Tandem-Fellowship: **T**ransfer einer **i**nnovativen **P**rogrammierausbildung mit **Min**i-Programmier-Welten (TiPMin)

Antragsteller: Universität Stuttgart, Institute of Software Engineering
Tandem-Partner: Karlsruher Institut für Technologie, Institut für Informationssicherheit und Verlässlichkeit
Lehrperson Universität Stuttgart: Prof. Dr.-Ing. Steffen Becker
Lehrperson Karlsruher Institut für Technologie: Prof. Dr.-Ing. Anne Koziolek

Motivation

Die Entwicklung von komplexer Software und insbesondere das Programmieren wird heutzutage zunehmend wichtiger und etabliert sich mehr und mehr in interdisziplinären Bereichen. Daher bieten wir an der Universität Stuttgart, als auch unser Tandem-Partner am Karlsruher Institut für Technik (KIT) entsprechende Grundlagenvorlesungen an. An beiden Universitäten zeigt sich, dass die Grundlagenvorlesungen zur Softwareentwicklung in einer zunehmenden Anzahl an fachfremden Studiengängen integriert sind. Aufgrund unterschiedlicher Vorkenntnisse und Affinitäten für das Programmieren, müssen Vorlesungen und Übungen möglichst greifbar und verständlich gestaltet werden. Nicht zuletzt ist es notwendig, eine nachhaltige und möglichst intrinsische Motivation für die Softwareentwicklung zu wecken und Studierende zu animieren, sich intensiv mit der Thematik vertraut zu machen. Hierzu finden Ansätze, wie beispielsweise Gamification, digitale Hilfsmittel wie spezielle Online-Entwicklungsumgebungen oder automatische Programmbewertungen, zunehmend Verwendung.

Dies führt zu unserer Bewerbung um Fördermittel für ein Tandem-Fellowship Projekt. Das Ziel des Projektes ist es mögliche Synergien für die Lehre im Kontext der Programmierung und Softwareentwicklung zwischen der Universität Stuttgart und dem KIT zu nutzen, um einen Wissenstransfer und einen Austausch rund um die Lehre der Softwareentwicklung durchzuführen. Insbesondere haben wir an der Universität Stuttgart in den letzten Jahren innovative Ideen erprobt und möchten unsere positiven Erfahrungen transferieren.

Aktueller Stand der Lehre

In der Lehre für die Softwareentwicklung werden traditionell Bottom-Up Ansätze angewendet, welche typischerweise zunächst konkrete Konzepte wie die prozedurale Programmierung behandeln, während abstraktere Inhalte wie Objekte oder Komponenten oftmals erst im späteren Verlauf thematisiert werden. Jedoch erfordert die Praxis üblicherweise ein tiefes Verständnis der abstrakteren Konzepte, welches in den traditionellen Lehrveranstaltungen durch den Bottom-Up-Ansatz nicht ausreichend vermittelt wird. Dieses Problem kann durch einen Outside-In Ansatz angegangen werden, welcher den Fokus zuerst auf die abstrakteren, praxisrelevanten Konzepte setzt und damit die erforderliche Denkweise der Studierenden besser fördert. Tabelle 1 zeigt einen Überblick über die Grundlagenvorlesungen an der Universität Stuttgart und dem KIT und ordnet den Ansatz in Outside-In und traditionell ein.

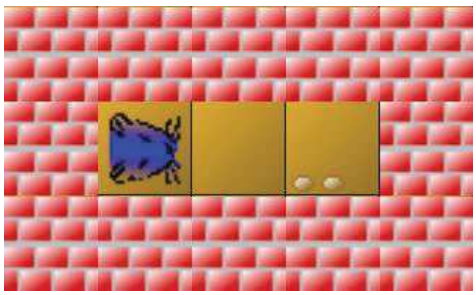

Abbildung 1: Hamster Simulator

Tabelle 1: Grundlagenvorlesungen der Universität Stuttgart und dem KIT

Universität	**Grundlagenvorlesung**	**Art**	**Stufe**	**Ansatz**
Universität Stuttgart	„Programmieren und Softwareentwicklung“	Pflichtmodul	Bachelor	Outside-In mit MPWs, Java
KIT	„Programmieren“	Pflichtmodul	Bachelor	Traditionell, Java

