Projekttitel: **Learning by Coding** - Programmierkompetenz nachhaltig und fachübergreifend fördern

Tandembeitrag / Antragstellende:
Prof. Dr. Sebastian Speiser (Informatik) und Johanna Sedlmair vom SkiLL (Servicezentrum für kompetenzorientiertes & innovatives Lernen & Lehren) der HFT Stuttgart. In Kooperation mit Prof. Dr. Melanie Baur (Wirtschaftsinformatik)

Inhalt

1 Persönliche Motivation / Ausgangspunkt

Ausgangssituation

Studierende der Informatik und Wirtschaftsinformatik benötigen solide Kenntnisse im Programmieren als Grundlage für ihr weiteres Studium und für ihr späteres Berufsleben. Die Vorlesung zur Einführung in die Programmierung beinhaltet hochkomplexe Themenstellungen, die für viele Studierende eine ernstzunehmende Hürde im Studium darstellen. Die Folge sind hohe Durchfallquoten. Diese liegt im Bachelor Informatik bei ca. 1/3, im Bachelor Wirtschaftsinformatik sogar bei bis zu 2/3. Somit ergibt sich ein hohes Frustrationspotential begleitet von fachlichen Schwächen, die sich durch mehrere Semester ziehen können. Ein nicht unerheblicher Teil der Studierenden bricht das Studium sogar ab. Darüber hinaus gewinnen technologische Kompetenzen, wie Softwareentwicklung und speziell das Programmieren als Schlüsselfaktoren für die erfolgreiche Bewältigung der digitalen Transformation an

Bedeutung, zunehmend auch in nicht-IT-basierten Studiengängen[1]. Die Nachfrage für IT- und Programmierkompetenzen steigt somit nicht nur in den IT-affinen Studiengängen, sondern beispielsweise auch für Studiengänge wie Bauphysik, Bauingenieurwesen und Architektur. Hier gehen wir von einem noch höheren Unterstützungsbedarf aus.

Klassischerweise werden diese Programmiermodule zweigeteilt angeboten:

- 90 Minuten Frontalvorlesung im Vorlesungsraum mit PowerPoint-Folien zu Konzepten und Beispielen
- 180 Minuten Übung im PC-Raum mit einem Übungsblatt und Betreuung durch Lehrende

Zwischen Frontalvorlesung und Übung müssen die Studierenden in der Regel die Vorlesung nachbereiten, z.B. indem die gezeigten Codebeispiele von den Folien in eine Programmierumgebung kopiert und ausgeführt werden. Dabei kann die Zeit zwischen Vorlesung und Übung aufgrund von externen Faktoren (Verfügbarkeiten von Räumen, Lehrenden und Studierende) sehr knapp sein. Aktuell findet so beispielsweise im WS22/23 die Vorlesung am Dienstag bis 17:15 Uhr statt, die Übung startet aber bereits am Mittwoch um 08:00 Uhr. Die Studierenden haben somit nur sehr begrenzt Zeit die Inhalte der Vorlesung zu wiederholen und zu verstehen. Eine Vorbereitung auf die Übungsaufgaben ist damit kaum machbar.

Didaktische Herausforderungen

Die Komplexität des Themas „Programmieren“ in der Vorlesung stellt Lehrende vor große Herausforderungen:

Bildungsanliegen: Fehlende (digitale) Kompetenzen

Das Aufsetzen der Programmierumgebung (erstellen des Projekts, anlegen einer Klasse etc.) bevor der eigentliche Inhalt überhaupt geübt werden kann, kostet die Studierenden viel Zeit, die ihnen dann für die Übung und evtl. das sinnvolle Wiederholen fehlt. Erschwert wird dies zusätzlich durch die zunehmende Diversität der genutzten Plattformen (Windows, MacOS, Linux, Android, iPadOS, ...).

Digital Natives: Die Studierenden sind mit digitaler Technologie aufgewachsen und gewohnt, dass alles mit nur wenigen Klicks funktioniert (z.B. App downloaden,

[1] vgl, Stifterverband, Für morgen befähigen: Hochschul-Bildungs-Report 2020. Jahresbericht 2019. [Online]. Available: https://www.stifterverband.org/download/file/fid/7803

installieren und direkt öffnen). Die Frustrationstoleranz ist teilweise sehr gering ausgeprägt und das benötigte Durchhaltevermögen zum Lösen der Aufgaben fehlt.

