Clemens SAUERWEIN[1] (Innsbruck), Ruth BREU (Innsbruck), Stefan OPPL (Krems), Iris GROHER (Linz), Tobias ANTENSTEINER (Innsbruck), Stefan PODLIPNIG (Wien) & Radu PRODAN (Klagenfurt)

CodeAbility Austria – Digital gestützte Programmierausbildung an österreichischen Universitäten

Zusammenfassung

Qualitativ hochwertige Programmierausbildung an Universitäten stellt aufgrund von stark steigenden Zahlen von Studierenden, knapp bemessenen Lehrebudgets und Mangel an Lehrkräften eine große Herausforderung dar. Ziel des Projekts „CodeAbility Austria“ ist es, diesen universitären Rahmenbedingungen gerecht zu werden und Programmierlernplattformen bereitzustellen. Im Rahmen dieses Beitrags stellen wir das Projekt näher vor, präsentieren Ergebnisse unserer empirischen Untersuchungen hinsichtlich der Erfahrungen und Herausforderungen im Umgang mit Programmierlernplattformen in der universitären Lehre und geben einen Ausblick auf zukünftige Arbeiten.

Schlüsselwörter

Programmierausbildung, Programmierlernplattformen, Empirische Untersuchung

1 E-Mail: Clemens.Sauerwein@uibk.ac.at

DOI: 10.3217/zfhe-SH-HL/06

CodeAbility Austria – Digitally supported programming education at Austrian universities

Abstract

High-quality programming education at universities is a significant challenge due to rapidly increasing student numbers, tight teaching budgets and a shortage of instructors. The “CodeAbility Austria” project aims to meet this challenge by establishing suitable programming learning platforms. In this paper, we introduce the project in more detail, present the results of our empirical research on the experiences and challenges of using programming learning platforms, and provide an outlook for future work.

Keywords

programming education, programming learning platforms, empirical study

1 Einleitung

Der Zugang zu einer qualitativ hochwertigen Programmierausbildung für Studierende aller Studienrichtungen bildet das Fundament für viele Weiterentwicklungen in Lehre und Forschung an heutigen Universitäten. Die Realisierung des Zugangs muss derzeit häufig unter den folgenden Rahmenbedingungen erfolgen: Eine stark steigende Zahl an Studierenden, die sich für eine Programmierausbildung interessieren, knapp bemessene Lehrebudgets, aber auch der Mangel an verfügbaren Lehrkräften, sowie die Qualität der Ausbildung (MEKTEROVIĆ & BRKIĆ, 2017). In Bezug auf das Erreichen von Lernzielen liegt die Herausforderung darin, Studierende dazu zu befähigen, Problemstellungen zu verstehen und eigenständig Lösungen zu entwickeln. Das Zusammenspiel von selbstständigem Üben und individuellem Feedback ist dabei eine zentrale Komponente (KEUNING et al., 2018).

Ziel von CodeAbility Austria ist es, an den sieben beteiligten Universitäten Strukturen und Plattformen zu schaffen, die diesen Rahmenbedingungen gerecht werden. Das Projekt baut dabei auf drei Ebenen auf: Auf der didaktischen Ebene werden Konzepte entwickelt und evaluiert, um Präsenz- mit Online-Elementen in der Programmierausbildung zu kombinieren. Auf technologischer Ebene wird eine Programmierlernplattform mit Learning Analytics eingeführt und eingesetzt, um einerseits Lehrende beim Entwickeln von individuellen, sich nach Lernniveau unterscheidenden Aufgabenstellungen sowie andererseits Studierende beim Lösen von Aufgaben zu unterstützen. Auf organisatorischer Ebene fördern wir die Vernetzung von Lehrenden und den Austausch von Lehrmaterialien.

In diesem Beitrag werden wir zum einen das CodeAbility-Austria-Projekt und zum anderen Ergebnisse unserer Begleitforschung präsentieren. Ziel der Begleitforschung ist es, in Form von empirischen Untersuchungen Anforderungen für Programmierlernplattformen zu erheben und die Qualität des Lernens bzw. Lehrens unter Nutzung einer solchen Plattform zu evaluieren. In einem ersten Schritt haben wir Erfahrungen bei der Nutzung einer Programmierlernplattform in einer Pilotphase untersucht. Dafür haben wir 15 Lehrende und 15 Studierende an vier österreichischen Universitäten in Form einer Umfrage und Experteninterviews befragt. Mit unseren Untersuchungen haben wir Erfahrungen und Herausforderungen von Lehrenden und Studierenden beim Umgang mit Programmierlernplattformen herausgearbeitet.

