Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

Prof. Dr.-Ing. MSc. Markus Dahm
Informatik & Software-Ergonomie

Münsterstraße 156
Gebäude 4, Raum 2.010
40476 Düsseldorf

T +49 211 4351-3260

markus.dahm@hs-duesseldorf.de
medien.hs-duesseldorf.de/dahm

Düsseldorf, 2.7.2018

Bewerbung für ein Fellowship

für Innovationen in der Digitalen Hochschullehre aus der Gemeinsamen Programmlinie des Ministeriums für Innovation, Wissenschaft und Forschung des Landes Nordrhein-Westfalen und des Stifterverbandes

Projekttitel:

Integriertes interaktives Lernen und Lehren - insbesondere der Grundlagen der Programmierung

Meine persönliche Motivation

Seit vielen Jahren lehre ich Programmieren für Unerfahrene und versuche, ihnen Grundzüge und Konzepte näher zu bringen, wie man von einem gegebenen Problem zu einem Programm als Lösung kommt. Dabei habe ich viele verschiedene Schwierigkeiten und Probleme bei meinen Studierenden erlebt, die z.B. durch kognitive Überlastung ihre Motivation verlieren. Dem möchte ich mit den hier beschriebenen didaktischen Konzepten und ihrer technischen Unterstützung begegnen und so für Studierende und auch für Lehrende den Lernerfolg und auch die Freude am Lernen verbessern. Durch die voranschreitende Digitalisierung werden grundlegende Programmierkenntnisse auch in vielen anderen Disziplinen wichtig, was die Bedeutung der Verbesserung der Lernbedingungen in diesem Bereich noch erhöht.

Die Erfahrung und Ansätze von Kolleginnen und Kollegen kennen zu lernen, erweitert nach meiner Erfahrung meinen Horizont und bringt mich auf neue Ideen, die das Projekt voranbringen können. Natürlich nehme ich sehr gerne an den geplanten Fellow-Treffen und Konferenzen aktiv teil, wodurch sich hoffentlich weitere, auch interdisziplinäre Kontakte ergeben, die neue Erkenntnisse für die Übertragbarkeit der hier vorgestellten Konzepte ergeben können.

Von dem Fellowship verspreche ich mir persönlich sowohl, dass das hier beschriebene Vorhaben, das mir sehr am Herzen liegt, unterstützt wird, als auch eine Erweiterung meines Netzwerks.

Die folgenden Probleme sollen gelöst werden -> Zieldefinitionen

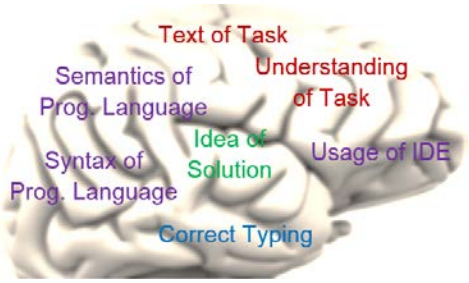
Grundlagen des Programmierens sind in vielen Studiengängen wichtig, nicht nur in der Informatik, sondern aufgrund der fortschreitenden Digitalisierung in steigendem Maße auch in vielen anderen Disziplinen und Fächern. Zusätzlich bilden sich auch immer mehr Berufstätige verschiedenster Sparten auf diesem Gebiet weiter.

„Ich habe mir die Aufgabe angesehen, habe aber keine Ahnung, wie und wo ich anfangen soll“ oder „Ich weiß, wie ich eine while-Schleife schreibe, ich weiß nur nicht, wann ich die hinschreiben soll.“ – das sind typische Aussagen von Lernenden, die als Anfänger[1] eine Aufgabe bekommen, die Sie mit Hilfe eines Programms lösen sollen. Sie deuten auf verschiedene Schwierigkeiten hin, die weit verbreitet sind.

Im Folgenden werden zentrale Probleme aufgeführt, die vielen Programmier-Anfängern zu schaffen machen. Daraus werden Ziele für die Lehrinnovation abgeleitet.