An der Universität Stuttgart lehren wir die Grundlagen zur Softwareentwicklung in der Vorlesung „Programmieren und Softwareentwicklung“[1]. Hier setzen wir einen Outside-In Ansatz ein, der Studierende besser für die Entwicklung komplexer Softwaresysteme vorbereitet und wichtige Paradigmen intensiver behandelt. Aufbauend darauf kombinieren wir unseren Ansatz mit innovativ angepassten Ideen des Gamifications, wofür wir sogenannte “**M**ini-**P**rogrammier-**W**elten (MPW)” einsetzen. In den letzten Jahrzehnten wurden zunehmend solche auf den Bottom-Up-Ansatz limitierte MPWs eingesetzt, z.B. Roboter Karol, Kara der Marienkäfer oder der Hamster Simulator. Studierende sollen hier in einem spielerischen Ansatz beispielsweise Hamster durch eine kleine, einfache virtuelle Welt steuern und kleine Aufgaben lösen. In unserer Lehrveranstaltung setzen wir eine eigene Outside-In spezifische Variante des Hamster Simulators ein und stellen dafür Übungsaufgaben, um die erforderlichen Lehrkonzepte zu vermitteln. Beispielsweise muss der Hamster Körner sammeln oder einen Weg durch das aktuelle Labyrinth finden.

Am KIT werden in einer vergleichbaren Vorlesung „Programmieren“[2] die Grundlagen der Softwareentwicklung gelehrt, welche einen traditionellen Bottom-Up Ansatz verfolgt. Es werden die grundlegenden Strukturen und Details der Programmiersprache Java gelehrt, mit einem Fokus auf Kontrollstrukturen, einfache Datenstrukturen und dem Umgang mit Objekten. Außerdem werden die Implementierung nichttrivialer Algorithmen und die grundlegende Programmiermethodik erzielt. Zudem spielt die eigenständige Erstellung mittelgroßer Java-Programme eine wichtige Rolle.

In der Praxis werden neben Java verschiedene Programmiersprachen eingesetzt, die sich unterschiedlich gut für gegebene Probleme eignen. Beispielsweise werden für eingebettete Systeme häufig maschinennahe Sprachen eingesetzt, während sich für Applikationen für das Web oder mobile Endgeräte üblicherweise andere Programmiersprachen eignen. Weit verbreitete Beispiele für Programmiersprachen neben Java sind C++ oder Python. Aufgrund des unterschiedlichen Bedarfs werden an anderen Hochschulen und Universitäten auch andere Programmiersprachen gelehrt.

Innovative Ideen zum Verbessern unserer Lehre an der Universität Stuttgart

Während unser Ansatz an der Universität Stuttgart für viele Teilnehmer aktuell bereits gut funktioniert und eine anhaltende Motivation ermöglicht, gibt es durch den zunehmenden Anteil von Teilnehmern interdisziplinärer Studiengänge einen dringlicheren Bedarf an mehr Abwechslung und Flexibilität hinsichtlich Programmiersprachen und Übungsaufgaben. Dies erfordert zwingend zusätzliche Innovation in unserem Ansatz, welche ermöglicht, unsere Lösung schneller für weitere Programmiersprachen zu erweitern und neben dem Hamster Simulator auch andere Beispielwelten anzubieten. Insbesondere können wir hier bereits bewährte MPWs durch unsere flexible Methode auf den Outside-In Ansatz adaptieren und leicht nach Bedarf an eine spezifische Programmiersprache anpassen. Damit könnten wir einerseits die Lehre optimal den Anforderungen der diversen Studiengänge anpassen, da bspw. Studierende aus dem Bereich Mechatronik häufig die Programmiersprache C++ beherrschen müssen, während in Bereichen wie künstliche Intelligenz oder Data Science das Beherrschen der Sprache Python eine unabdingbare Voraussetzung ist. Andererseits sinkt die wichtige Motivation signifikant bei einer mehrmonatigen Lehrveranstaltung, wenn stets