Der Beitrag ist wie folgt aufgebaut: In Kapitel 2 stellen wir das CodeAbility-Austria-Projekt vor und führen in die grundsätzliche Funktionsweise von Programmierlernplattformen unter Rückgriff auf relevante Forschungsarbeiten ein. In Kapitel 3 beschreiben wir das methodische Vorgehen unserer empirischen Untersuchungen zu den Erfahrungen mit und der Nutzung von Programmierlernplattformen durch Studierende und Lehrende. Die Ergebnisse unserer Untersuchung präsentieren wir in Kapitel 4, in Kapitel 5 werden sie diskutiert, um die bisherigen Haupterkenntnisse unseres Projekts abzuleiten. Mit Kapitel 6 schließen wir unseren Beitrag und geben einen Ausblick auf zukünftige Arbeiten.

2 Hintergrundinformationen

In diesem Kapitel stellen wir zunächst das CodeAbility-Austria-Projekt (siehe 2.1) und die Herausforderungen in der Lehre (siehe 2.2) vor. In einem weiteren Schritt diskutieren wir die grundsätzliche Funktionsweise von Programmierlernplattformen anhand relevanter Forschungsarbeiten (siehe 2.3).

2.1 CodeAbility Austria

Das Projekt CodeAbility Austria wird im Rahmen der Initiative „Digitale und soziale Transformation in der Hochschulbildung“ des Österreichischen Bundesministeriums für Bildung, Wissenschaft und Forschung über eine fünfjährige Laufzeit von 1. Januar 2020 bis 31. Dezember 2024 gefördert. Das Konsortium umfasst die sieben österreichische Universitäten Universität Innsbruck (Konsortialleitung), Technische Universität Wien, Technische Universität Graz, Johannes-Kepler-Universität Linz, Paris-Lodron-Universität Salzburg, Alpen-Adria-Universität Klagenfurt und Universität für Weiterbildung Krems.

CodeAbility Austria konzentriert sich auf Lehrveranstaltungen, die in die Grundfertigkeiten des Programmierens einführen. Der Schwerpunkt liegt dabei auf dem Übungsteil der universitären Kurse, in dem die Studierenden alleine oder im Team sukzessive kleine Programmieraufgaben bearbeiten und dadurch die Kompetenz erwerben, zu einer Problemstellung selbstständig korrekten und lauffähigen Quellcode zu entwickeln. Zusätzlich werden die Studierenden von Dozent:innen unterstützt, die ihnen formatives Feedback, Hilfestellungen und Verbesserungsvorschläge geben sowie motivierend auf den Lernerfolg der Gruppe während der Lehrveranstaltungseinheit reagieren. Die Interaktion zwischen Lehrenden und Lernenden kann unterschiedlich sein, z. B. synchron oder asynchron, im Präsenzunterricht oder online. Da die Lehrenden auch für die Beurteilung der Studierenden verantwortlich sind, unterscheidet sich die in unserem vorliegenden Beitrag behandelte Lernsituation von anderen, teilweise non-formalen (Weiter-)Bildungsangeboten, die beispielsweise ohne Instruktor:innen auskommen.

Die Lehrkontexte im Projektkonsortium weisen Unterschiede auf, z. B. in Bezug auf die Programmiersprachen und die curriculare Einbettung der einzelnen Lehrveranstaltungen. Zu den infrage kommenden Lehrveranstaltungen gehören typischer-

weise einführende Programmierkurse im 1. Semester des Bachelorstudiums *Informatik, Wirtschaftsinformatik* oder *Betriebswirtschaftslehre*.

2.2 Herausforderungen in der Lehre

Die Realität der Lehre stellt Studienorganisatoren, Lehrkräfte und Studierende vor zahlreiche Herausforderungen. Dazu zählt unter anderem der latente Mangel an erfahrenen Lehrenden, heterogenes Vorwissen der Studierenden sowie Zeitmangel der Lehrenden, sich kontinuierlich einen Überblick über die aktuellen Leistungen der Studierenden zu verschaffen, um ihnen z. B. individuelles Feedback geben zu können und den Inhalt ihres Kurses entsprechend zu adaptieren.

