aus

Oliver J. Bott, Peter Fricke, Uta Priss, Michael Striewe (Hrsg.)

Automatisierte Bewertung in der Programmierausbildung

Digitale Medien in der Hochschullehre Band 6
2017, 420 Seiten, br., 42,90 €, ISBN 978-3-8309-3606-0

Waxmann Verlag GmbH
www.waxmann.com info@waxmann.com

25 Automatisierte Bewertung in der Programmierausbildung – Ausblick

Nils Jensen

Zusammenfassung
Das vorliegende Buch hat Einblicke in aktuelle Konzepte und Tools zum halb- oder vollautomatisierten Assessment in der hochschulspezifischen Programmierausbildung gegeben. In diesem Kapitel soll ausgehend von den erreichten, erprobten Ansätzen gezeigt werden, welche Herausforderungen in der praktischen Anwendung liegen und welche neuen Entwicklungs- und Forschungsansätze sich daraus ergeben. Die folgenden Ausführungen sind dabei weder umfassend noch erschöpfend. Gleichwohl sollen sie der Leserschaft Inspiration und Impuls sein.

25.1 Zielsetzung der hochschulspezifischen Programmierausbildung

An einer Hochschule sollen die Grundlagenkenntnisse in einer Programmiersprache, meistens Java, vermittelt werden. Vielfach schließt das Vorlesungseinheiten und Übungen zu Debugging, Korrektheit und Stil ein. Es ist hierbei nicht Aufgabe der Hochschulen, eine abgeschlossene Ausbildung in der Programmierung zu bieten. In der Praxis müssen daher die Kenntnisse durch „Training-on-the-job“ und maßgeschneiderte Schulungen vertieft werden. Nur wenige Informatikerinnen und Informatiker werden im Beruf langfristig als Programmierer eingesetzt werden.

Gleichwohl sollen viele Studierende durch den Erwerb der Grundlagenkenntnisse die Kompetenz erlangen, Probleme rational erfassen, in Teilaufgaben zerlegen und algorithmisch verallgemeinert lösen zu können. Als Computational Thinking bezeichnet man hierbei das Vermögen, realweltliche Prozesse konsequent

algorithmisch zu erfassen (vgl. [Guz08]). Der Auftrag der Programmierausbildung an Hochschulen ist demnach, im engeren Sinne, eine Programmiersprache als Werkzeug begreifbar zu machen, und die Programmierausbildung im weiteren Sinne als Vehikel zu nehmen, um Computational Thinking zu befördern.

25.2 Didaktik

In einem konstruktivistischen Ansatz wird man versuchen, den Studierenden vielfältige Möglichkeit zur individuellen Schaffung eigener Lernerlebnisse zu ermöglichen. Die Information des zu Vermittelnden steht damit als notwendiges, aber für viele Lernende nicht hinreichendes Element im Zentrum. In dieser Hinsicht sind die folgend genannten Tools und Innovationen eine Möglichkeit, den Studierenden einen erweiterten, selbstbestimmten und vielfältigen Raum zur Selbsterprobung und Selbstreflexion zu geben, in dem sie Mut fassen und Fortschritte erzielen können.

Die Begleitforschung zum Einsatz der genannten Tools steht in der ProFormA-Gruppe im eCULT-Team am Anfang. Es müssen beispielsweise Untersuchungen zur gleichbleibenden oder höheren Wirksamkeit beim Lernerfolg ab 2017, d. h. in der zweiten Förderphase des Projekts eCULT, durchgeführt werden. Das Messen eines Lernerfolgs im Vergleich zu einer Kontrollgruppe oder zu vorherigen Kohorten ist schwierig, weil in der Lehrpraxis die Rahmenbedingungen unvorhergesehen variieren können. Es ist deshalb notwendig, mit Didaktikern gemeinsam formative und summative Studien zu konzipieren.

Abseits dessen sollten folgende Punkte konkret im Vordergrund stehen:

- Review der Online-Aufgaben, insbesondere um Diskrepanzen zwischen automatisierter Beurteilung und Aufgabentexten aufzudecken.
- Kontrollierte Usability-Studien, in denen der Umgang mit dem System beobachtet und Defizite im System identifiziert werden, z. B. unverständliche Bedienung oder nicht nachvollziehbares Systemverhalten.
- Qualitative Beurteilungen durch Studierende und Lehrende in mehreren Semestern, z. B. über ein Online-Feedbacksystem.