Die Lernsituation von Programmieranfängern ist von starker mentaler Belastung geprägt

Programmieranfänger werden viel über Syntax und Semantik von Programmiersprachen unterrichtet, sie bekommen deren Anwendung auch an Beispielen erläutert. Zur Umsetzung und Festigung dieses Wissens in praktischen Anwendungen müssen aber (fast) immer mehrere Hürden überwunden werden:



Studierende müssen gleichzeitig 1) den Text der Aufgabenstellung, 2) das eigene Verständnis der Aufgabe, 3) die semantischen und 4) syntaktischen Mittel der verwendeten Programmiersprache „im Kopf haben“. In diesem Kontext soll 5) ein Lösungskonzept entwickelt werden und dann auch noch 6) fehlerfrei und richtig formatiert eingetippt und dabei 7) eine (komplexe) integrierte Entwicklungsumgebung (IDE) genutzt werden.

Damit ergibt sich eine sehr hohe kognitive Belastung nach der Cognitive Load Theory von Sweller und Chandler. Anders als Erfahrene, die sich auf gefestigte Schemata, Mentale Modelle und erlernte Pläne stützen können, und damit das Arbeitsgedächtnis entlasten, müssen Anfänger alle sieben oben aufgeführten Bereiche als gesamten *Kontext* gleichzeitig im Arbeitsgedächtnis behalten.

➔ *Ziel 1) Die Lehrinnovation soll die kognitive Belastung von Programmieranfängern gezielt verringern.*

Programmieren wird nur als „Codieren“ wahrgenommen

Die Kompetenz, in einer Programmiersprache codieren zu können ist ein wesentliches Lernziel, da so das sichtbare Ergebnis der Problemlösung entsteht. Software zu programmieren bedeutet aber natürlich nicht nur das reine Schreiben von Code, sondern

[1] Eine Gender-gerechte Sprache habe ich fast überall umsetzen können, an manchen Stellen jedoch zugunsten des leichteren Leseflusses auf das generische Maskulinum („Anfänger“) zurückgegriffen.

erfordert vor allem zur Vorbereitung die geistige Auseinandersetzung mit der Aufgabe und den Überlegungen für die eigene Lösung. Diese Vorgehens-Schritte sind also wesentlich, sie sind in der Grundlagen-Lehre aber häufig unterrepräsentiert.

Die dem Codieren, das heißt dem reinen Eintippen von Code vorangehenden wesentlichen Phasen der Analyse der Aufgabe sowie dem Design der Lösung werden außerdem in keinem Tool unterstützt oder gar nahegelegt (siehe auch unten bei Ziel 4).

Programmieranfängern wird so (häufig ungewollt) vermittelt, dass das reine Codieren deutlich wichtiger ist als Analyse und Design. So wird – überspitzt formuliert – ein trial-and-error Prinzip erlernt, was für eine berufliche Zukunft in professioneller Softwareentwicklung sehr unvorteilhaft ist.

➔ *Ziel 2) Diese didaktische Lücke soll gezielt geschlossen werden, alle Phasen des Programmierens sollen gleichwertig behandelt werden. So wird nicht nur der Lernerfolg gesteigert, sondern auch von Anfang an auf eine professionelle Software-Entwicklung hin gelernt.*

Konzepte und Features von Programmiersprachen werden häufig ohne Kontext dargestellt

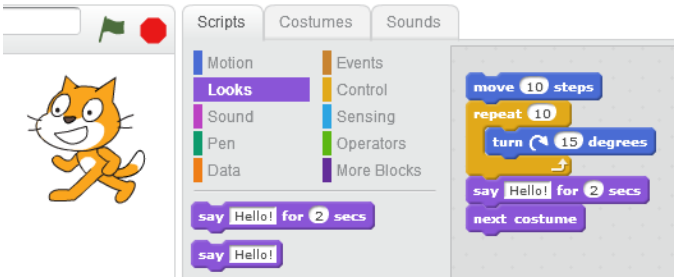
Bücher und andere Lehrmittel stellen sehr häufig die Features von Programmier-sprachen in den Vordergrund: Was ist eine Schleife, wie sieht diese im Code aus, etc. Wer bereits Vorkenntnisse oder hohe Kompetenz in der Informatik besitzt, kann diese Eigenschaften und vor allem ihren Nutzen einordnen. Wenn die Beschreibung der Problemstellung fehlt, können aber Anfänger sich nicht von alleine vorstellen, wozu man komplizierte Dinge wie z.B. Generics lernen soll; in meinem Vorlesungsfeedback wurden sogar „Beispiele von Schleifen im echten Leben“ eingefordert.