Hinzu kommt, dass mit der wachsenden Bedeutung digitaler Methoden in allen Wissenschaftsdisziplinen die Zahl der Studierenden, die eine Programmierausbildung benötigen oder wünschen, stark steigt. Somit müssen auch nicht-technische Studienprogramme in die Programmierausbildung integriert werden. Beispielsweise wird an der Universität Innsbruck ein 30-ECTS-AP-Wahlpaket *Digital Science* angeboten, das Studierende unterschiedlicher Disziplinen als gesamtes oder durch Absolvierung einzelner Module in den Wahlbereich ihres Studiums integrieren können. Die Nachfrage nach der Programmierlehrveranstaltungen in diesem Wahlpaket liegt regelmäßig bei über 70 Prozent der ursprünglich bemessenen Lehrkapazität.

Wir sehen im Projekt den Einsatz einer Programmierlernplattform vor, um mit den vorgenannten Herausforderungen umzugehen. Mehrere Lehrende an den im Konsortium beteiligten Universitäten verfügten bereits vor Projektbeginn über Erfahrungen mit verschiedenen Programmierlernplattformen. Als gemeinsame Ziele wurden primär die Verbesserung des mitunter automatisiert generierten Feedbacks für Lernende, die Unterstützung der Lehrenden, insbesondere in virtuellen Lehrveranstaltungen, sowie die Bewältigung steigender Studierendenzahlen ins Auge gefasst.

2.3 Programmierlernplattformen

Zu den primären Zielen von Programmierlernplattformen gehören die Steigerung der Motivation von Studierenden, Monitoring des Lernfortschritts, Qualitätsverbesserung der Lehre sowie der von Studierenden abgegebenen Lösungen, die Minimierung der Einstiegshürde für Programmierneulinge, die Erhöhung der Objektivität sowie Standardisierung von Feedback und die Reduktion der Studienabbruchquote (KEUNING et al., 2018; MEKTEROVIĆ et al., 2020). Grundsätzlich unterstützen Programmierlernplattformen die Kommunikation von Lehrenden, z. B. durch Bereitstellung von Übungsaufgaben oder effizientes Handling eingereichter Lösungen. Studierende werden beispielsweise mit dem Zugriff auf Programmieraufgaben und vorgegebene Programmteile oder Einreichung von Lösungen unterstützt. Darüber hinaus können Lehrende eingereichte Programme der Studierenden durch bereitgestellte und selbst definierte Kontrollen und Auswertungen prüfen lassen, z. B. durch Ausführen von Testfällen, Prüfung der Struktur der Lösung oder Plagiat-Checks, und erhalten hilfreiche Informationen zu den Leistungen ihrer Lehrveranstaltungsteilnehmenden. Studierende können Lösungen mehrmals einreichen und somit aus den Kontrollen und Auswertungen sowie dem damit verbundenen automatisch generierten Feedback lernen und haben während der Lehrveranstaltung stets eine Übersicht über erzielte Leistungen.

Mithilfe von Progammierlernplattformen gelingt es, einige der mit manueller Auswertung von Programmieraufgaben einhergehenden Hürden, wie beispielsweise objektives und effizientes Bewerten bei großen Studierendenzahlen und das Bereitstellen von zeitnahem, individuellem und hilfreichem Feedback, zu umgehen (MEKTEROVIĆ & BRKIĆ, 2017). In den letzten Jahren wurde eine Vielzahl von Programmierlernplattformen, wie beispielsweise „Checkpoint“ (ENGLISH & ENGLISH, 2015), „JACK“ (GOEDICKE et al., 2008) oder „ArTEMiS“ (KRUSCHE & SEITZ, 2018) publiziert (KEUNING et al., 2016, 2018; MEKTEROVIĆ et al., 2020).

Um für das CodeAbility-Austria-Projekt eine geeignete Programmierlernplattform zu finden, wurden „Checkpoint“ (ENGLISH & ENGLISH, 2015), „JACK“ (GOEDICKE et al., 2008) und „ArTEMiS“ (KRUSCHE & SEITZ, 2018) hinsichtlich angebotener Funktionalität, Softwarearchitektur und Softwarelizenz evaluiert. Basierend auf den Evaluierungsergebnissen wurde KRUSCHEs und SEITZ’ (2018) „Automated Assessment Management System“ (ArTEMiS) ausgewählt, da es alle grundlegenden Eigenschaften einer Programmierlernplattform, z. B. Management

von Programmieraufgaben, Testen von Programmieraufgaben oder Generieren von individuellem Feedback, unterstützt. Darüber hinaus ist ArTEMiS programmiersprachenagnostisch, quelloffen und wird aktiv weiterentwickelt.