Die Verbindung zwischen Learning Analytics und didaktischer Innovation ist seit 2011 im Projekt eCULT erkannt worden und stellt einen immer wichtiger werdenden Ansatz zur empirisch basierten Lehr- und Lerninnovation dar. Die Notwendigkeit der mentoriellen und tutoriellen Betreuung scheint dadurch nicht geschmälert, sondern vielmehr bestätigt und um hochentwickelte Werkzeuge zur Daten- und Nutzungsanalyse erweitert worden zu sein (vgl. [Clo13]).

Es sollte mithilfe der Learning Analytics untersucht werden, ob die Tools geeignet sind, sehr schwachen Studierenden die nötige Hilfe zu geben, basale Konzepte des Programmierens zu verinnerlichen. Nach unserer Erfahrung benötigt insbesondere die Gruppe der schwächeren Studierenden, jenseits des Wiederholens des Stoffes nach erfolgloser Prüfung, weitere Betreuung. Wo hierbei die Grenzlinie zwischen automatisierter und manueller Intervention verlaufen kann, ist nach unserer Meinung unklar und bedarf weiterer Analysen. Eine hierbei möglicherweise nützliche Theorie des Lernens mathematischer Inhalte stellt die APOS-Theorie von [DM02] dar (vgl. Kapitel 8). Diese Theorie könnte auf die Programmierausbildung adaptiert werden. Gelänge dies, so wäre gezeigt, dass ähnliche mentale Prozesse erforderlich sind, um jeweils Mathematik oder das Programmieren zu erlernen. Der Vorteil der APOS-Theorie ist folgender: In ihr können typische Schwierigkeiten vorhergesagt, analysiert und zerlegt werden. Darauf abgestimmt kann Lernmaterial entwickelt und ein Vorgehensmodell in Anlehnung an [MP04] befolgt werden. Unter Zuhilfenahme der Learning Analytics ergäbe sich somit ein empirisch fundierter, wissenschaftlich nachvollziehbarer, Zyklus zur Optimierung der Lehre.

25.3 Operationalisierung und Rollen

Die Lehrenden wählen Inhalt und Ziele der Lehrveranstaltung, anhand der Ziele die zu ergreifenden Tätigkeiten und, dazu passend, die Tests. Die Lehrenden helfen den Studierenden, die zu erwerbenden Kompetenzen einzuüben. Die Lehrassistenz und Tutorenschaft unterstützt sie in den operativen Aspekten einer Lehrveranstaltung durch Aus- und Aufarbeitung von Materialien, Kontaktzeit mit den Studierenden, sowie Übungsaufgaben, Tests und Prüfungen. Entsprechende Collaboration Tools legen bisher ihren Fokus auf die Interaktion zwischen Studierenden und Lehrenden, nicht aber auf die Interaktion zwischen arbeitsteilig wirkenden Lehrenden. Terminierung, Qualitätsprüfung und Arbeitsfeedback finden bisher informell statt und können in Learning Management Systemen (LMS), zum Beispiel Moodle, nicht standardisiert und formal erfasst werden. Oft werden mehrere Tools gleichzeitig eingesetzt, die Verzahnung und die gemeinsame Datenbasis schwinden dadurch. Das gilt insbesondere beim gleichzeitigen Einsatz mehrerer LMS um deren Vorteile zu vereinen: Beispielsweise bieten nicht alle LMS ein gemeinsames Single-Sign-On-System.