Nachgeschobene Anwendungen und Beispiele kommen häufig zu spät, da ohne hinreichende Motivation nicht vollständig die Beschreibung der Features aufgenommen werden konnte.

➔ *Ziel 3) Konzepte, Eigenschaften und Features in den Grundlagen der Informatik sollen vom Problem zur Lösung entwickelt werden*[2]*, ihr Sinn und Nutzen soll so klar werden*.

Für jede Entwicklungsphase müssen andere Werkzeuge verwendet werden

Erstaunlicherweise sind nicht nur professionelle IDEs wie Eclipse, NetBeans oder IDEA *rein Code-zentriert*, sondern auch diejenigen Lernumgebungen, die sich explizit an Anfänger bis hin zu Schülern richten: Auch aktuell eingesetzte Mikrowelten wie Scratch (siehe Abbildung), Kara, Greenfoot oder auch interaktive IDEs wie BlueJ bieten immer lediglich die Phase der Umsetzung in



[2] Diese Vorgehensweise ist in der Informatik bekannt aus der kanonischen Beschreibung von Software-Entwurfsmustern. Sie wird dort erfolgreich angewandt, erstaunlicherweise aber sonst eher selten.

Code. Weder können in ihnen Aufgabenstellungen dargestellt oder bearbeitet werden, noch kann man sinnvoll eigene Notizen oder gar ein Konzept erstellen; lediglich Code kann erstellt werden – ob mit bunten Blöcken oder mit Programmiersprachen wie Logo, Java oder Python. Die gleiche Situation herrscht bei online angebotenen Kursen vor, z.B. bei codecademy.com, KhanAcademy.com oder code.org: Es geht immer nur um das Erlernen des *Codierens*, der Umsetzung in eine Programmiersprache.

Es müssen also mehrere Programme eingesetzt werden: für 1) die Arbeit mit der Aufgabenstellung (z.B. pdf-files, Ausdrucke), 2) eigene Notizen (Papier, Texteditor) und 3) die Umsetzung der Lösung in Code (IDE). Durch den somit notwendigen, ständigen Wechsel zwischen mehreren Tools ist es noch schwieriger, den gesamten Kontext präsent zu haben - insbesondere für Anfänger.

➔ *Ziel 4) Alle Inhalte – vorgegebene, beschreibende, illustrierende und selbst erstellte – sollen durch die Lerninnovation in einem Kontext integriert werden*

Studierende haben häufig wenig Übersicht über ihren Lernstand

Viele Studierende wissen nicht, was sie vom vermittelten Lerninhalt noch nicht verstanden haben. Sie machen sich auch häufig nicht klar, wo ihre Schwierigkeiten liegen und wissen daher nicht, wie sie diese Situation verbessern können. Dieser ungünstige Kreislauf lässt sich durch Selbstreflektion der Lernenden unterbrechen, ein häufig eingesetztes Mittel dafür ist ein Lerntagebuch.

Das Lerntagebuch kann separat geführt werden, z.B. handschriftlich auf Papier. Es ist aber sicher sinnvoll, die Reflexionen gemeinsam mit den Unterlagen zu Vorlesung und Praktikum – besonders gemeinsam mit den eigenen Lösungen – zu führen. So können beim Nach- und Vorbereiten des Stoffs direkt die auftretenden Fragen notiert werden.

