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

Feedback als Licht im Dunkeln – Zwischen Lust und Frust beim Lernen von Programmieren mit Online-Tools

Natalie Kiesler[1]

Abstract: Das Erlernen von Programmierfähigkeiten und –fertigkeiten ist schwierig und erfordert viel aktives Üben und Beharrlichkeit. Traditionell finden Lehrveranstaltungen allerdings in Form von Vorlesungen und Übungen mit wenig Zeit für individuelle Rückmeldung statt. Als Ergänzung dazu bieten aktuelle Online-Tools und deren Feedback für Lernende neue Chancen und Potentiale, besonders in selbstgesteuerten Szenarien ergänzend zum Curriculum. Mit Hilfe von Feedback kann die in der Programmierung unabdingbare Übung und Erfahrung unterstützt werden, indem Frustrationsquellen reduziert werden. Daher wird Feedback vor dem Hintergrund bekannter motivationspsychologischer Konzepte diskutiert. So wird deutlich, dass Feedback an die Kompetenz der Lernenden angepasst sein muss, um positiven Einfluss nehmen, und Unterstützung zur Überwindung kognitiver Anforderungen leisten zu können.

Keywords: Programmierung, Programmierausbildung, Herausforderungen, Feedback, Motivation

Abstract: Coding can be challenging for novice learners. It requires a lot of effort, exercise and persistence. Traditionally, classes take place as lectures accompanied by a tutorial. However, there is little time left for questions and individual feedback. Online tools may help with that issue. Since they offer feedback, they can constitute an addition to the curriculum and support learners while studying at their own pace. Feedback supports the essential exercise and help gain experience, which is vital for becoming a programmer. Moreover, it can help reduce sources of frustration. It is for these reasons, that this article discusses feedback within the framework of several well-known motivational and psychological theories. As a consequence of which feedback should be adapted to learners' competencies in order to generate positive effects and help overcome cognitive challenges while learning how to code.

1 Die aktuelle Situation in der Programmierausbildung

Die Programmierung als essentieller Bestandteil eines jeden Informatik-Studiums stellt eine wichtige Kernkompetenz im digitalen Zeitalter dar. Wer diese Kompetenz erreicht, dem stehen vielfältige Chancen auf dem Arbeitsmarkt und in der Gesellschaft insgesamt offen. Gleichzeitig werden Studienanfängerinnen in der Informatik mit hohen Leistungsanforderungen konfrontiert [He17]. Während es in manchen Fachdisziplinen zu Studienbeginn ausreicht, Literatur zu lesen, oder sich an Formeln und Konzepte zu

[1] Goethe-Universität Frankfurt, Akademie für Bildungsforschung und Lehrerbildung, Juridicum, Senckenberganlage 31-33, 60325 Frankfurt am Main, kiesler@em.uni-frankfurt.de

erinnern, wird in der Programmierung direkt zu Studienbeginn das Erreichen kognitiver Lernziele wie Anwenden, Analysieren und Erzeugen von Code angestrebt [AK01].

Das Image des stundenlang isoliert am Rechner arbeitenden Nerds kommt nicht von ungefähr [GM07]. Programmieren erfordert Problemverständnis und konzeptionelles Problemlösen, das Verständnis einzelner Programmstrukturen, die Übersetzung eines Algorithmus in die Programmstrukturen und Handlungsstrategien zum Debugging [Xi14]. Tatsächlich geht es nicht ohne intrisische Motivation [DR93] und eine gewisse Beharrlichkeit. Programmieren lernen erfordert viel Übung, Zeit und Hartnäckigkeit, besonders bei ausbleibenden Erfolgen zu Beginn des Lernprozesses. Studierende müssen zum einen selbst aktiv werden und eigene Programme schreiben, und zum anderen auch so lange, bis sie funktionieren, Ausgaben erzeugen und darüber hinaus effizient arbeiten. Es dauert nicht grundlos etwa zehn Jahre, bis aus einer Anfängerin eine Expertin wird [Wi96]. Fähigkeiten zur Selbststeuerung und –regulierung werden dabei im Studium vorausgesetzt, aber wenig thematisiert oder adressiert durch entsprechende Formate.