25.4 Tools

Aspekte des Collaborative und Blended Learning können geeignet sein, erfolgreich Lehre zu befördern, indem sie beispielsweise zu einer stärkeren Aktivierung der Studierenden führen und ihren individuellen Bedarfen näher kommen. Insbesondere kann der Vorteil darin bestehen, jederzeit und jederorts mit den Lehrenden in Kontakt zu treten, die beste Zeit zum Lernen selbst zu bestimmen, ortsunabhängig auf die Lehrmaterialien zuzugreifen und gegebenenfalls alternative grafische Schnittstellen nutzen zu können. Dieser Punkt dient auch der Barrierefreiheit. Weiterhin erlaubt das Collaborative und Blended Learning den Studierenden zwecks besserer Aktivierung miteinander zu beraten (Peer-teaching PT, Peer-instruction PI) und technische Setups zu bedienen, in denen insbesondere vermittels des Problem-Based Learning (PBL) praxisnähere Lehr-/Lernszenarien angeboten werden können.

Hierfür bieten die diskutierten Konzepte und Tools vielfache Anknüpfungspunkte. Die Systeme sind webbasiert und daher prinzipiell jederzeit und von überallher erreichbar. Integrierte Foren- und Chatfunktionen können zielgerichtet von Lehrenden und Studierenden eingesetzt werden. Mit beispielsweise Peer Reviews sollen Studierende in die Lage versetzt werden, untereinander ihre Lösungen konstruktiv zu kritisieren und ihre Fähigkeit zu verbessern, fremden Quelltext zu analysieren. Dies kann ein angemessener Weg sein, eine Programmiersprache oder eine Programmiertechnik zu erlernen (s. [SM12]). Das Potenzial erscheint nach unserer Ansicht nicht ausgeschöpft.

Mehr noch als bisher müssen die Tools als Komponenten in komplexen Lehrszenarien konfigurier- und einsetzbar sein. Möglich ist zum Beispiel die kombinierte Tool-Bereitstellung in einer PBL-Umgebung zum Messen und Simulieren, so dass die Ergebnisse des Programms oder der Ablauf des Programms mittels Visualisierungskomponenten (Software Visualization, Information & Scientific Visualization) in die Umgebung integriert dargestellt werden. Obwohl die Tools webbasiert sind, sind sie untereinander noch eher schlecht verzahnt. Ein erster Ansatz können Web Mashups sein, in denen die entsprechenden Tools über eine zentrale Seite gesteuert und in eingebetteten IFrames in HTML laufen.

Mit Tools fallen zum Teil erstmalig digitale Daten zum individuellen Lernprozess oder Prüfungsverlauf von Studierenden an. Somit sind rechtliche Fragen zu klären. So muss die Prüfungsordnung den Einsatz von technischen Hilfsmitteln in der Lehre zwar nicht explizit erlauben, dennoch kommt es in der Praxis zu Unsicherheiten, ob E-Prüfungen erlaubt sind oder in der Prüfungsordnung gesondert zugelassen sein müssen. Die Prüfungsakten müssen dokumentiert und darin zweifelsfrei der Verlauf der Prüfung einsehbar sein. Die Begründung der Bewer-

tung muss aus den Daten hergeleitet werden können. Die Aufbewahrung der entsprechenden Akten erfolgt mehrere Jahre lang. Es ist nicht ersichtlich, weswegen diese Daten nur analog und auf Papier erhoben und aufbewahrt werden können. Es ist möglich, dass hierzu entsprechende Tools erst nach einem Audit zertizifiert und zugelassen werden müssten. Eine entsprechende Zertifizierungsstelle gibt es unseres Wissens in Deutschland nicht.

25.5 Community

Tools können didaktisch sinnvoll in Lehr-/Lernszenarien der Programmierausbildung eingesetzt werden. Einige Lehrende scheuen allerdings aufgrund mangelnder positiver Erfahrung den Einsatz und verzichten auf den Erfolg.

Den ersten Schritt in Richtung automatisierter Programmbewertung machen Lehrende in der Regel, um Zeit bei der Auswertung von Übungsaufgaben zu sparen und mehr Zeit für den qualitativen Diskurs mit den Studierenden über Inhalt und Systematik, nicht Form(alien), zu gewinnen. Daraus erwächst der Wunsch, auf bestehendes Material und bestehendes Wissen effektiv zuzugreifen und es in eigenen Lehrveranstaltungen individualisiert einzusetzen. Daher sind in Zukunft Netzwerke von Expertinnen und Experten gefragt, in denen in Fachzirkeln zur Programmierausbildung aktuelle Fragestellungen bekannt gemacht und diskutiert werden können (vgl. [ABP13]).