➔ *Ziel 5) Das Schreiben eines Lerntagebuchs soll nahtlos in den Kontext der Lern-Dokumente in der Lerninnovation integriert werden.*

Einfache Konzepte können nicht einfach ausprobiert werden

Die in den ersten beiden Semestern typischerweise gelernten Informatik-Inhalte sind noch eher einfach (im Sinne von punktuell und einzeln behandelbar), wie Datentypen, Klassen, Generics. Zum Erlernen und Festigen der so erworbenen Kompetenzen sollten sie so schnell und so viel wie möglich von den Lernenden selber angewendet werden. Einem einfachen Ausprobieren, stehen aber komplexe IDEs gegenüber, die einen hohen Initialaufwand für das Testen verlangen: Einarbeitung in Projektstrukturen, diverse Ansichten der Organisation von Code und Files, Debugger, Konsolen, etc.

➔ *Ziel 6) Konzepte des Codierens sollen einfach ausprobiert werden können, das Erlebnis der Selbstwirksamkeit soll so gestärkt werden. Der Code soll dabei immer seinen Bezug zur Aufgabenstellung bzw. zum Konzept behalten. Automatische Rückmeldungen können Hinweise auf Fehler geben.*

Vorlesungen sollen durch „Live Coding“ anschaulicher werden, das ist aber aufwändig

In Informatik-Vorlesungen kann man viele Inhalte und Konzepte durch Beispiel-Code veranschaulichen. Wie in vielen Evaluationen von Veranstaltungen deutlich wird,

wünschen sich Studierende, dass dieser Code nicht nur statisch gezeigt, sondern auch ausgeführt wird, um die Effekte und Auswirkungen sofort sehen zu können. Außerdem können so spontane Beiträge der Lernenden sofort aufgegriffen und getestet werden.

Das ist möglich, ist aber häufig umständlich, da typische Code-Werkzeuge (IDEs) wie Eclipse sehr leistungsfähig aber (vor allem für Anfänger) unübersichtlich sind und auf einem Beamer-Bild nicht gut auf die wesentlichen Inhalte fokussiert werden kann. Außerdem geht beim Darstellungs-Wechsel von einer Folie auf eine IDE der Bezug zur Aufgabenstellung und damit der Problemkontext verloren.

➔ *Ziel 7) Live-Coding soll in eine Vorlesung einfach integriert werden können. Dabei soll der Bezug zur Aufgabenstellung und zum beschriebenen Lösungsansatz immer sichtbar bleiben.*

Ziele und Lösungsansätze

Die oben vorgestellten und mit ihren Problemstellungen begründeten Ziele 1) bis 7) werden hier noch einmal kurz zusammengefasst

1) Verringerung der kognitiven Belastung der Lernenden
2) Einbeziehung aller Phasen der Software-Entwicklung
3) Problembeschreibungen motivieren die Lösungsansätze
4) Integration aller Lehr/Lerninhalte
5) Einbeziehung eines Lerntagebuchs
6) Verstärkung der Selbstwirksamkeit durch vereinfachtes Ausprobieren
7) Live-Coding in der Vorlesung im Kontext der Problemstellung

Sie sollen im Fellowship erreicht werden durch einen doppelten Lösungsansatz:

- Durch einen durchgängigen *didaktischen Ansatz* vom Problem zum Programm: ***Integration*** aller Lehr/Lerninhalte, die weitgehend ***interaktiv*** nutzbar sind. Dieser Ansatz soll weiter ausformuliert, umgesetzt und evaluiert werden.

- Durch die Weiterentwicklung und Evaluation der *technischen Unterstützung* durch die in Vorarbeiten entwickelte integrierte Umgebung.

Vorarbeiten

In Vorarbeiten wurde bereits in mehreren Iterationen ein didaktisches und technisches ***Grundkonzept*** erarbeitet, evaluiert und ständig verbessert [Da15] [Da16] [Da17], um ***Inhalte integriert und interaktiv verfügbar zu machen***.

