Bewertung nach hinten und der Aspekt der Betreuung und Unterstützung nach vorne. (4) Der Fokus der Studierenden liegt nicht darin, eine gute Note im Praktikum zu erzielen und die Kriterien für das Bestehen mit Blick auf einen Hochschulabschluss zu erfüllen, sondern selbst fundierte Programmierkenntnisse zu entwickeln auch mit Blick auf das weitere Studium und das spätere Berufsleben.

3.2 Funktionalität des eLearning-Systems

Während Freitext-Aufgaben aufgrund des hohen Schwierigkeitsgrades in der Verarbeitung von natürlicher Sprache im eLearning problematisch sind, liegt in der Programmierausbildung eine Sondersituation vor: Freitextantworten werden in einer formalen Sprache, der Programmiersprache, gegeben, deren Semantik zudem klar spezifiziert ist. Dies erschließt die Antwort einer automatisierten Verarbeitung. In der Tat erhalten Studierende bereits durch die Nutzung eines Compilers eine Rückmeldung über ihr Programm, z.B. hinsichtlich syntaktischer Korrektheit. Weitere Werkzeuge erlauben z.B. die Kontrolle der Einhaltung von Programmierrichtlinien (z.B. korrekte Einrückung von Programmzeilen) und geben ein Feedback z.B. zur Lesbarkeit und

Konformität des Programms. Compiler sind aber nicht in der Lage, die korrekte Funktionsweise des Programms zu überprüfen. Aus der Softwaretechnik sind aber Werkzeuge zur Durchführung von Tests bekannt, z.B. für Unit-Tests.

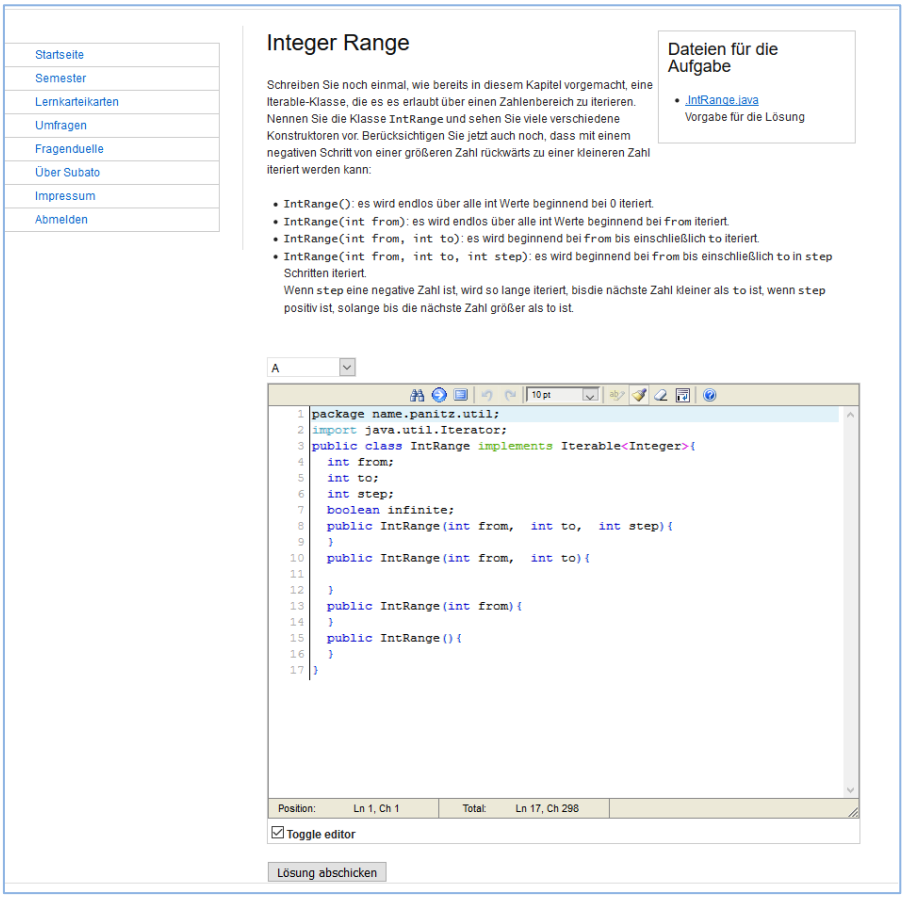

Abb. 1: Screenshot einer Aufgabe in Subato

Um verschiedene Aspekte von Feedback gleichzeitig geben zu können als auch um die Freiheit zu haben, didaktische Funktionen einzubauen, eine eigene E-Learning Plattform entwickelt: Subato. Unsere Installation ist auf subato.org erreichbar.