Erst durch den Austausch von Tools und Content vermittels ihrer kompatiblen Schnittstellen, Austauschformate und Bedienungsstandards einerseits und einer vitalen offenen Community andererseits können „Economies of Scale“ erreicht werden. Nutznießer und Betroffene sind dabei zentral die Studierenden, deren Feedback, Meinungen und Wünsche man einholen und in diesen Prozess einfließen lassen muss.

25.6 Aufgabenpools und Qualitätssicherung

Tools können helfen, bisher genutzte Übungen und Assessments wiederzuverwenden und ähnlich einer Tauschbörse interessierten Lehrenden zur Verfügung zu stellen. In dem an der Hochschule Ostfalia eingesetzten System LON-CAPA verfolgen wir dieses Paradigma („Tausch statt Bau“), siehe [Kor+03]. Die wichtigsten Herausforderungen liegen dabei in der Qualitätssicherung von heterogen zusammengestellten Online-Aufgaben und -Ressourcen, die auch nach vielen Jahren noch problemlos funktionieren müssen. Ein weiteres Problem betrifft die Er-

reichbarkeit und Auffindbarkeit dieser Ressourcen auch durch Nutzer jenseits der Grenze des eigenen Systems, zum Beispiel in LON-CAPA ([KDP14]), aus beispielsweise Moodle und Stud.IP heraus. Dies kann durch ein gemeinsames oder durch mehrere vernetzte, verteilte Repositories erfolgen. Die Anforderungen an solche Repositories umfassen:

- Versionierung: Eingestelle Ressourcen müssen versioniert und historisiert werden.

- Lizenzmodelle: Die Ressourcen müssen beim Einstellen mindestens mit einer Lizenze annotiert werden, z. B. Creative Commons, Public Domain, BSD oder GPL.

- Export in andere Formate und Pools: Die Ressourcen müssen in verschiedene LMS oder Grader importiert werden können. Hierzu dient die Unterstützung des Austauschformats (vgl. Kapitel 24).

- Probezugänge und Vorschau: Eine begrenzte Untermenge der im Pool angelagerten Ressourcen sollte öffentlich zugänglich sein, ohne den Schutz der anderen Teile zu gefährden. Registrierten, autorisierten Nutzenden sollte es möglich sein, die Ressourcen vorab zu betrachten und testweise zu bedienen.

- Zugang/Suche: Es sollten die relevanten Metadatenformate unterstützt werden.

- Technik: Die Bereitstellung der Tools erfordert mehrere Server, die beispielsweise in einem Peer-to-Peer-System vernetzt sind. Zum Beispiel bietet LON-CAPA diese Funktionalität seit längerem.

- Datenschutz und Sicherheit: Sowohl die Nutzerdaten (Verhalten, Passworte) als auch die Ressourcen müssen durch hohe IT-Sicherheitsstandards nach z. B. den Empfehlungen des Bundesamtes für Sicherheit in der Informationstechnik (BSI) abgesichert sein.

- Host/Provider: Die Bereitstellung und Administration der Server erfordert finanzielle Zuwendungen, die bisher nicht von den Hochschulen eingeplant sind.

- Bewertung/Feedback: Ressourcen sollten bewertet und empfohlen werden können.

- Zugangskontrolle: Nur registrierten, nachgewiesenen Lehrenden ist der Zugang zu gestatten. Authentifizierung, Autorisierung und gegebenenfalls Audits im Rahmen der Datenschutzgesetze sind erforderlich.

- Mehrsprachigkeit: Die Ressourcen sollten auf Deutsch und in andere Sprachen übersetzt sein. Es sollte die Möglichkeit bestehen, dass Übersetzungen von anderen als den Autorinnen und Autoren der Ressourcen eingepflegt und qualitätsgesichert werden. Insbesondere die Qualität und Mehrsprachigkeit der Aufgabentexte erfordert mehrfache Peer Reviews, mitunter durch die Studierenden.