A) Die für Programmieranfänger eher abstrakten Phasen des Software-Engineering werden in fünf leicht verständliche und sofort umsetzbare Schritte operationalisiert:

Lesen → Verstehen → Überlegen → Aufschreiben → Codieren

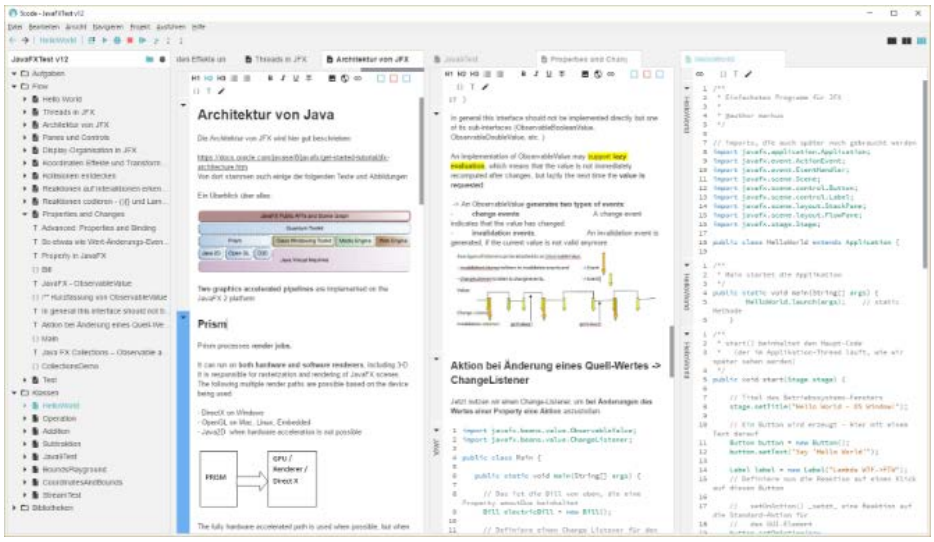
B) Meine Vorlesungen zur Objektorientierten Programmierung, OOP1 und OOP2, orientieren sich an diesen Schritten, die damit explizit auch zuerst das Problem darstellen, danach erst die folgenden Schritte zur Lösung.

Zur Unterstützung speziell dieses Ansatzes wurde als Tool die IDE *5Code* entwickelt. Informationen und Funktionalitäten für die genannten fünf Schritte werden integriert verfügbar und bearbeitbar gemacht (siehe Abbildung unten).

- Beschreibungen, Aufgaben, eigene Überlegungen und Code werden *gemeinsam* in *einem Fenster* dargestellt.

- Alle Dokumente und Artefakte aller fünf Bearbeitungs-Schritte sind immer zugänglich und können gleichzeitig (!) angesehen und bearbeitet werden. Der komplette Arbeits-Kontext in einer einzigen Anwendung zugänglich.

-> Die mentale Belastung wird so stark gesenkt.



C) Eine weitere *einzigartige Besonderheit* ist die Möglichkeit, Beschreibungen, Überlegungen, Notizen, Teile der Aufgabe und andere Text-Teile formatiert in einem *einzigen* Dokument schreiben zu können, und diese Texte mit dem beschriebenen Code zu *mischen*. *Text, Grafik und Code* stehen also nicht in getrennten Dokumenten, sondern im gleichen Dokument. Der *Kontext* wird so immer gewahrt.

Außerdem wird der Arbeitsfluss nicht durch den Wechsel zwischen verschiedenen Anwendungen oder Fenstern, gestört, der Fluss der Gedanken kann durchgängig fließen, daher heißt dieses Konzept „*Flow-Programming*“

D) Weiterhin „leben“ Querverweise von einem Dokument in ein anderes: Markierte Textstellen in Ausgangsdokumenten können mit darauf Bezug nehmendem Text oder Code in selber erstellten Artefakten in beiden Richtungen miteinander verknüpft werden. Man kommt so immer sehr schnell (mit einem Klick auf den Querverweis) vom identifizierten Problem zur Beschreibung der Lösung – und von der Lösung auch wieder zurück zur zugehörigen Problembeschreibung.
-> Der *Kontext von Problem und Lösung* wird so gewahrt und ist leicht zugänglich.

Der Code wird in Java geschrieben, er kann kompiliert und ausgeführt werden. Fehler dabei werden wie in anderen IDEs direkt im Quellcode angezeigt.