Lehrarrangement: Kein Wechsel des Lernmodus

Eine starre Aufteilung in Vorlesungsblock, in dem frontal umfangreiche Inhalte präsentiert werden, und ein separater Übungsblock, der erst später den vermittelten Stoff aufgreift, führt zu einer mangelnden Rhythmisierung des Lernstoffs: Die Studierenden verlieren während des 90-minütigen Vorlesungsblocks den roten Faden und finden keinen Anschluss mehr, da die Dauer der Aufmerksamkeitsspanne der meisten deutlich unter 90 Minuten liegt. Somit müssen die Inhalte der Vorlesung in der Übung dann in den Skripten gesucht werden, bevor diese angewendet und eingeübt werden können. Weiterhin führt der lange 180-minütige Übungsblock mit wenig Strukturvorgaben zu einer permanenten Überforderung schwächerer Studierender – im Einzelfall auch zu Langeweile bei sehr starken Studierenden.

Rahmenbedingungen: Fest installierte Programmierumgebung

Aufgrund der fest installierten Programmierumgebung in den PC-Hörsälen besteht nur eine geringe Möglichkeit Praxisteile zu Hause zu wiederholen und zu üben. Die Gruppenarbeit und Hilfestellungen durch Lehrende sind durch eine fixe und enge Aufteilung beschränkt. Hybride Formen online und offline sind schwer, bzw. mit momentaner technischer Ausrüstung in den PC-Pools nicht zu verbinden.

Die Zusatzarbeit für eine Programmierübung ist hoch: Es müssen die PCs hochgefahren werden, Einloggen, Starten der Programmierumgebung, Aufsetzen des Projekts, Anlegen oder Kopieren eines Grundgerüsts etc. Dies lohnt sich nur bei größeren Übungsaufgaben oder Übungsblöcken.

Lehrorganisation: Entwicklung und Verwaltung von Übungsaufgaben

Übungsaufgaben werden bisher händisch ohne Toolunterstützung verwaltet. Übungsblätter im PDF-Format und Codegerüste als ZIP-Dateien werden in Moodle abgelegt. Eine kontinuierliche Sammlung, Weiterentwicklung und Pflege von Übungsaufgaben durch ein Team von Lehrenden sind somit nur begrenzt möglich.

Best-Practices zur Code-Verwaltung aus dem Software-Engineering sind vielversprechend, müssen aber angepasst werden, u.a. weil Übungen kleinteiliger sind als produktive Programme und eine Unterteilung zwischen Aufgabenstellung, Grundgerüst und Musterlösung notwendig ist.

2 Angestrebte Ziele

Die oben beschriebenen Herausforderungen haben deutlich gemacht, dass im Modul Programmieren ein neues Konzept implementiert werden sollte, für das die folgenden Ziele handlungsleitend sind:

Kompetenzorientierung: Eine browserbasierte Anwendung zum Erwerb von Programmierkompetenz soll sukzessive auf- und ausgebaut werden. Der Zugriff per Browser ist leichtgewichtig und von jedem Gerät aus möglich. Die Anwendung enthält die vorlesungsbegleitende Dokumentation, Code und Übungsaufgaben in interaktiven Dokumenten. Das Dokument wird durch die Studierenden während der Vorlesung und in der Nachbereitung kontinuierlich ergänzt und Lücken werden gefüllt. Auf diese Weise werden die geforderten theoretischen Wissensbereiche mit praktischer Programmierkompetenz verzahnt und die Lernenden bauen schrittweise ihre Wissensstruktur und Kompetenzen auf und aus. Die Dokumente sind stets synchron – egal ob von zu Hause, unterwegs oder an der Hochschule darauf zugegriffen wird.

Verknüpftes, erweiterbares Wissen: Die Anwendung ist so aufgebaut, dass die Themenbereiche nicht abschließend behandelt, sondern im Verlauf der nachfolgenden Übungen immer wieder miteinander verknüpft werden. Die wiederholendenden Elemente ermöglichen dabei die stoffliche Festigung und Vertiefung.