Aufgrund der Wahl von ArTEMiS entschied sich das Konsortium für eine Kooperation mit der Technischen Universität München, die die Weiterentwicklung von ArTEMiS hauptverantwortlich koordiniert. Seit dem Start der Pilotphase im Oktober 2020 wurden 13 Lehrveranstaltungen mit über 1000 Teilnehmenden über die CodeAbility ArTEMiS Instanz abgewickelt und dabei über 300 Programmieraufgaben in den drei Programmiersprachen C/C++, Java, Python erstellt.

3 Methodisches Vorgehen

Die Einführung der Programmierlernplattform wurde von Beginn an durch wissenschaftliche empirische Studien begleitet. Dabei wurden die Erfahrungen und Herausforderungen von Lehrenden und Studierenden bei der Nutzung von Programmierlernplattformen untersucht. Zum einen sollten diese Untersuchungen neue Anforderungen für die Weiterentwicklung der eingesetzten Programmierlernplattform liefern, zum anderen generalisierbare, programmierlernplattformunabhängige Resultate für die Lern- und Lehrpraxis liefern.

Im Zuge dieser Untersuchungen wurde eine Fallstudie über den Verlauf eines Semesters an der Universität Innsbruck, Johannes-Kepler-Universität Linz, Paris-Lodron-Universität Salzburg und Alpen-Adria Universität Klagenfurt durchgeführt. Ziel dieser Untersuchung war, die Erfahrungen von Lehrenden und Studierenden im Umgang mit der Programmierlernplattform zu untersuchen.

In einem ersten Schritt wurden die Dozent:innen der beteiligten Universitäten im Rahmen eines Workshops mit der Benutzung und dem Umgang mit der Programmierlernplattform vertraut gemacht. Insgesamt beteiligten sich 15 Dozent:innen der vier beteiligten Universitäten an den Workshops.

In einem zweiten Schritt setzten die Dozent:innen die Programmierlernplattform in sieben unterschiedlichen Programmierlehrveranstaltungen an denselben vier Universitäten in einem Semester ein. Diese Lehrveranstaltungen waren primär Einführungskurse in die Programmierung für Studierende der Bachelorstudien *Informa-*

tik und *Wirtschaftsinformatik* oder anderer Studienprogrammen mit einem starken Fokus auf Digitalisierung, z. B. Erweiterungsstudien oder Wahlpakete. Insgesamt wurden ein C-Kurs, drei Java-Kurse und drei Python-Kurse mit der Programmierlernplattform durchgeführt. Dafür wurden 38 C-, 168 Java- und 17 Python-Aufgaben in die Plattform eingepflegt. Pro Aufgabe wurden durch die Dozent:innen entsprechende Kontrollen und Auswertungen definiert, um eine automatische Überprüfung der Ergebnisse zu ermöglichen. Die Studierenden mussten die Programmieraufgaben als Hausaufgabe lösen und diese über die Programmierlernplattform einreichen. Die Korrektur dieser Programmieraufgaben wurde durch die Plattform automatisiert durchgeführt, sodass die Studierenden sofortiges Feedback zu ihren Abgaben erhielten.

In einem dritten und letzten Schritt wurden jeweils 15 Dozent:innen und Studierende in Form von individuellen Expert:inneninterviews bzw. Studierendenbefragung hinsichtlich ihrer Erfahrungen und Herausforderungen mit der Programmierlernplattform zu Semesterende befragt. Abschließend wurden die Ergebnisse zusammengeführt und analysiert, um qualitative Aussagen zu treffen und Informationen zu extrahieren (CAMPBELL et al., 2013).

4 Ergebnisse

In diesem Kapitel diskutieren wir die Ergebnisse der Expert:inneninterviews (siehe 4.1) sowie der Studierendenbefragung (siehe 4.2).

4.1 Erfahrungen der Lehrenden

Die Dozent:innen gaben die Rückmeldung, dass der Einsatz der Programmierlernplattform eine objektivere Leistungsbeurteilung, verglichen mit herkömmlichen, manuellen Beurteilungsmethoden, zulässt. Dies ist darauf zurückzuführen, dass die Bewertung aller Abgaben mit den gleichen Kontrollen und Auswertungen durch ein System und nicht durch mehrere, subjektiv bewertende Lehrende erfolgt.

Während der Bearbeitung der Übungsaufgaben durch die Studierenden kann der Lernfortschritt der Lernenden durch die Programmierlernplattform untersucht so-

wie überprüft werden. Dieses Monitoring wurde von den Lehrenden als sehr hilfreich empfunden, da dadurch unter anderem Wissenslücken oder Fehlvorstellungen der Studierenden, gerade bei einer hohen Anzahl von Lehrveranstaltungsteilnehmenden, frühzeitig erkannt werden können und schließlich die Qualität der Lehre verbessert.