Ergebnisse der Evaluationen

Der didaktische Ansatz der Integration aller Dokumente von den Aufgaben bis zur Lösung in 5Code wurde in mehreren Semestern im Praktikum der Veranstaltung Objektorientierte Programmierung eingesetzt und von den Studierenden evaluiert. Dieses Grundkonzept hat sich dabei prinzipiell als sehr tragfähig erwiesen und *bietet großes Potential*. Es ist aber noch viel zu *erweitern, ergänzen* und *verbessern*, auch aus weitergehenden Anfragen von interessierten Hochschulen.

Weiterentwicklung und Ausbau im Fellowship

Bisher werden Informationen und Basis-Funktionen für alle Phasen geboten, nun sollen diese für unterschiedliche Anwendungsfälle optimiert und ausgebaut werden. Dazu sind vertiefte Konzepte und Funktionalitäten nötig, für die sowohl konkrete didaktische Ziele als auch ein Nutzungskonzept erstellt und umgesetzt werden soll:

- Erstellung von Vorlagen für die interaktive Vorlesung (Ziele 2), 3))
- Interaktive Durchführung der Vorlesung, incl. Live Coding (Ziel 7)
- Veröffentlichung der Vorlesungsinhalte (Script), incl. Beiträge des Auditoriums
  ➔ Beispielhafte Skripte für die Vorlesung Objektorientierte Programmierung
- Erstellung von Praktikumsaufgaben als Ergänzung des Skripts (Ziel 4))
  ➔ Beispielhafte Praktikumsaufgaben für die Objektorientierte Programmierung
- Nachbearbeitung des Scripts und Bearbeitung der Aufgaben durch Lernende (Ziele 1), 2), 3), 4))
- Integration eines Lerntagebuchs, u.a. mit Struktur-Vorlagen (Ziel 5)
- Exploration von Programmier-Konzepten und Features im Code
- Betreuung und Abnahme von Lösungen durch MitarbeiterInnen und TutorInnen
- Aufgrund des integrierten Ansatzes, der die Ziele 1), 2), 3) und 4) prinzipiell sehr gut adressiert, sind zwar alle Informationen vorhanden, die Übersicht hat sich aber als schwierige Herausforderung herausgestellt. Die Navigation soll daher von Grund auf neu konzipiert werden und eine deutlich verbesserte Orientierung zu bieten.
- Verbesserung der Usability für Lehrende und Lernende auf Basis der vorliegenden und noch erhobenen Rückmeldungen
- Anfänger sollen, z.B. im Script-Kontext, Features und Elemente der Programmier-sprache mit einfachen „Code-Snippets“ ausprobieren können, ohne, wie in einer IDE, eine komplette Projekt/Klassen-Umgebung aufsetzen zu müssen (Ziel 6).
- Anfänger sollen durch weitere Funktionen unterstützt werden, die eine viel tiefer gehende automatische Analyse des Codes und seiner Ergebnisse erfordert, z.B. welche Anweisungen wo möglich sind, welche Fehler drohen, ob die geforderten Funktionen wie gefordert arbeiten, etc. Die wichtige Eigenschaft des Lernens durch Exploration (Ziele 3), 6)) wird so stark unterstützt.
- Um auch Vorlesungen (Ziel 7) integriert zu unterstützen soll ein intelligentes Konzept erarbeitet und umgesetzt werden, dass es ermöglicht, dass Dozierende sowohl eigene Notizen sehen und nutzen können als auch Inhalte zur Darstellung für die Lehrenden auswählen. Sie sollen live Text, Grafik und Code ergänzen können.
- Weitere Sprachen sollen integriert werden, mit z.B. Kotlin, HTML/CSS/Javascript und C ein breiter Bereich von Grundlagen-Sprachen. Außerdem für den Transfer in andere Disziplinen das Erzeugen von formatiertem Text anstelle von Code.

Diese Lehrveranstaltungen können primär profitieren

Die Lehrinnovation soll in folgenden Lehrveranstaltungen vom Antragsteller selber eingesetzt werden: Objektorientierte Programmierung 1 (Pflicht), Informatik-Projekt 1 (Wahlpflicht)

Darüber hinaus ist ein zumindest teilweiser oder optionaler Einsatz geplant in: Objektorientierte Programmierung 2 (Pflicht), Webprogrammierung (Pflicht), Software-Engineering (Pflicht), Informatik für Ingenieure (Pflicht).