Aus den zahlreichen Herausforderungen der Programmierausbildung und den aktuell sehr hohen Zahlen zum Studienabbruch in der Informatik [He17] resultiert die Forderung nach neuen, unterstützenden Übungsformaten für Studierende – besonders in der Studieneingangsphase. Selbst in Übungen oder Praktika ist die Lehrperson nicht immer für alle Studierenden gleichzeitig greifbar. Große Gruppen verstärken das Problem [GM07]. Direktes Feedback, nicht nur vom Compiler, sondern von einer Tutorin oder Lehrperson, ist unerlässlich für das Lernen in der Programmierung. Es kann beim Problemlösen unterstützen, zum Verständnis beitragen und Zeitverschwendung mit trivialen Syntaxproblemen vermeiden – eine häufige Frustrationsquelle.

Die traditionellen Formate können dieser Anforderung nicht entsprechen, müssen sie allerdings auch nicht. Die zunehmende Digitalisierung der Hochschullehre birgt zahllose Potentiale in Bezug auf E-Learning und Selbstlernszenarien für Studierende. Nur durch die aktive Auseinandersetzung mit Problemstellungen können neue Informationen und Strategien der Programmierung nachhaltig im Langzeitgedächtnis gespeichert und damit immer verfügbar werden [Ma81]. Daher bedarf es Tools, die Lernende zum aktiven Schreiben von Programmen auffordern und motivieren. Feedback wird dabei eine große Bedeutung zugeschrieben.

2 Aktuelle Online-Tools zum Lernen von Programmieren

Aktuell ist eine Vielzahl an Angeboten frei im Web verfügbar, die verschiedenen Zielgruppen ganze Lerneinheiten und Aufgabenpools zu verschiedensten Themen der Programmierung anbieten. Beispiele solcher Plattformen sind CodingBat oder Scratch. Darüber hinaus gibt es unzählige Angebote an kompletten Selbstlernkursen über Codecademy, oder auf Massive Open Online Course (MOOC)-Plattformen wie dem

openHPI oder Udacity. In Bezug auf die Programmierausbildung in der Hochschullehre stellen CodingBat und CodeRunner, aber auch Blockly relevante Angebote dar.

2.1 CodingBat

So stellt Nick Parlante, Lehrender der Stanford University, die Seite CodingBat bereit, um Programmierfähigkeiten in Java und Python zu vermitteln. Das aus seiner Lehrerfahrung heraus entstandene Aufgaben-Repositorium ist für selbstständig zu erarbeitende Arbeitsaufträge denkbar, bietet sich aber auch für das Demonstrieren von Beispielen im Präsenzunterricht an. Vorteil ist, es bedarf keiner Vorbereitung oder Installationen, um Aufgaben zu bearbeiten. Lernende können direkt, ohne Anmeldung, im Browser Methoden schreiben und ausführen. Auf eine Eingabe hin werden Unit Tests durchgeführt und ein visuelles Feedback durch grüne, respektive rote Markierungen zu einer begrenzten Zahl an Testfällen erfolgt. Teilweise sind textuelle Hinweise und Lösungen zu den Aufgaben verfügbar. Die Hinweise erklären in der Regel die Aufgabenstellung oder geben einen Teil der Lösung vor. Das Hinweis-System ist allerdings nicht einheitlich gestaltet. Ziel ist es, Routine bei „einfachen“ Programmierproblemen zu erlangen, um sich auf das Lösen komplexerer Probleme konzentrieren zu können. Daher richtet sich das Angebot vorrangig an Studierende der ersten Semester. Lehrende können auf einer Unterseite Vorschläge für neue Aufgaben an den Autor senden und das Angebot damit erweitern [Pa17].