- Bepunktung: Die Art der Bepunktung sollte den Lehrenden, nicht den Autorinnen und Autoren, obliegen, d. h. im LMS vorgenommen werden. Dennoch kann eine Bepunktungsempfehlung Teil der Ressource sein.

25.7 E-Assessments und Feedback

E-Assessments stellen eine elektronisch unterstützte Erhebung der Lernleistung der Studierenden dar. Hierbei kommen wie auch im nichttechnischen Fall formative (begleitende) und summative (abschließende) Tests zum Einsatz. Den formativen Assessments liegt im Gegensatz zu den Prüfungen nicht das Erfordernis zugrunde, eine Wertung zu vergeben, sondern den Lernfortschritt abseits einer quantitativen Beurteilung zu kennen, um wiederum selbst oder gemeinsam daraus zu lernen.

25.7.1 Die rechtliche Seite

Die niedersächsische Lehrverpflichtungsordnung (LVVO) sieht streng betrachtet nur den Fall der Präsenzlehre, d. h., die Kontaktzeit mit den Studierenden im selben Raum vor. Genauso sieht sie streng betrachtet nur die Prüfung ohne Zuhilfenahme technischer Hilfsmittel vor. Diese Rahmenbedingungen sind nicht mehr zeitgemäß und verbauen vielfach den effektiven, angemessenen Einsatz von Tools in Lehre und Prüfungswesen. Im Projekt eCULT etablieren wir seit 2011 den Einsatz von Vorlesungsaufzeichnungen, Response-Systemen, computerbasierten Lernräumen und E-Assessments sowie E-Prüfungen. Es ist zu erwarten, dass mehr als bisher der Gesetzgeber und die Fakultäten klar benennen müssen, welcher Einsatz von Tools rechtlich abgesichert ist. Auch ist hierbei wichtig zu benennen, welche Funktionen in Tools ohne Einwilligung der Nutzer laut Datenschutzgesetz

verboten sind. Hierbei bewegen sich zurzeit Tool-Entwickler und -Nutzer in einer rechtlichen Grauzone, zum Beispiel wenn die Aktivitäten eines Studierenden bei der Bearbeitung protokolliert und an die Lehrperson zur, durchaus wohlmeinenden, Auswertung gesendet werden.

25.7.2 Feedback

Eine Hemmschwelle und Ablehnung erfahren Tools, wenn Studierende durch deren Einsatz frustriert werden und studentische Anforderungen nicht oder unzureichend in die Softwareentwicklung einfließen. Insbesondere soll die GUI einfach benutzbar, widerspruchs- und fehlerfrei sein. Der Content soll nach und nach an die Bedienung des Tools heranführen. Die Ergebnisse im Tool müssen zu den manuellen Bewertungen und sonstigen Prozessen in der Lehrveranstaltung passen. Zum Beispiel ist es problematisch, zueinander widersprüchliche Tools einzusetzen, die die Studierenden nicht einwandfrei nachvollziehen lassen, ob ihre Einsendungen/Antworten richtig oder falsch sind. Die Unflexibilität der Tools oder auch „verletzende Härte“ ihrer Bewertungsergebnisse (objektiv, maschinell überprüfbar) kann dazu führen, dass sich Studierende mit „halbrichtigen“ Lösungen zurückgesetzt und demotiviert fühlen. Hier ist auf die richtige Begleitung und Erläuterung zu achten und auch eine Einzelfallprüfung nicht abzulehnen. Kryptische Feedbacks führen dazu, dass die Bedienung der Tools als Eingabesysteme zu einem Trial-and-Error-System transmutiert, dem man nur durch den richtigen „Zauberspruch“ die Akzeptanz der eingereichten Lösung zu entlocken versucht. Eine Erläuterung der wichtigsten Feedbackmeldungen ist erforderlich, um dem vorzubeugen. Es muss mehrmalige Versuche geben und gleichzeitig eine zu schnelle Wiedereinreichung durch Verzögerungen verhindert werden. Zu Anfang sollten Fehlermeldungen bewusst gemeinsam provoziert werden, um die Reaktionen des Systems einschätzen zu können. Kritisch sehe ich Versuche, dem System natürlichsprachlich wirkende Feedbacks zu entlocken (die zudem didaktisch Sinn ergeben müssen). Wir sind mindestens ein Jahrzehnt von einer solchen Entwicklung entfernt, aktuelle Versuche mit Chatbots machen dies noch einmal deutlich (vgl. [Gra16]).