Der Erfolg sowie die Risiken können sinnvoll bewertet werden

Als Erfolg wird angesehen, dass Studierende durch verbesserte Bedingungen in Kursen für in Programmieranfänger, profitieren. Für sie kann dadurch die Motivation erhalten oder sogar erhöht werden, Durchfallquoten und Studienabbruchquoten könnten gesenkt werden. Lehrende profitieren dadurch ebenfalls, wie auch durch die vereinfachte Organisation ihrer Lehrinhalte und die neuen interaktiven Möglichkeiten.

Die Lehrinnovation lässt sich gut in Lehrveranstaltungen erproben und mehrstufig evaluieren. Dazu hat sich in den Vorarbeiten folgender Ablauf bereits bewährt:

- Das oben definierte und mit dem Tool unterstützte didaktische Konzept wird schrittweise in den Lehrveranstaltungen in Vorlesungen und Praktika eingesetzt
- Die Didaktik und die Tool-Unterstützung wird direkt in den Lehrveranstaltungen durch die Projektgruppe begleitet und unterstützt:
    - Auftretende Probleme und Anregungen werden aufgenommen.
    - Probleme können häufig schnell behoben werden.
    - Die Arbeitsweise wird beobachtet und dokumentiert.
- Mindestens einmal im Semester wird die Lehrinnovation mit einer Umfrage durch die Studierenden bewertet.
- Gegen Ende der Vorlesungszeit werden die Beobachtungen mit allen Praktikums-Betreuern und der Projektgruppe reflektiert und besprochen.

Aus all diesen Evaluationen ergeben sich Hinweise für die weitere Entwicklung und Verbesserung.

Aufgrund der Erfahrungen in den Vorarbeiten wird die Erfolgswahrscheinlichkeit als recht hoch angesehen, die Risiken als beherrschbar. Sowohl der Erfolg als auch die Risiken des Einsatzes lassen sich sehr gut ermitteln und beurteilen:

- Die systematische Beobachtung des Nutzungsverhaltens während des laufenden Semesters zeigt die Meinung recht ungeschminkt
- Die formale Evaluation mittels Fragebogen unterstreicht nach unserer Erfahrung in den Vorarbeiten diese Beobachtungs-Ergebnisse
- Eine Betrachtung der Bestehens-Quote bei Prüfungen sagt zusätzlich etwas über den Erfolg der Lehrinnovation aus

Die Lehrinnovation kann intern und extern verstetigt werden

Das oben beschriebene didaktische Konzept und die unterstützende IDE soll intern im Fachbereich Medien in diesen Modulen eingesetzt und weiterentwickelt werden:

Objektorientierte Programmierung 1 und 2, Informatik-Projekt 1, Webprogrammierung, Software-Engineering, Informatik für Ingenieure

Der Fachbereich wird Mittel zum Einsatz, zur Betreuung und zur Weiterentwicklung bereitstellen. Damit kann die Lehrinnovation intern verstetigt werden.

Darüber hinaus soll das Tool und das Lehrkonzept auch Kolleginnen und Kollegen aus anderen Fachbereichen angeboten werden kann. Geplant ist ein Angebot für

- FB Elektro- und Informationstechnik, Informatik 2 – Software-Technik
- FB Maschinenbau und Verfahrenstechnik, Informatik 1 – Programmierung

Schließlich soll das weiterentwickelte Konzept und das entstandene Tool auch anderen Hochschulen angeboten werden. Anfragen und Austausch gibt es bereits mit

- HTWK Leipzig, Informatik, Mathematik und Naturwissenschaften, Mario Wenzel
- Heinz-Nixdorf-Institut, Universität Paderborn, Dr. Harald Selke
- hdw nrw, Prof. Dr. Pook, Dipl.-Päd. Schumacher
- FH Südwestfalen, FB Elektro- und Informationstechnik, Prof. S. Alisan-Ipek
- Arbeitskreis Informatik des hdw nrw, Annett Garten
- Uni Duisburg-Essen, Univ.-Prof. Dr. Maik Masuch

Im Kontakt mit weiteren Fellows dieses Programms sollten sich Kontakte ergeben, die über das hier beantragte Projekt hinausgehen und auch in zukünftigen Vorhaben zur innovativen Hochschullehre von allen Beteiligten genutzt werden können.