Subato ermöglicht es nach Semestern und Lehrveranstaltungen sortiert Übungsblätter anzulegen. Ein Übungsblatt ist eine Sammlung von Aufgaben. Aufgaben können für verschiedene Übungsblätter verwendet werden. Eine einzelne Aufgabe kann über eine

XML-Datei importiert werden. Hierfür liegt ein DTD Spezifikation vor. Eine Aufgabe enthält:

- einen Aufgabentext in HTML
- eine Vorgabe für die Lösungsdatei
- eine Musterlösung
- Testdateien mit Unit-Tests
- weiterer Quelltext, der zur Kompilierung der Lösungsdatei notwendig ist.
- eine Maximalanzahl von Versuchen, die pro Kursteilnehmer als Abgabe gespeichert wird.

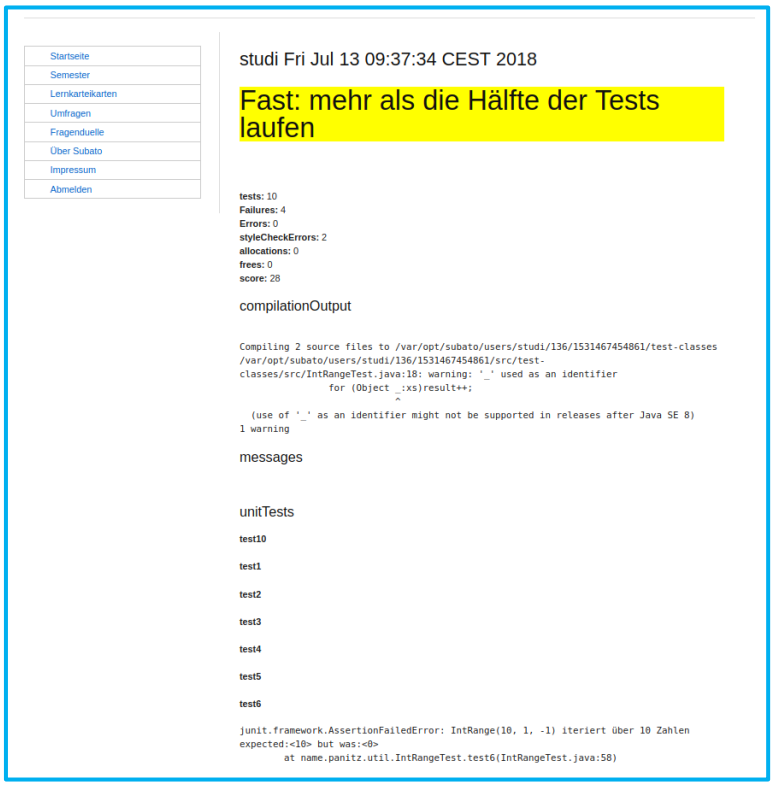

Abb. 2: Rückmeldung zu einer falschen Lösung der in Abb.1 gezeigten Aufgabe

Zum Lösen einer Aufgabe ist die Lösungsdatei im Browserfenster einzufügen und abzuschicken (vgl. Abb. 1). Die Abgabe durchläuft einen Style-Check, wird kompiliert und die Unit-Tests gestartet. Als Rückmeldung werden die Testausgaben und Style-Check Ausgaben gemeldet. Bei Aufgaben in der Programmiersprache C gibt es Feedback hinsichtlich möglicher Speicherlecks.

Dozenten einer Lehrveranstaltung können jederzeit die Abgaben mit ihren Ergebnissen einsehen und individuell kommentieren. Eine Übersicht aller Abgaben zu einem Übungsblatt steht der Lehrkraft zur Verfügung. Pro Teilnehmer wird dabei die beste Abgabe angezeigt. Hat ein Teilnehmer die maximale Anzahl von Versuchen überschritten, kann er weiterhin Lösungen austesten lassen, allerdings werden diese Versuche nicht mehr persistiert.