Bei den Abgaben der Studierenden konnten die Lehrenden feststellen, dass durch die automatisierten Tests die Lösungen genau die erwartete Funktionalität implementieren. Dadurch verschob sich der Fokus in den Lehrveranstaltungseinheiten von der Korrektur einer präsentierten Lösung zur Diskussion von verschiedenen Lösungsansätzen sowie der Codequalität. Jedoch wurde auch bemerkt, dass der testgetriebene Einsatz der Programmierlernplattform gewisse Einschränkungen bei der Kreativität und Problemlösungsfähigkeit der Studierenden mit sich bringt.

Als negativ wurde von den Dozent:innen empfunden, dass der Einsatz einer Programmierlernplattform als Insellösung nicht optimal ist, da die meisten Universitäten für die Kursabwicklung bereits Learning Management Systeme (LMS), z. B. Moodle oder OpenOLAT, einsetzen. Demzufolge wünschen sich Lehrende eine Integration der Programmierlernplattform in bestehende Softwarelösungen.

Vor Kursbeginn müssen die Lehrenden die Programmieraufgaben entsprechend in die Programmierlernplattform einpflegen. Es ist notwendig, dass sie dafür zum einen die einzelnen, in der Lehrveranstaltung verwendeten Übungsaufgaben anlegen und zum anderen Kontrollen und Auswertungen mit entsprechenden Feedbacktexten für die automatisierte Korrektur der Programmieraufgaben erstellen. Dieser testgetriebene Ansatz stellt laut den Befragten einen nicht zu vernachlässigenden Mehraufwand für Lehrende bei der Erstellung von Programmieraufgaben dar.

4.2 Erfahrungen der Studierenden

Die Studierenden beziehen ihre Übungsaufgaben über die Programmierlernplattform und können diese auf zwei Wegen bearbeiten: Entweder sie lösen die Programmieraufgaben im Programmiereditor der Programmierlernplattform oder sie bearbeiten diese in lokalen Entwicklungsumgebungen und geben die gelösten Aufgaben über die Programmierlernplattform ab. Nach der Einreichung werden die Lösungen mittels der vordefinierten Kontrollen und Auswertungen überprüft. Im Zuge

dieser Kontrolle gibt die Programmierlernplattform den Studierenden formatives Feedback. Sollte eine Abgabe fehlerhaft oder unzureichend gelöst sein, können Studierende diese beliebig oft erneut einreichen.

Lehrende beobachten, dass Studierende dazu tendieren, ein gewisses Versuch-und-Irrtum-Verhalten an den Tag legen: Sie versuchen, so lange ihre Abgaben zu verbessern und wiederholt einzureichen, bis die Programmierlernplattform kein negatives Feedback liefert. In diesem Zusammenhang wurde von Studierenden erwähnt, dass das Feedback zwar hilfreich ist, jedoch noch zielgerichteter sein könnte. Darüber hinaus wurde von den befragten Studierenden auch der Wunsch nach individuellen Lernpfaden geäußert, deren konzeptionellen Ansätze heterogene Lernvoraussetzungen wie -interessen annehmen, um diese bei der Lernumgebungsgestaltung zu berücksichtigen (HANFT et al., 2019). Dies ist mitunter darauf zurückzuführen, dass die eingesetzte Lernplattform die Reihenfolge der zu bearbeitenden Aufgaben vorgibt und nicht auf die individuellen Probleme der Studierenden eingeht.

5 Diskussion

Basierend auf den Erfahrungen der Lehrenden und Studierenden leiten wir in diesem Kapitel drei Haupterkenntnisse ab, die zu Verbesserungsmöglichkeiten beim Einsatz von Programmierlernplattformen führen und somit potenzielle zukünftige (Forschungs-)Themen im CodeAbility-Austria-Projekt darstellen.