Problemlösendes Lernen: Durch die Integration der theoretischen Programmierkonzepte der Vorlesung in einen praktischen Anwendungskontext, werden konstruktive Lernprozesse angeregt und der Wissenstransfer schrittweise gefördert. Die Plattform unterstützt das Anknüpfen an das persönliche Vorwissen und fördert selbstgesteuertes Lernen im eigenen Tempo.

Programmierkompetenz als „Future Skill“: Da Programmierkompetenzen auch über informationstechnische Studiengänge hinaus an Bedeutung gewinnen, steht die Anwendung als Zusatzangebot im Rahmen des Studium Integrale zur Verfügung. Beim Studium Integrale handelt es sich um ein extracurriculares Studienangebot, dass allen Studierenden der HFT Stuttgart zur Verfügung steht. In Zukunft sehen wir eine Möglichkeit zur Integration in weitere Studiengänge.

Datenbasiertes Feedback für Studierende und Lehrende: Die browserbasierte Lernumgebung erfasst, welche Themen und Aufgabenstellungen von den Studierenden fehlerfrei bearbeitet werden und an welchen Stellen die Fehlerhäufigkeit steigt. Inhalte können auf diese Weise gezielt in der Vorlesung wiederholt oder als

gewusst vorausgesetzt werden. Darüber hinaus können Studierende basierend auf ihren Leistungen innerhalb der Anwendung gezielt Rückmeldungen von der Lehrperson erhalten.

Geplantes Konzept „Programmieren – Reloaded“

Stufe 1:

Entwicklung und Pilotierung in den IT-Studiengängen Bachelor Informatik und Bachelor Wirtschaftsinformatik:

Interaktive Veranstaltungen mit häufigem Wechsel des Modus zwischen "Konzeptvorstellung" und "selbst ausprobieren". Hierbei steht das selbst Programmieren und Ausprobieren im Vordergrund von eben erlernten kleinen Theorieeinheiten. Code-Beispiele auf Folie sollen durch direkte Praxis ersetzt werden. Durch diese kleineren Übungseinheiten mit schnellem Feedback können die Studierenden unmittelbar Wissen anwenden und ihre Wissenslücken schließen.

Die Zusammenarbeit soll gefördert werden, in dem nicht mehr in fixen PC-Räumen unterrichtet wird, sondern in Räumen mit Gruppentischen. Die Studierenden können mit dem neuen Konzept mit ihrer eigenen Hardware arbeiten, ein Tablet reicht hierfür bereits aus. Durch das Zusammensitzen in Kleingruppen, können verschiedene Lösungsansätze diskutiert und direkt erprobt werden. Durch das Arbeiten mit eigener Hardware entfällt auch die Abhängigkeit von PC-Räumen: Diese haben nur eine fest installierte Anzahl an PCs und sind zeitlich begrenzt verfügbar. Auch können nur so viele Teilnehmende im Kurs mitarbeiten, wie PCs zur Verfügung stehen. Eine digitale Teilnahme wird damit automatisch möglich – wir sehen hier aber weniger den Regelfall, als mehr eine Möglichkeit bei Krankheit oder ähnlichen Umständen weiterhin teilhaben zu können und vor allem virtuelle Lerngruppen zu bilden, auch wenn Studierende weit auseinander wohnen. Konzepterklärungen, ausführbare Codebeispiele und in Lücken eingefügte eigene Programmierübungen sollen als integrierte Dokumente erstellt werden, die als Lernportfolio und Nachschlagewerk bzw. Skript dienen können.

Stufe 2:

Das in Stufe 1 entwickelte Konzept soll in Stufe 2 im Studium Integrale für alle Studierende der Hochschule angeboten werden und als OER / ZOER Plattform bereitgestellt werden. Weiterhin kann das Konzept bei entsprechendem Bedarf gezielt

für Studierende anderer Fachrichtungen angepasst und in deren Curriculum angeboten werden. Hierbei sehen wir folgende Hauptaufgaben:

Für das Studium Integrale ist Erstellung von Kursinhalten mit Fokus auf fachübergreifenden Anwendungsbeispielen essenziell. Technische Implementierungsdetails sollen hierbei in den Hintergrund rücken. Das primäre Ziel ist die Vermittlung von allgemeinen Konzepten des Programmierens und algorithmischen Denkweisen erlernen. Diese können die Teilnehmenden dann in ihrem fachspezifischen Kontext anwenden.