[1] https://campus.uni-stuttgart.de/cusonline/pl/ui/$ctx/wbLv.wbShowLVDetail?pStpSpNr=282170
[2] https://sdq.kastel.kit.edu/wiki/Vorlesung_Programmieren_WS_2022/23

dieselbe MPW eingesetzt wird. Indem wir unseren Ansatz leicht für weitere MPWs erweitern, könnten wir die Übungsaufgaben abwechslungsreicher gestalten oder sogar innerhalb einer Lehrveranstaltung für jeweilige Zielgruppen individuell auswählen, um die Aufgabenstellung flexibel an einen Studiengang anzugleichen. Als Beispiel könnten Studierende aus der Automobilbranche eine größere Motivation für eine vertrautere MPWs aufbringen, die mit einem LKW Waren von einem Ort an einen anderen transportieren muss. Zeitgleich könnte dies flexibel in einer etablierten Programmiersprache für die Automobilbranche erfolgen. Die heutigen MPW Ansätze können hinsichtlich dieser Anforderungen nur schwierig in einer agilen und wenig fehleranfälligen Art und Weise erweitert werden. Für diese notwendige Flexibilität benötigen wir daher einen innovativen Weg, um möglichst agil neue Programmiersprachen zu erweitern oder neue MPWs zu entwickeln.

Um diese Ideen umzusetzen, haben wir kürzlich in einer Masterarbeit einen geeigneten Ansatz evaluiert, der Code-Generatoren einsetzt, um solche MPWs in einer flexiblen Art und Weise für unterschiedliche Programmiersprachen zu generieren. Gleichzeitig erlaubt dieser Ansatz, auf einer Modellierungsebene in kürzerer Zeit eigene MPWs zu erstellen oder flexibel zu erweitern, ohne sich vorher ein tieferes Verständnis der zugrundeliegenden Architektur aneignen zu müssen. Ein wesentlicher Vorteil einer neu generierten MPW ist außerdem, dass diese automatisch die wichtigen “Best Practices” und Konzepte demonstriert, die wir lehren.

Außerdem haben wir eine weitere innovative Idee erprobt, um Feedback-Zyklen auf Basis der dynamischen Beobachtung von Testläufen zu verbessern. Dies basiert auf Linear-Temporal-Logic (LTL) Formeln. Studierende lösen mehrere Übungsaufgaben auf Basis unseres Hamster Simulators mit Quellcode, welche zur Laufzeit eine Ausführungshistorie erzeugen. Um diese Lösungen automatisiert zu verifizieren, definieren wir für jede Aufgabe LTL-Formeln und können somit unseren Aufwand zum Prüfen deutlich reduzieren. Dies erlaubt mehr Zeit, um auf qualitative Rückfragen von Studierenden einzugehen.

Tandem-Partnerschaft mit dem KIT

Wir möchten unseren Ansatz gemeinsam mit dem KIT weiterentwickeln und unsere positiven Erfahrungen transferieren. Indem wir unsere Entwicklung mit einer weiteren Universität abgleichen, streben wir eine flexiblere Lösung an. Durch den Abgleich vermeiden wir, dass die Lösung nur für unseren Ansatz an der Universität Stuttgart geeignet ist.

Unser Ziel ist eine öffentlich verfügbare Plattform, die als Produktlinie für die Entwicklung von MPWs genutzt werden kann. Die entwickelten Artefakte sollen dafür im zentralen OER-Repositorium veröffentlicht zu werden. Somit können Dozierende in kürzerer Zeit für mehr Abwechslung in der Lehre sorgen und ihre Lösung an aktuelle Anforderungen besser anpassen. Zudem soll diese Plattform flexibler einsetzbar sein, indem beispielsweise direkt im Web-Browser auf einem Gerät ohne besondere Anforderungen programmiert werden kann und MPWs ohne vorherige Installation von Entwicklungsumgebungen eingesetzt werden können. Mittelfristig planen wir außerdem moderne Lehrmaterialien auf Basis unseres Ansatzes zu konzipieren, um unsere Methoden und Lösungen für andere Lehrbetriebe wiederverwendbar zu machen. Diese können wir ebenfalls durch das Tandem-Fellowship für eine breitere Zielgruppe anfertigen und eine höhere Flexibilität sicherstellen.

Um den Erfolg nach der Erprobung unseres Entwicklungsvorhabens zu beurteilen, könnte der erweiterte Ansatz in einem Semester am KIT beispielhaft eingesetzt werden. Mittels Feedbacks der Studierenden zu den neuen Inhalten könnte anschließend quantitativ untersucht werden, ob der Transfer erfolgreich war. Eventuelle Risiken können nach dem Entwicklungsvorhaben qualitativ durch die Erfahrung der Lehrstuhlverantwortlichen ermittelt werden.