Ein Feedbackformat muss erweiterbar sein, von Prüfelementen („Checkern“) erzeugt und vom LMS weiterverarbeitet werden können. Es enthält mindestens folgende Felder:

- ID der einreichenden Person: Die Matrikel-Nr. und gegebenenfalls Institution muss bei der Einreichung gespeichert werden und darf durch die Clients nicht verfälscht werden.

- Zeitstempel: Die Serverzeit der Einreichung ist festzuhalten.
- Versuch: Der aktuelle Versuch der Einreichung muss gespeichert und hochgezählt werden. Auch hier darf die Nummer nicht durch den Client verändert werden können. Versuche, die durch klare Fehlbedienung des Clients entstanden sind, sollten nicht gezählt werden.
- Maximale Anzahl Versuche (Max. Versuch): Nach Überschreiten dieses durch die Lehrenden gesetzten Feldes (d. h. Anzahl Versuche > Max. Versuch) ist die Einreichung abzuweisen.
- Einreichung: Dieses Feld enthält alle zur Einreichung erforderlichen Daten.
- Pass/Fail: Der Grader erzeugt ein Flag, das vom LMS als Pass/Fail gespeichert wird. Eine erfolgreiche Einreichung verhindert die Eingabe weiterer Versuche.
- Punktzahl (Score): Der Score wird auf Basis des Rohergebnisses des Graders vom LMS ermittelt und angezeigt.
- Mindestpunktzahl (Min. Score): Dieser Score wird von den Lehrenden gesetzt. Es gilt, dass Pass genau dann wahr ist, wenn Score $\geq$ Min. Score gilt.
- Höchstpunktzahl (Max. Score): Der Max. Score dient den Studierenden als Richtschnur, wieweit ihre Lösung vom Optimum entfernt ist. Auch der Max. Score wird auf Basis von Empfehlungen, die mit der Aufgabenressource verbunden sind, im LMS gesetzt.
- Ausführungsreport (formatiert): Hier erhält jede Studentin und jeder Student ihren bzw. seinen Nachweis der Einreichung und nachgewiesenen Prüfung der Einreichung durch das System.
- Grund der Ablehnung (nur bei Fail): Es muss klar erkennbar sein und somit als Fehlermeldung oder Empfehlungstext vorliegen, an welcher Stelle bei der Prüfung der Einreichung welcher Fehler vorliegt.
- Lösungshinweise (Hints – nur wenn Score < Max. Score): Die Hints sind auf Basis von Heuristiken von den Autorinnen und Autoren gesetzte Hilfstexte, die erläutern, welche Verbesserungen bei der Einreichung möglich sind. Es sollten auch Hints ausgegeben werden können, wenn die Einreichung akzeptiert worden ist.

Außer dem Zeitgewinn durch den Einsatz erprobter Ressourcen und der Teil- oder Vollautomatisierung des Bewertungsvorgangs (was nicht die Notenvergabe einschließt, diese obliegt den Prüfenden nach Einsicht der Ergebnisse) lässt sich beobachten, dass Tools einen Qualitätsgewinn in der Lehre nach sich ziehen können, denn nunmehr erhält der oder die Lehrende zeitnah Ergebnisse aus den automatisch bewerteten Programmieraufgaben und kann daran seine Lehre ausrichten, zum Beispiel Lehrstoff wiederholen oder Teile überspringen. Studierende erhalten schneller als bisher eine umfangreiche Auswertung ihrer Leistung und Hinweise zur Verbesserung. Die Optimierungen der formativen Feedbacks aus didaktischer Sicht und der datenschutzrechtlich abgesicherten und sabotage- und spionagegeschützten summativen Feedbacks (E-Prüfungen) sind nach unserem Ermessen ein offenes Forschungsgebiet.