Bei entsprechender Nachfrage kann bzw. soll das Konzept zur Unterstützung weiterer Programmiersprachen und anderer Schwerpunkte erweitert werden.

Da die Inhalte nun an eine diverse Studierendenschaft vermittelt werden sollen, müssen die Plattform und die Verwaltungsprozesse entsprechend skalierbar sein. Auch werden verschiedene Lehrpersonen die Veranstaltungen für das Studium Integrale durchführen, so dass die Plattform zum einen nutzerfreundlich aber auch einfach erweiterbar aufgebaut sein muss.

Konzeptionelle Vorarbeiten

Seit Sommersemester 2022 werden einzelne Ideen bereits in verschiedenen Vorlesungen im 1. und 2. Semester der Bachelorstudiengänge Informatik und Wirtschaftsinformatik prototypisch erprobt.

Lehrarrangement:

Das Modul Programmieren findet im jetzigen Wintersemester in einem der beiden Bachelorstudiengänge im 1. Semester erstmalig ohne Frontalvorlesung in einem Hörsaal statt. Stattdessen werden kurze Inputs gefolgt von einer direkten Übungseinheit. Nach vier Wochen Durchführung ist das Feedback der Teilnehmenden zum häufigen Wechsel des Modus bisher ausschließlich sehr gut. Ein erster Differenzierungstest zwischen verschiedenen Programmierkursen und im Vergleich zu vergangenen Semestern findet in Kürze statt.

Rahmenbedingungen:

Proof-of-Concept Einsatz einer browserbasierte Programmierumgebung für eine Vorlesung, die dank Infrastructure-as-Code Vorlagen, nachvollziehbar und wiederholbar für weitere Veranstaltungen aufgebaut werden kann. Erste Erweiterungen von Jupyter Notebooks für die Programmiersprache Java sind entstanden: Die Jupyter Notebooks dienen als Format für interaktive Dokumente, die

Text, Graphiken und Programmiercode integrieren. Hierbei wurden bestehende Open Source Tools angepasst sowie eigene Bibliotheken entwickelt. Eine Vorlesung im 2. Semester wird nun in diesem Wintersemester erstmalig mit Jupyter Notebooks begleitet.

Lehrorganisation:

Seit letztem Sommersemester findet sich GitLab-Repository für Programmieraufgaben inkl. Prozesse, Automationstools und Dokumentation im Aufbau. So wurden im letzten Semester alle Programmieraufgaben inkl. Musterlösungen, welche im 1. Semester in beiden Bachelor-Studiengängen zum Einsatz kamen, in das Repository geladen.

Ein erster Ansatz im jetzigen Wintersemester zeigt die Schwierigkeiten: Die Skalierung von zwei auf vier Lehrveranstaltungen und somit von zwei auf vier Lehrpersonen, erschwert die bisher implementierte Struktur beizubehalten. Positiv lässt sich aber sagen, dass die Bibliothek der Aufgaben stetig wächst und die Aufgaben des letzten Semesters einfach und mi sehr geringem Aufwand wiedergenutzt werden können.

Die Übersicht in Moodle wird deutlich erhöht, da hier keine Übungsblätter mehr abgelegt werden und auch keine verschiedenen Versionen von Übungen mehr verwaltet werden müssen.

Nächste Schritte

Nachdem erste Vorarbeiten bereits getätigt wurden und einzelne Konzepte in verschiedenen Lehrveranstaltungen in Erprobung sind, sind folgende weitere Arbeiten als nächstes zu tun:

Rahmenbedingungen:

Skalierung auf mehrere Lehrveranstaltungen was folglich mehrere Lehrende miteinbezieht. Die browserbasierte Programmierumgebung ist bisher nur in einer Lehrveranstaltung von einer Lehrperson im Einsatz. Dies soll sowohl technisch als auch organisatorisch in einem strukturierten Vorgehen ausgeweitet werden:

Technisch: Die Verwaltungstools müssen mit bindenden Prozessen versehen werden. Dazu ist eine klare und verbindliche Dokumentation notwendig. Damit dies gelingen kann, sind organisatorische Maßnahmen zu treffen.