2.2 CodeRunner

Weiterhin existiert für Moodle das Open-Source Plug-In CodeRunner, mit dessen Hilfe Lehrende Aufgaben für Studierende generieren können, die Programmcode ausführen. CodeRunner funktioniert Programmiersprachen-unabhängig und wurde vor allem für Programmierkurse im Hochschulbereich entwickelt. So können Übungsaufgaben automatisch bewertet werden. Lernende können ihren Code zur Aufgabenlösung einfügen, und daraufhin die Ergebnisse von Testfällen einsehen, kombiniert mit farblich codierter Rückmeldung. Grün steht für korrekte Testfälle, rot für inkorrekte, ähnlich wie bei CodingBat. Der Vorteil für Lehrende ist, dass Ergebnisse zugänglich und Lernfortschritte transparenter werden. Die Vergabe von Punkten, wie auch die Bewertung insgesamt sind dadurch leicht möglich [Mo18]. CodeRunner hat damit vermehrt Assessment als Ziel, statt das reine Üben von Programmieraufgaben zu fördern. Hinweise zur Problemlösung oder Erklärungen zu Fehlerfällen sind daher aktuell nicht vorgesehen.

2.3 Blockly

Blockly als Open Source Bibliothek für Entwicklerinnen erlaubt die Gestaltung einer eigenen Block-basierten, visuellen Programmiersprache. Das im Jahr 2012 veröffentlichte Angebot bietet im Gegensatz zu Scratch eine sogenannte Block-Factory zur individuellen

Erstellung von Blöcken und einem Framework zur Generierung von Code in einer textbasierten Programmiersprache. Im Jahr 2018 sind Generatoren für JavaScript, Python, PHP, Lua und Dart verfügbar. Zudem können eigene Programmiersprachen entwickelt werden. Die Bibliothek ist in JavaScript geschrieben, sodass eine Integration in Webseiten leicht möglich ist. Mittlerweile sind auch iOS und Android Versionen verfügbar, mit dem Ziel, leistungsfähige mobile Anwendungen entwickeln zu können. Da Blockly client-seitig ohne zusätzlichem Bedarf an Servern auskommt, bestehen für Entwicklerinnen keine Abhängigkeiten von Dritten. Die Bibliothek ist keine App für Lernende, stellt aber eine Struktur und Grammatik dar, um individuelle Code-Editoren bereitzustellen, in Anlehnung an den gewünschten Kontext. Code-Ausschnitte werden als Blöcke repräsentiert und können einfach per Drag-and-Drop über den Bildschirm gezogen und an andere Blöcke geknüpft werden, sodass ein Programm entsteht. Welches Ergebnis die Code-Blöcke produzieren, wie Schlüsselworte gestaltet sind und wie das Programm läuft, welche Rückmeldung gegeben wird, hängt ganz vom angestrebten Nutzungskontext und den Entwicklerinnen ab [Go18] [PFM17] [Fr15]. Dadurch besitzt Blockly Potential für die Ausgestaltung von Feedback und einer eigenen, didaktischen Programmiersprache für die Studieneingangsphase.

3 Anforderungen an Feedback aus motivationspsychologischer Perspektive

Die Nutzung von Online-Übungsaufgaben bietet zahllose Vorteile, wie etwa zeitliche, örtliche Unabhängigkeit, unbegrenzte Möglichkeit der Wiederholung, Zeiteinteilung und ständige Verfügbarkeit. Darüber hinaus können bekannte, immer wiederkehrende Probleme und Fragen mit Hilfe von automatischem Feedback adressiert werden. Das breite Kontinuum zwischen allgemeiner und individualisierter System-Rückmeldung wirft allerdings die Frage auf, wie detailliert Feedback gestaltet sein muss, um sowohl kognitive Lernprozesse, als auch die Motivation zu fördern und nicht das Gegenteil.