Als weitere Forschungsperspektive können adaptive Feedbacks gelten, in denen das System intelligent entscheidet, in welcher Form es Feedback und Leseempfehlungen vergibt. Diese als Intelligent Tutoring Systems (ITS) bekannten Systeme sind kommerziell erhältlich. Sie fristen aus unserer Sicht eher ein Nischendasein, weil ihr Einsatz kostenpflichtig ist, Datenschutzbedenken bei Cloud-basierten ITS existieren und ITS das „klassische“ Zusammenspiel zwischen Lehrenden und Studierenden neu definieren, siehe [Bak16].

25.8 Plagiarismus

Man unterscheidet hier Plagiarismus in mindestens den folgenden Ausprägungen:

1. Plagiate von außerhalb des Studierendenkreises,
2. Plagiate einer anderen Lösung desselben Semesters,
3. Plagiate einer anderen Lösung aus vorhergehenden Semestern.

Plagiate umfassen neben wortwörtlichen Übernahmen auch strukturelle Plagiate, eine Übersicht und Diskussion geben [WWW06]. Dem weiteren Problem des Ghostwritings kann nur durch prüfungs- oder prüfungsähnliche Situationen entgegen gewirkt werden. Dem ersten Punkt kann durch geeignete Konzeption neuer Aufgaben und Ressourcen Abhilfe geschaffen werden. Auch kann unter Berücksichtigung der Datenschutz- und Urheberrechtsgesetzgebung Plagiatserkennungssoftware eingesetzt werden. Rechtlich gangbar ist allem Anschein nach nur eine Ähnlichkeitsprüfung zwischen studentischen Einreichungen an genau derselben Hochschule. Sie sollte hinreichend sein, da auch bei Plagiaten von anderen Hochschulen die Anzahl der „Incidents“ an derselben Hochschule hoch und damit

erkennbar sein müsste. Doch diese Prüfungen können erst im Nachhinein eingesetzt werden. Wünschenswerter sind konstruktive Verfahren, die ein Plagiat von vornherein hinreichend erschweren.

Dem Plagiarismus nach Punkt zwei entgegenwirken kann zum einen eine gemeinsame Eingabefrist für alle. Die Abgabe einer fremden Lösung wäre einerseits aufgrund des hohen Grads der Entdeckungswahrscheinlichkeit risikoreich, und zum anderen wäre sie keine Garantie auf ein korrektes Lösen der Aufgabe, da alle zum selben Zeitpunkt ihr Feedback erhielten. Zum anderen kann die Randomisierung von Aufgaben oder deren ständiger Austausch Abhilfe schaffen. Sie ist auch bei Punkt drei anwendbar.

Die Randomisierung von Programmieraufgaben ist diffizil. Einen möglichen Ansatz stellt die Parametrierung dar. Eine Ausprägung der Parametrierung hinterlegt bestimmte Felder mit variablen Ausdrücken, zum Beispiel „Schreiben Sie ein Programm, das zwei Zahlen des Datentyps <DATENTYP> <multipliziert | dividiert>“. Nehmen wir an, dass als Datentyp Float, Double und Integer hinterlegt sind, so ergeben sich 6 mögliche Ausprägungen. Entweder muss für jede dieser Ausprägungen eine andere Musterlösung hinterlegt werden, oder es existiert dieselbe Lösung, die mittels eines Präprozessors in die jeweilige Ausprägung gebracht und dann verwendet wird. Die Schwierigkeit, solche Varianten zu debuggen, ist erheblich. Es wäre hilfreich, eine Form der Randomisierung zu entwickeln, deren „Komplexität“ konstant oder logarithmisch mit der gewünschten Anzahl der Möglichkeiten wüchse. Eine andere Herausforderung ergibt sich aus der Wahrscheinlichkeit, dass mit der Zeit eine Sammlung an studentischen Lösungen entsteht, die außerhalb des Systems ebenfalls in einer Art Tauschbörse genutzt werden könnte.