Organisatorisch: Schulungen für die anderen Lehrenden sind anzubieten, so dass diese sich in den neuen Tools auskennen und die aufgesetzten Prozesse einhalten.

Lehrorganisation:

Die Theorieinhalte müssen in kleinere Inputs unterteilt werden. Hierfür müssen die Lehrmaterialien in kleinere Abschnitte eingeteilt werden, welche von Praxiseinheiten gefolgt werden. Der Schnitt der Folienpräsentationen soll auf kleineren Lernetappen basieren. Zusätzlich müssen die Programmierbeispiele von Folienbeispielen in Notebook/Programmierbeispiele übersetzt werden. Das Strukturierungssystem für die Aufgaben muss erweitert, ggf. auch neu überdacht werden, da sich im ersten Skalierungsschritt bereits Inkonsistenzen gebildet haben. Die Schulung der anderen Lehrpersonen ist auch hierfür notwendig, damit diese in der Lage sind, Aufgaben im Repository hinzuzufügen. Bereits die Schulung nur einer weiteren Lehrperson hat dieses Semester mehrere Stunden in Anspruch und trotzdem sind bereits erste Inkonsistenzen im Repository zu erkennen. Wenn sich herausstellen sollte, dass es nicht möglich ist, die bestehende Struktur trotz Schulungsmaßnahmen beizubehalten, muss ggf. ein neues Tool gefunden und aufgesetzt werden, um Aufgaben und Musterlösungen synchron zu halten – über mehrere Lehrpersonen inkl. akademische Mitarbeiter hinweg. Die Erhöhung der Anzahl der Aufgaben ist ein wichtiger Baustein, so dass die Studierenden die Möglichkeit haben, ihre Programmierkenntnisse immer weiter auszubauen. Hierfür sollen die Aufgaben in Schwierigkeitsgrade eingeteilt werden, so dass auf individuellem Niveau geübt werden kann. Tests, die den Lernfortschritt kontrollieren, sollen digital gestützt erfolgen und nach klaren Kriterien Punkte vergeben. Diese erden sich am individuellen Lernstand orientieren.

3 Geplante Kooperation

Da die browserbasierte Programmier- und Dokumentationsplattform sowohl in ausgewählten Studiengängen als auch hochschulweit eingesetzt werden soll, findet eine Kooperation zwischen Prof. Dr. Sebastian Speiser (Informatik) sowie Johanna Sedlmair als Vertreterin des Servicezentrums für kompetenzorientiertes und innovatives Lernen und Lehren (SkiLL) statt. Weiterhin ist eine Mitarbeit von Prof. Dr. Melanie Baur (Wirtschaftsinformatik) vorgesehen.

4 Curriculare Integration

Die IT-Anwendung wird in den beiden Studiengängen BA Informatik und BA Wirtschaftsinformatik in den Lehrveranstaltungen „Programmieren 1“ und

„Programmieren 2“, welche Pflichtveranstaltungen im 1. und 2. Semester sind, pilotiert. Weiterhin findet sie im 2. Semester des BA Informatik in der Pflichtveranstaltung „Datenstrukturen und Algorithmen“ Anwendung.

Darüber hinaus sollen vorerst zwei Module in das Studium Integrale der HFT Stuttgart integriert werden. Hierbei handelt es sich um ein extracurriculares Angebot, das von allen Studierenden der Hochschule als profilgebender Zusatz absolviert werden kann.