Viele verschiedene auf die Motivation einwirkende innere und äußere Faktoren sind bekannt. Neben Bedürfnissen, Interessen und Zielen beeinflussen Gelegenheiten und Anreize das Verhalten eines Subjekts, und damit von Lernenden [Br13]. Daher ist es wichtig, individuelle Voraussetzungen der Lernenden, Vorwissen, sowie Unterschiede bezüglich Motivation und Haltung bei der Gestaltung von Feedback mit einzuziehen [Na06]. Individuelles Leistungsstreben, das akademische Selbstkonzept, und metakognitive Strategien zur Wissensdurchdringung können sich bei Lernenden innerhalb einer Lehrveranstaltung wesentlich unterscheiden. Feedback sollte sich daher an den verschiedenen kognitiven und nicht-kognitiven Dimensionen orientieren und die Existenz verschiedener Ausprägungen berücksichtigen. Vor allem auch deshalb, weil Lernende im Umgang mit Fehlermeldungen immer wieder irrationales Verhalten aufweisen. Sie kompilieren ihre Programme immer und immer wieder, ohne Veränderung am Code vorzunehmen, oder mit wahllose Veränderungen. Insofern liegt die Schlussfolgerung

nahe, dass Fehlermeldungen des Compilers nicht oder nicht vollständig wahrgenommen und berücksichtigt werden [Xi14].

Im Zusammenhang mit dem Studium allgemein, aber besonders mit selbstgesteuertem Lernen in Ergänzung zur Präsenzlehre ist theoretisch von intrinsischer Motivation auszugehen. Nichtsdestotrotz sind Motivation und Volition [He10] von Studierenden in der Praxis verschieden stark ausgeprägt. Vor diesem Hintergrund lassen sich aus der Selbstbestimmungstheorie nach Deci und Ryan [DR93] zusätzliche Implikationen für die Bearbeitung von Übungsaufgaben und die Auswirkungen von Feedback ableiten. So sind kontrollierende Lernbedingungen, externe Bewertung, Belohnung und Zeitdruck beispielsweise Faktoren, die intrinsische Lernmotivation untergraben können, aber nicht müssen. Fehlt das Gefühl der Selbstbestimmung, können kognitive Leistungen und Kreativität abnehmen und insofern das Lernen negativ beeinflussen. Dieser Korrumpierungseffekt zeigt sich am Stärksten, wenn Belohnungen angekündigt werden und Lernende diese für Anstrengung, statt für gute Leistung erhalten. Die Gestaltung und der soziale Kontext beeinflussen die Wirkung der äußeren Eingriffe [DR93] [He10].

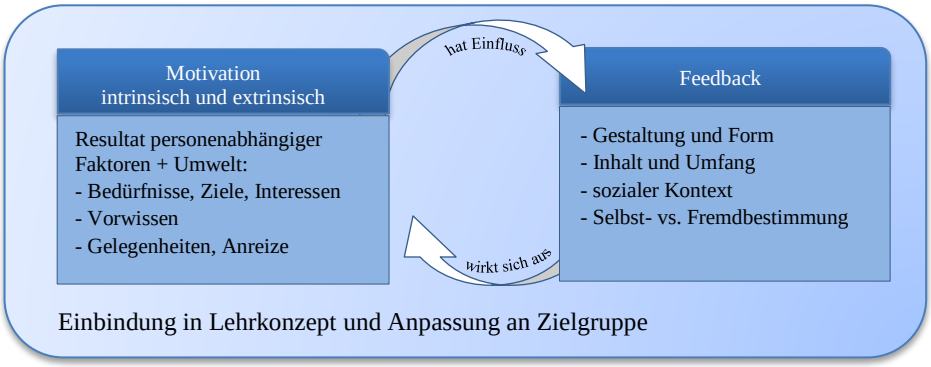

Abb. 1: Wechselwirkung von Motivation und Feedback

Daher ist es bei der Nutzung von Online-Tools mit Feedback-Optionen von immenser Wichtigkeit, deren Qualität und Passung in das eigene Lehrkonzept zu prüfen. Sollten Belohnungen oder Punkte verteilt werden, ist zwischen einem transparenten Punktesystem für Anstrengung und einem leistungsorientierten System mit überraschender Punktevergabe abzuwägen. Weiterhin ist zu berücksichtigen, dass Belohnungen und äußere Anreize, sprich Verursacher extrinsischer Motivation, die Internalisierung der extrinsischen Motivation bewirken können [DR93] [He10].