5.1 Aufgabenerstellung

Für Lehrende hat sich der zu investierende Zeit- und Arbeitsaufwand von der Korrektur von Programmieraufgaben hin zur Ausarbeitung von testbaren Übungsaufgaben, für die umfangreiches und einfach verständliches Feedback generiert wird, verlagert. Im Vergleich zu herkömmlichen Programmieraufgaben ist es bei automatisiert ausgewerteten Übungsaufgaben unerlässlich, dass die Aufgabenstellung unmissverständlich formuliert ist und genau vorgibt, was erwartet wird, um das Überprüfen der erwarteten Funktionalität mittels automatisierter Kontrollen und Auswertungen zu ermöglichen. Neben der genaueren Aufgabenspezifikation gehört zum Erstellen einer Übungsaufgabe das Implementieren der Kontrollen und Aus-

wertungen und – abhängig von der eingesetzten Testart – das Design des Feedbacks für eine Programmieraufgabe. All diese Faktoren führen zu einer Vervielfachung des notwendigen Zeitaufwands, um eine Programmieraufgaben vollständig auszuarbeiten.

Aus diesem Grund empfehlen wir, vor dem Einsatz einer Programmierlernplattform abzuwägen, ob der mit der Aufgabenerstellung einhergehende Mehraufwand gerechtfertigt ist. Um dem erhöhten Zeit- sowie Arbeitsaufwand bei Rückgriff auf Programmierlernplattformen entgegenzuwirken, ist ein Austausch von Unterrichtsmaterialien zwischen Lehrenden erfolgversprechend (siehe 5.3).

Bezogen auf die Studierenden bestand aus Sicht der Lehrenden aufgrund der genaueren Aufgabenspezifikation die Befürchtung, dass die Kreativität in der Lösungsfindung eingeschränkt wird. Dieses Bedenken hat sich beim Einsatz der Programmierlernplattform in Lehrveranstaltungen für Programmieranfänger:innen allerdings nicht bestätigt. Vielmehr kam es zu einer merklichen Verbesserung der Qualität der abgegebenen Lösungen. Jedoch hat sich auch gezeigt, dass bei komplexeren Programmieraufgaben die Problemlösungskompetenz und -kreativität der Lernenden eingeschränkt wird und somit auch Grenzen des Einsatzes bestehen.

5.2 Formatives Feedback

Das festgestellte Versuch-und-Irrtum-Verhalten der Studierenden bei der Lösung von Programmieraufgaben (siehe 4.2) motiviert, die Bereitstellung von individuellem, formativem Feedback beim Einsatz von Programmierlernplattformen weiter zu erforschen, um das Potenzial automatisierter Auswertung von Programmcodes (IHANTOLA et al., 2010) ausschöpfen zu können.

In Einführungskursen in die Programmierung erhalten Studierende – unabhängig davon, ob automatisiert oder nicht – formatives Feedback (BLACK & WILIAM, 2009) in erster Linie beim Bearbeiten von Programmieraufgaben: nicht nur klassisch durch Hinweise und Instruktionen von Lehrpersonen, sondern auch durch Programmierwerkzeuge, z. B. integrierte Entwicklungsumgebungen (eng. Integrated Development Environment, IDE). Im Regelfall stellt dieses formative Feedback eine kontinuierliche Rückmeldungen zu Programmierfehlern und -warnungen (engl. Pro-

gramming Error Messages, PEMs) (Zhou et al., 2021) sowie, je nach verwendeter Software, etwaigen Verbesserungsvorschlägen dar.

Insbesondere für Programmierneulinge ist es entscheidend, PEMs derart zu gestalten, dass sie für Studierende verständlich sind, da diese als primäre Informationsquelle dienen, um Novizen bei der Behebung von Programmierfehlern zu unterstützen (BECKER et al., 2019). Daraus resultierende, potenzielle weiterführende Forschungsthematiken sind unter anderem eine Klassifizierung von Programmierfehlern auf der Grundlage eines Kompetenzmodells, eine Ursachenanalyse für das Auftreten von Programmierfehlern, wie z. B. tiefgreifende Fehlvorstellungen von Programmierkonzepten oder rein syntaktische Fehler (STAUB & CHOTHIA, 2022), sowie eine Untersuchung der Möglichkeiten und Grenzen von PEMs (siehe 4.1.) als (formative) Feedbackmethode von in Programmierlernplattformen integrierten IDEs (KOHN, 2017).

Der von Expert:innen vorhergesagte Bedeutungszuwachs von Learning Analytics in der Hochschullehre (LANG et al., 2022) inspiriert darüber hinaus eine Untersuchung der Möglichkeiten der aus PEMs abgeleiteten Informationen für individuelle Lernpfade (siehe 4.2), die aus Kompetenzgraphen (LICHTENBERG & OLIVER, 2016) und den bereits erzielten Leistungen der Lernenden generiert werden.