Modul 1 – Grundlagen:

Das erste Modul beschäftigt sich mit Grundkonzepten des Programmierens und schult die Teilnehmenden in ersten algorithmischen Denkweisen. Es ist für Studierende ohne Vorkenntnisse aller Studiengänge geeignet. Hierbei steht nicht das Erlernen einer bestimmten Programmiersprache im Vordergrund, sondern die Herangehensweise an die Denkweise des Codings und die Befehlsstruktur von Maschinen, wie sie z.B. unter zu Hilfenahme von Robotern (z.B. LEGO Education – Mindstorms oder ähnlichen Modellen) darstellen, geschult werden kann. Roboter zählen als sehr motivierendes Unterrichtswerkzeug, womit spielerisch Kenntnisse im Programmieren erworben werden. Auch erfolgt durch die Programmierung dieser ein direktes Feedback, ob der eingegebene Code richtig war oder nicht, was die Problemlösungsfähigkeit schult.

Modul 2 - Vertiefung:

Das zweite Modul ist als Vertiefungsmodul gedacht, welches eine gängige Programmiersprache (z.B. Java oder Python) einführt, führende Programmierparadigmen behandelt und in einem eigen zu erstellenden Programmierprojekt anwendet. Nach Abschluss des zweiten Moduls haben die Studierenden Grundkenntnisse im Programmieren erlernt.

Bei Belegung des Profilfachs Informatik in der Schule oder bei vorhandenen sonstigen Vorkenntnissen, kann auch direkt das zweite Modul absolviert werden.

Erweiterungen: Bei entsprechender Nachfrage kann das Angebot langfristig mit einem dritten Modul erweitert werden, welches weitere Programmiersprachen (wie z.B. JavaScript, TypeScript) vermittelt und mit dem Erstellen einer Webapplikation abschließt.

5 Evaluationskonzept

Das Konzept wird aus Lehrenden- und Lernendenperspektive evaluiert:

Lehrpersonen evaluieren die Module auf Basis folgender Kriterien per Fragebogen oder mit halbstandardisierten Interviews: Eignung der Lernmaterialien, Eignung für

datenbasierte Feedbackprozesse, Fortschritt und Lernprozess der Studierenden, Vergleich der Prüfungsergebnisse (mit Vorjahren) oder mit Parallelkursen.

Studierende evaluieren die Lernmodule auf Basis folgender Kriterien per Fragebogen oder mit halbstandardisierten Interviews: Eignung der Lernmaterialien, subjektive Einschätzung des Fortschritts und Lernprozesses, subjektive Einschätzung der Schwierigkeit einzelner Programmierkonzepte / Programmierschwierigkeit, Fortschritt und Lernprozess anhand der erzielten Ergebnisse auf der Plattform. Die Evaluationsergebnisse fließen in die Optimierung der Lehrveranstaltung ein. Dabei können die Evaluationsergebnisse der curricularen und der extracurricularen Teilnahme miteinander vergleichen werden, auch hinsichtlich typischer Lernmuster und Problemstellungen.

6 Nachhaltige Verankerung und Transfer

Einfache Tools mit Fokus auf Reproduzierbarkeit und Wartbarkeit sollen die Weiterverwendung nicht nur an der eigenen Hochschule, sondern darüber hinaus auch an anderen Hochschulen ermöglichen. Bisher ist dies nicht gegeben, da die Herangehensweise zu sehr auf die eigenen Bedürfnisse zugeschnitten ist (Robustheit über Komplexität).

Die Innovation kann prinzipiell auf alle einführenden Programmierveranstaltungen übertragen werden, die typischerweise in mathematischen und technischen Studiengängen stattfinden. Der fachübergreifende Ansatz soll sogar die Verwendungen für alle Studiengänge zum Ziel haben. Aus diesem Grund eignet sich die browserbasierte Umgebung sowie die entsprechenden Aufgabenstellungen zur Bereitstellung als Open Educational Resources über das ZOERR Repository BW.

Durch die fachübergreifende Zusammenarbeit mit dem SkiLL (Servicezentrum für kompetenzorientiertes & innovatives Lernen & Lehren) wird zunächst die nachhaltige Nutzung in allen Studiengängen der HFT Stuttgart sichergestellt. Hiermit wird der steigenden Bedeutung von Programmieren nicht nur für die Fachgebiete der Informatik und Wirtschaftsinformatik, sondern auch darüber hinaus Rechnung getragen.