Ein probabilistischer Ansatz dürfte ein durch Learning Analytics/KI ermitteltes Wahrscheinlichkeitsmodell sein, das die kognitiven Fortschritte jeder Studentin und jedes Studenten verzeichnen würde. Das Modell könnte nicht nur automatisch darauf hinweisen, wie wahrscheinlich es ist, dass die Einreichung von der Studentin oder dem Studenten selbst stammt, also zur Plagiatserkennung herangezogen werden. Es könnte darüber hinaus didaktisch sinnvoll dazu einsetzbar sein, Defizite und daraus abgeleitet erforderliche Lernwege zu prognostizieren. Hier schließt sich wieder der Kreis zu den ITS und zur eingangs zitierten APOS-Theorie, die eine Grundlage für ein Modell sein könnte. Auch andere technische Modelle, zum Beispiel künstliche neuronale Netze (KNN), können verwendet werden.

25.9 Fazit

Die Entwicklung von Tools in der akademischen Programmierausbildung hat Fortschritte gemacht und mehr Verbreitung an der Hochschule gefunden. Dennoch stehen wir erst am Anfang. Es ist zu hoffen, dass die Resultate verstetigt werden können und in Zukunft auf breite Akzeptanz bei Lehrenden und Studierenden treffen werden.

Literatur für dieses Kapitel

[ABP13] *Workshop „Automatische Bewertung von Programmieraufgaben“ (ABP 2013)*. Bd. 1067. CEUR Workshop Proceedings. (http://ceur-ws.org/Vol-1067/). 2013.

[Bak16] Ryan S. Baker. „Stupid Tutoring Systems, Intelligent Humans“. In: *International Journal of Artificial Intelligence in Education* 26.2 (2016), S. 600–614. DOI: 10.1007/s40593-016-0105-0.

[Clo13] Doug Clow. „An overview of learning analytics“. In: *Teaching in Higher Education* 18.6 (Aug. 2013), S. 683–695.

[DM02] Ed Dubinsky und Michael A. McDonald. „The Teaching and Learning of Mathematics at University Level: An ICMI Study“. In: Hrsg. von Derek Holton u. a. Dordrecht: Springer Netherlands, 2002. Kap. APOS: A Constructivist Theory of Learning in Undergraduate Mathematics Education Research, S. 275–282. DOI: 10.1007/0-306-47231-7_25.

[Gra16] Bernd Graff. „Rassistischer Chat-Roboter: Mit falschen Werten bombardiert“. In: *Süddeutsche Zeitung* (2016). URL: http://www.sueddeutsche.de/digital/microsoft-pro\-gramm-tay-rassistischer-chat-roboter-mit-falschen-werten-bombardiert-1.2928421 (besucht am 13. 05. 2016).

[Guz08] Mark Guzdial. „Education: Paving the Way for Computational Thinking“. In: *Commun. ACM* 51.8 (Aug. 2008), S. 25–27. DOI: 10.1145/1378704.1378713.

[KDP14] Gerd Kortemeyer, Stefan Dröschler und David E. Pritchard. „Harvesting latent and usage-based metadata in a course management system to enrich the underlying educational digital library – A case

study“. In: *Int. J. on Digital Libraries* 14.1-2 (2014), S. 1–15. DOI: 10.1007/s00799-013-0107-6.

[Kor+03] Gerd Kortemeyer u. a. „The Learning Online Network with Computer-Assisted Personalized Approach (LON-CAPA)“. In: *PGLDB 2003, Proceedings of the I PGL Database Research Conference, Rio de Janeiro, Brazil, April 10-11, 2003*. Hrsg. von Rubens N. Melo, Diva de Souza e Silva und Sean W. M. Siqueira. Bd. 70. CEUR Workshop Proceedings. CEUR-WS.org, 2003.

[MP04] Joan Middendorf und David Pace. „Decoding the disciplines: A model for helping students learn disciplinary ways of thinking“. In: *New directions for teaching and learning* 2004.98 (2004), S. 1–12.

[SM12] Harald Søndergaard und Raoul A. Mulder. „Collaborative learning through formative peer review: pedagogy, programs and potential“. In: *Computer Science Education* 22.4 (2012), S. 343–367. DOI: 10.1080/08993408.2012.728041.

[WWW06] Debora Weber-Wullf und Gabriele Wohnsdorf. „Strategien der Plagiatsbekämpfung“. In: *Information: Wissenschaft & Praxis* 57.2 (2006), S. 90–98.