5.3 Austauschplattform

Um dem Mehraufwand von Lehrenden bei der Erstellung der Programmieraufgaben entgegenzuwirken, ist es sinnvoll, eine Austauschplattform für Lernmaterialien zu entwickeln. Durch den Austausch von Übungsaufgaben kann der Aufwand des Erstellens von testbaren Übungsaufgaben auf eine größere Community aufgeteilt werden. Als zusätzlicher Nebeneffekt kann durch den Austausch mit anderen Dozent:innen und entsprechenden Reviewprozessen von Programmieraufgaben ein vielfältiger Pool qualitativ hochwertiger Übungsaufgaben erstellt und gepflegt werden.

Aus diesen Gründen wird im Zuge des CodeAbility-Austria-Projekts eine Sharing-Plattform für die Programmierlehre entwickelt. Sie unterstützt den nahtlosen Austausch von Übungen von und zur Programmierlernplattform. Um die Usability dieser Plattform zu erhöhen, werden die Inhalte basierend auf Kompetenzmodellen

mit Metadaten annotiert. Ebenso wird neben einzelnen Programmieraufgaben auch der Austausch von größeren Einheiten, z. B. Übungssuiten oder ganzen Lehrveranstaltungen, umgesetzt. Erste Kontakte zur Integration mit dem OERHub wurden aufgenommen.

6 Schlussfolgerung und Ausblick

Grundsätzlich besteht eine große Akzeptanz der eingesetzten Programmierlernplattform im Projekt CodeAbility Austria. Die Lehrenden sehen Effizienzgewinne bei der Korrektur der Programme und einen Effektivitätsgewinn beim Monitoring des Lernfortschritts, z. B. ist es möglich, Fehlvorstellungen der Studierenden früh zu erkennen und darauf zu reagieren. Studierende schätzen das sofortige Feedback beim Einreichen der Abgaben, die Möglichkeit, fehlerhafte Lösungen selbst korrigieren zu können und eine transparente Bewertung der Übungen zu erhalten. Verbesserungen werden von den Lehrenden in Richtung einer Integrationsmöglichkeit der Programmierlernplattform in bestehende LMS gewünscht.

Zu den Faktoren, die beobachtet werden müssen, zählt, dass sich manche Studierende zu einem Versuch-und-Irrtum-Verhalten verleiten lassen, das sich auf die wenig aufgabenorientierte Überarbeitung des eigenen Quellcodes auf Basis des zur Verfügung gestellten Feedbacks fokussiert. Um diesem Verhalten entgegenzuwirken, ist zielgerichtetes formatives Feedback erforderlich.

Zu den Haupterkenntnissen aus dem Pilotbetrieb zählt außerdem, dass der zu investierende Zeit- und Arbeitsaufwand für die Erstellung einer qualitativ hochwertigen Programmieraufgabe, die z. B. ein aussagekräftiges Feedback bei einer fehlerhaften Lösung bereitstellt, für die Lehrenden hoch ist. Der Austausch von Übungen, z. B. als Open Educational Resources (OER), gewinnt dadurch an Bedeutung.

Demensprechend werden sich die zukünftigen Arbeiten im CodeAbility-Austria-Projekt mit der Verbesserung und Weiterentwicklung des durch die Programmierlernplattform automatisiert gegebenen Feedbacks und dem Austausch von Lehrunterlagen beschäftigen.

7 Literaturverzeichnis

Becker, B. A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D. J., Harrington, B., ... & Prather, J. (2019). Compiler error messages considered unhelpful: The landscape of text-based programming error message research. In *Proceedings of the working group reports on innovation and technology in computer science education* (S. 177–210). https://doi.org/10.1145/3344429.3372508

Black, P. & Wiliam, D. (2009). Developing the theory of formative assessment. *Educational Assessment, Evaluation and Accountability (formerly: Journal of Personnel Evaluation in Education), 21*(1), 5–31.

Campbell, J. L., Quincy, C., Osserman, J. & Pedersen, O. K. (2013). Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement. *Sociological Methods & Research, 42*(3), 294–320.

Ellegaard, M., Damsgaard, L., Bruun, J. & Johannsen, B. F. (2018). Patterns in the form of formative feedback and student response. *Assessment & Evaluation in Higher Education, 43*(5), 727–744.

English, J. & English, T. (2015). Experiences of using automated assessment in computer science courses. *Journal of Information Technology Education: Innovations in Practice, 14*, 237–254.

Goedicke, M., Striewe, M. & Balz, M. (2008). *Computer aided assessments and programming exercises with jack (tech. rep.)*. ICB-Research Report. https://doi.org/10.17185/duepublico/47108

Hanft, A., Kretschmer, S. & Hug, V. (2019). Hochschullehre aus der Studierenden-Perspektive denken: Individuelle Lernpfade im Inverted Classroom. *Zeitschrift für Hochschulentwicklung, 14*(3), 323–340.