3.3 Implementierung von Subato

Subato ist mit dem Webframework Lift [Pol11] in der Programmiersprache Scala [OMM06] entwickelt. Hochgeladene Lösungen werden in Falle der JVM mit einem restriktiven Security Manager ausgeführt, bei Sprachen, die Executables generieren, auf einem abgeschotteten virtuellen Server. Die maximale Ausführungszeit für eine Lösung ist auf wenige Sekunden beschränkt, um nicht-terminierende Programme abzufangen. Der Quelltext zu Subato steht auf gitlab zur allgemeinen Verfügung. Dank gebührt Kim Stebel, der ein Vorgängersystem von Subato entwickelt hat.

4 Evaluation

Erstmals wurde Subato ab WS17/18 komplett angewendet und keine Noten, sondern nur noch ein Bestanden/Nichtbestanden auf die Studienleistung vergeben. Nach zwei Übungsblättern kam von Teilnehmern der Wunsch auf, ein plastischeres Feedback für korrekte Abgaben zu bekommen. Daraufhin wurde Subato um einen kleinen Gamification Aspekt erweitert. Komplett korrekte Abgaben werden mit einem kleinen Gimmick belohnt. Dieses ist derzeit ein Feuerwerk, ein Konfetti-Regen, ein Schiebepuzzle, ein Puzzle, ein 2D-Jump -Spiel oder eines der Strategiespiele Vier-Gewinnt oder Othello.

4.1 Verlauf der Praktikumsstunden

Die Praktikumsstunden verlaufen nicht nur für die Dozenten entspannter, sondern in jedem Semester hochschulweit durchgeführten Lehrevaluation wurde auch von studentischer Seite mehrfach die gute Atmosphäre explizit gelobt. Hier ist eine deutliche Verbesserung gegenüber Lehrevaluationen vor der Umstellung auf das neue Praktikumskonzept zu erkennen.

Statt bei jedem Teilnehmer individuell abzuprüfen, wie gut die abgegebene Lösung verstanden ist, kann schon im Vorfeld eingesehen werden, welche Fehler typischer Weise gemacht wurden und wo die Schwierigkeiten auftraten. Diese können im Plenum und anhand von abgegebenen Lösungen diskutiert werden. Es bleibt viel Zeit, um Teilnehmern, deren Verständnis noch etwas geringer ist, explizit zu helfen. Da die

Abgaben nicht benotet werden, zeigen die Teilnehmer ihre eigenen Versuche und anhand der gescheiterten Versuche kann zielgerichtet Hilfestellung gegeben werden.

Obwohl es keine eigentliche Note für die Programmieraufgaben mehr gibt, ist auch mit diesem Gamification Aspekt der Ehrgeiz beobachtbar gewachsen, komplett fehlerfreie Lösungen abzugeben. Viele Teilnehmer geben sich erst zufrieden, wenn sie das Feuerwerk sehen, obwohl zum Bestehen des Kurses nur eine sichtbare selbstständige Bearbeitung der Übungsblätter verlangt wird.

4.2 Verlauf unter der Woche

Entspannt sich die Arbeit für die Lehrkraft in den einzelnen Praktikumsstunden, so erhöht sich der Aufwand in der kontinuierlichen Betreuung unter der Woche. Ein regelmäßiger Blick in die aktuell abgegebenen Lösungen ermöglicht eine permanente Rückmeldung. Schon allein aus Neugier darüber, wie die Bearbeitung der Aufgabe läuft, schauen die Lehrkräfte gerne oft über die aktuell abgegebenen Lösungen. Hier lassen sich konkret im Arbeitsprozess gezielt und individuell Tipps geben.