Um das aus motivationspsychologischer Perspektive bedeutsame Gefühl der Selbstbestimmung vermitteln zu können, sollte Feedback daher an Lernende angepasst sein, Mündigkeit der Lernenden fördern und nur bei Bedarf, auf eine Eingabe hin, vom System bereitgestellt werden. So kann das für intrinsische Motivation so wichtige Gefühl der Selbstbestimmung vermittelt werden [DR93]. Gezieltes, individualisiertes Feedback

kann insgesamt Lernwege weisen, Frustrationsquellen abschwächen und damit Kompetenzerleben unterstützen [Ke00].

Darüber hinaus stellt Feedback eine der drei Bedingungen für das Flow-Erleben beim Lernen dar. Ein Flow kann als extremes Hochgefühl begriffen werden und zeichnet sich durch hohe Zufriedenheit, verstärkte Handlungsausführung, hohes Selbstbewusstsein und ein insgesamt positives Lebensgefühl aus. Um dieses Gefühl beim Lernen zu erreichen, sind die Passung einer Aufgabe an die eigenen Fähigkeiten, sowie eine klare Zielsetzung als weitere Bedingungen nötig. Neue, ungewöhnliche Aufgaben, oder selbstbestimmte Tätigkeiten können unterstützen. Ergebnis des Flow-Zustands sind positive Performanz und zunehmende Resistenz gegen Ablenkungen oder Prokrastination. So kann eine positive Grundhaltung und die beim Programmieren so wichtige Erfolgszuversicht und strategische Arbeitsweise erreicht werden - ein Aspekt, der Anfängerinnen von Expertinnen unterscheidet [Br13] [Wi18].

4 Fazit und Ausblick

Traditionelle Lehrmethoden wie Vorlesungen erscheinen an bekannten Bedürfnissen und Schwierigkeiten in der Programmierung gemessen, als unpassend, insofern Lernende nicht aktiviert oder eingebunden werden. Die Rahmenbedingungen einer Lehrperson lassen es oft nicht zu, angemessene und notwendige Unterstützung via Feedback zu geben. Online-Angebote zum Selbstlernen mit Feedback-Optionen bieten eine interessante Alternative.

Selbstständiges Arbeiten kann für Studierende grundsätzlich eine Herausforderung darstellen. Daher ist bei der Nutzung von Online-Tools deren Einbindung in ein didaktisches Szenario durch die Lehrperson umso wichtiger. Nur so kann gewährleistet werden, dass mitunter gegebene Schwächen der extra-curricularen Anwendungen transparent werden. Die Rückmeldungen einiger Systeme sind lückenhaft und Musterlösungen nicht für alle Aufgaben vorhanden. Außerdem passt nicht jedes Angebot zur Zielgruppe und deren Vorkenntnissen. Motivationspsychologische Grundlagen des Lehrens und Lernens sind darüber hinaus auch in der Informatik zu berücksichtigen. Feedback muss zu dem Informationsstand und der Kompetenz der Lernenden passen, um nicht zu demotivieren.