Ihantola, P., Ahoniemi, T., Karavirta, V. & Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli calling international conference on computing education research* (S. 86–93). https://dl.acm.org/doi/10.1145/1930464.1930480

Keuning, H., Jeuring, J. & Heeren, B. (2016). Towards a systematic review of automated feedback generation for programming exercises. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (S. 41–46). https://dl.acm.org/doi/10.1145/2899415.2899422

Keuning, H., Jeuring, J. & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education (TOCE), 19*(1), 1–43.

Kohn, T. (2017). *Teaching Python programming to novices: Addressing misconceptions and creating a development environment*. ETH Zurich.

Krusche, S. & Seitz, A. (2018). Artemis: An automatic assessment management system for interactive learning. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (S. 284–289). https://dl.acm.org/doi/10.1145/3159450.3159602

Lang, C., Siemens, G., Wise, A. & Gasevic, D. (Hrsg.). (2022). *Handbook of learning analytics* (2. Aufl.). Vancouver: SoLAR.

Lichtenberg, G. & Oliver, R. (2016). Kompetenzgraphen zur Darstellung von Prüfungsergebnissen: ein Visualisierungsinstrument für individualisierte Leistungsbeobachtungen. In B. Berendt, A. Fleischmann, N. Schaper, B. Szczyrba & J. Wildt (Hrsg.), *Neues Handbuch Hochschullehre* (S. 99–120). Berlin: DUZ Verlags- und Medienhaus GmbH.

Mekterović, I. & Brkić, L. (2017). Setting up automated programming assessment system for higher education database course. *International Journal of Education and Learning Systems, 2,* 287–294.

Mekterović, I., Brkić, L., Milašinović, B. & Baranović, M. (2020). Building a comprehensive automated programming assessment system. *IEEE Access, 8*, 81154–81172.

Staub, J. & Chothia, Z. (2022). Large-Scale Analysis of Error Frequencies in Logo Programming. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1* (S. 571–577). https://dl.acm.org/doi/10.1145/3478431.3499393

Zhou, Z., Wang, S. & Qian, Y. (2021). Learning From Errors: Exploring the Effectiveness of Enhanced Error Messages in Learning to Program. *Frontiers in Psychology, 12*. https://doi.org/10.3389/fpsyg.2021.768962

Autor:innen

Ass.-Prof. Clemens SAUERWEIN, PhD || Universität Innsbruck, Institut für Informatik || Technikerstraße 21a, A-6020 Innsbruck

https://www.uibk.ac.at/informatik/

Clemens.Sauerwein@uibk.ac.at

Univ.-Prof. Dr. Ruth BREU || Universität Innsbruck, Institut für Informatik || Technikerstraße 21a, A-6020 Innsbruck

https://www.uibk.ac.at/informatik/

Ruth.Breu@uibk.ac.at

Univ.-Prof. Dr. Stefan OPPL || Universität für Weiterbildung Krems, Department für Weiterbildungsforschung und Bildungstechnologien || Dr.-Karl-Dorrek-Straße 30, AT-3500 Krems

https://www.donau-uni.ac.at/dwb

Stefan.Oppl@donau-uni.ac.at

Assoz. Univ.-Prof. Dr. Iris GROHER || Johannes-Kepler-Universität Linz, Institut für Wirtschaftsinformatik - Software Engineering || Altenbergerstrasse 69, A-4040 Linz

https://se.jku.at/iris-groher/

Iris.Groher@jku.at

Tobias ANTENSTEINER || Universität Innsbruck, Institut für Informatik || Technikerstraße 21a, A-6020 Innsbruck

https://www.uibk.ac.at/informatik/

Tobias.Antensteiner@uibk.ac.at

Dr. Stefan PODLIPNIG || Technische Universität Wien, Institut für Logic and Computation, Forschungsbereich Theory and Logic, 192/5 || Favoritenstrasse 9, A-1040 Wien

https://informatics.tuwien.ac.at/people/stefan-podlipnig

Stefan.Podlipnig@tuwien.ac.at

Univ.-Prof. Dr. Radu PRODAN || Alpen-Adria Universität Klagenfurt, Institut für Informationstechnologie || Universitätsstr. 65-67, 9020 Klagenfurt am Wörthersee

https://itec.aau.at/

radu.prodan@aau.at