4.3 Fazit

Wir sind mit der Umstellung der Praktika der Programmierausbildung im beschriebenen Umfang bisher mehr als zufrieden. Die Sorge, dass nach Wegfall der Benotung nur noch das Nötigste zum Bestehen gemacht wird, bestätigt sich nicht. Ganz im Gegenteil lässt sich eindeutig sagen, dass durch das System das Gros der Teilnehmer animiert wird, sich so lange immer wieder mit den Aufgaben zu beschäftigen, bis eine fehlerfreie Lösung erstellt wird. Das Ziel, dass mehr Zeit in die Programmierung investiert wird, wurde definitiv erreicht. Studenten schätzen auch die Plattform, um für die Prüfungsvorbereitung sich wiederholt an den Aufgaben zu versuchen. Ein Student versteigerte sich in der Lehrevaluation zu dem Kommentar: »Subato ist die Zukunft des Lernens.« Diese Behauptung halten wir zwar für weit übertrieben, aber sie zeigt zumindest, dass die Art, wie wir die Praktika jetzt durchführen und das System Subato von den Studierenden gut angenommen und gerne damit gearbeitet wird.

5 Zusammenfassung und Ausblick

Eine eLearning-Plattform, die automatisiertes Testen und Erstellen von Feedback in der Programmierausbildung realisiert, hat es erlaubt, bei der Durchführung von Praktika in der Programmierausbildung auf Abnahmen zu verzichten und eine bessere Unterstützung von Studierenden zu gewähren, da zum einen Studierende kontinuierlich Feedback erhalten und die Ressourcen der Lehrenden effizienter eingesetzt werden können. Wichtig war diese Umstellung mit organisatorischen Maßnahmen wie Änderungen der Prüfungsordnung geeignet zu flankieren.

Ein solches System lässt natürlich weitere Wünsche aufkommen. Es ist zu beobachten, dass besonders Anfänger die Abgabeplattform als Entwicklungsumgebung missbrauchen. Vielleicht sollte man dieser Tatsache Rechnung tragen, indem man für eine gängige Entwicklungsumgebung wie z.B. Eclipse ein Plug-in entwickelt, das direkt aus Subato-Aufgaben ein Projekt importiert und die Abgabe direkt aus der Entwicklungsumgebung erfolgen kann, so dass ein Kopieren und anschließend in den Browser Einfügen wegfallen. Statt wie bisher die Lehrkräfte in den Praktikumsstunden durch Tutoren zu unterstützen, wird angedacht, Tutoren zur online Betreuung unter der Woche einzustellen, um hier das kontinuierliche individuelle Feedback noch weiter zu intensivieren.

6 Literaturverzeichnis

[Abb17] Abbasi, S.; Kazi, H.; Khowaja, K. A systematic review of learning object oriented programming through serious games and programming approaches. In Engineering Technologies and Applied Sciences (ICETAS), 2017 4th IEEE International Conference on, S. 1-6, 2017

[AK01] Anderson, L.W.; Krathwohl, D.: A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives. Addison Wesley. 2001

[GI16] Gesellschaft für Informatik e.V. (GI): Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen, Bonn, 2016.

[Jen02] Jenkins, T.: On the difficulty of learning to program. Proc. of the 3rd Annual Conf. of the LTSN Centre for Information and Computer Sciences. Vol. 4. No. 2002

[Lah05] Lahtinen, E.; Ala-Mutka K.; Järvinen HM: A study of the difficulties of novice programmers. ACM SIGCSE Bulletin 37.3: 14-18, 2005

[Mil02] Milne, I.; Glenn R.: Difficulties in learning and teaching programming—views of students and tutors. Education and Information technologies 7.1: 55-66, 2002

[OMM06] Odersky, M.; Micheloud, S.; Mihaylov, N et al.: An overview of the Scala programming language (Second Edition), École Polytechnique Fédérale de Lausanne Technical Report LAMP-REPORT-2006-00, 2006

[Off17] Offutt J. et al.: A Novel Self-Paced Model for Teaching Programming. In: Proceedings of the Fourth, ACM Conference on Learning@ Scale. ACM, 2017. S. 177-180, 2017

[Pol11] Pollak, D.: Simply Lift. Website, 2011. –http://simply.lift.net. Datum des letzten Zugriffs: 13. Juli 2018

[Rob03] Robins, A. et al.: Learning and teaching programming: A review and discussion. Computer science education 13.2: 137-172, 2003