Für das selbstständige Üben während der Programmierausbildung ist vor dem Hintergrund der komplexen Aufgabenstellungen besonders auf inhaltlicher Ebene ein feingranulares Angebot an Feedback notwendig. So kann zum Beispiel bei hohem Vorwissen der Lernenden die Kennzeichnung der Fehlerquelle im Code ausreichen. Für Lernende mit geringerem Vorwissen ist es möglicherweise nötig, zusätzlich die Art und Ursache eines Fehlers, sowie dahinterliegende Konzepte, Begriffe oder Schlüsselworte zu erklären. Dementsprechend ist eine Zuordnung zwischen notwendigem Feedback und den in der Programmierung zu erwerbenden Kompetenzen und Lernzielen zu erstellen. Diese

Zuordnung stellt das Ziel der dem Artikel übergeordneten Promotion dar und wird aktuell im Detail anhand verschiedener Online-Tools, sowie unter Berücksichtigung des aktuellen Forschungsstandes ausgearbeitet.

Eigene Ansätze zur Entwicklung eines Prototyps basierend auf der Blockly Bibliothek [Ki16] machen weiterhin deutlich, wie aufwendig sich die Vorbereitung und Umsetzung von automatisiertem, individualisiertem Feedback gestaltet - von der Forderung nach Adaptivität ganz zu schweigen. Trotzdem bietet sich die Entwicklung und Integration digitaler Tools und Feedback in die Programmierausbildung durchaus an und sollte vermehrt unterstützt und untersucht werden.

Literaturverzeichnis

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

[Br13] Brandstätter V.; Lozo, L.; Puca, R.; Schüler J.: Motivation und Emotion. Springer Verlag, Berlin, Heidelberg, 2013.

[DR93] Deci, E. L.; Ryan, R. M.: Die Selbstbestimmungstheorie und ihre Bedeutung für die Pädagogik. Zeitschrift für Pädagogik 39. Jg./1993 Nr. 2, S. 223-238, 1993.

[Fr15] Fraser, N.: Ten things we've learned from Blockly, IEEE Blocks and Beyond Workshop (Blocks and Beyond), Atlanta, GA, 2015, S. 49-50, 2015.

[GM07] Gomes, A.; Mendes, A. J.: Learning to program-difficulties and solutions. International Conference on Engineering Education–ICEE, Vol. 2007.

[Go18] Google Developers: Blockly, https://developers.google.com/blockly/, Abrufdatum: 8.6.2018.

[He10] Heckhausen, J.: Motivation und Handeln, Springer Verlag, Berlin, Heidelberg, 2010.

[He17] Heublein, U., et al.: Zwischen Studienerwartungen und Studienwirklichkeit: Ursachen des Studienabbruchs, beruflicher Verbleib der Studienabbrecherinnen und Studienabbrecher und Entwicklung der Studienabbruchquote an deutschen Hochschulen, Forum Hochschule 1/2017, 2017.

[Ke00] Keller, J.: How to integrate learner motivation planning into lesson planning: The ARCS model approach. VII Semanario, Santiago, Cuba, 2000.

[Ki16] Kiesler, N.: Ein Bild sagt mehr als tausend Worte – interaktive Visualisierungen in web-basierten Programmieraufgaben. Delfi 2016 Proceedings, Gesellschaft für Informatik e.V., Bonn, S. 335-337, 2016.

[Ma81] Mayer, R. E.: The Psychology of How Novices Learn Computer Programming. ACM Comput. Surv. 13(1), S. 121-141, 1981.

[Mo18] Moodle (Hrsg.), Welcome to Moodle CodeRunner, www.coderunner.org.nz/, Abrufdatum: 9.4.2018.

[Na06] Narciss, S.: Informatives tutorielles Feedback, Waxmann, Münster, 2006.

[Pa17] Parlante, N., CodingBat, www.codingbat.com/java, Abrufdatum: 9.4.2018.

[PFM17] Pasternak, E.; Fenichel, R.; Marshall, A. N.: Tips for creating a block language with blockly," 2017 IEEE Blocks and Beyond Workshop (B&B), Raleigh, NC, USA, S. 21-24, 2017.

[Wi96] Winslow, Leon E.: Programming pedagogy—a psychological overview. SIGCSE Bull. 28(3), S. 17-22, 1996.

[Xi14] Xinogalos, S.: Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. Education and Information Technologies. Springer Science+Business Media, New York, 2014.