Die Ergebnisse der Arbeiten im Fellowship sollen auch auf Fachkonferenzen, z.B. DeLFI (e-Learning Fachtagung Informatik) und Messen wie der didacta und der Learntec als Beitrag eingereicht werden um den Adressatenkreis noch zu erweitern.

Damit werden die beschriebenen besonderen Eigenschaften der Lehrinnovation einem weiteren, wachsenden Kreis von Studierenden zugänglich gemacht. So kann die Lehrinnovation auch extern verstetigt werden.

Die Innovation kann übertragen werden, auch Disziplinen-übergreifend

Andere Informatik-Bereiche

Die oben beschriebenen Lehr/Lern-Konzepte können auch in andere Programmier-Sprachen und Paradigmen übertragen werden. Objektorientiert: Python, C#, C++, Kotlin; Prozedural: C; Web: HTML, CSS, Javascript, Typescript.

Eine Übertragung in andere Informatikbereiche, insbesondere in die Informatik für Ingenieure (Elektrotechnik, Informationstechnik, Maschinenbau), ist geplant. Dazu wird unter anderem die European Society für Engineering Education mit einbezogen.

Andere Disziplinen

Darüber hinaus kann das Konzept der Integration von Artefakten aller Phasen auch bei jeglicher Arbeit mit Texten eingesetzt werden, auch wenn als Ergebnis kein Code, sondern wiederum Text entsteht. Die Phasen Analyse, Notizen, Überlegungen, Synthese werden auch in typischen Arbeiten in den Ingenieurs-, Natur- und Geistes-wissenschaften durchlaufen und können daher auch von der Lehrinnovation unterstützt werden: Philosophen der Technischen Universität Darmstadt arbeiten beispielsweise bereits seit geraumer Zeit an Tools für die effiziente Unterstützung der Arbeit mit elektronisch vorliegenden Texten. Das zeigt, dass speziell die Integration von Ausgangs-Dokumenten mit den daraus entstehenden und erarbeiteten Texten sowie die wechselseitige Verknüpfung von Textstellen und Referenzen in einem einzigen Tool auch in diesem Fachgebiet sinnvoll sind.

Das vorgestellte Konzept und das darauf basierende Tool hat damit das Potential, bei der Erarbeitung und Analyse von jeder Art von Texten eine Alternative oder Ergänzung zu üblichen Werkzeugen wie Texteditoren zu sein.

Weiterbildung und Erwachsenenbildung

Auch in der Weiterbildung und Erwachsenenbildung werden Kurse für Programmier-anfänger angeboten. Diese Lernenden sind typischerweise schon älter, sie werden nicht selten von der Digitalisierung in die Informatik gedrängt. Sie fühlen sich einerseits von der bunten Anmutung von Mikrowelten (wie in der Abbildung auf S.3 beispielhaft gezeigt) sicher nicht ernst genommen. Andererseits sind sie aber häufig auch von den ungewohnten und komplexen IDEs wie Eclipse eingeschüchtert, die für nicht-triviale Anwendungen unnötig kompliziert erscheinen. Zudem müssen sie mehrere Programme erlernen, um alle Dokumente lesen und bearbeiten zu können. Die zu Beginn geschilderten Probleme und Schwierigkeiten treten also auch für diese Personen auf.

Hier kann das hier vorgestellte integrierende didaktische Konzept ebenfalls sehr gut unterstützen: Man braucht nur ein einziges Tool erlernen und anwenden und muss nicht zwischen mehreren Programmen hin- und herschalten – sowohl kognitiv als auch operational. Die Anmutung ist ernsthaft und professionell aber in der Komplexität reduziert. Schließlich können alle Phasen des Software-Entwicklungs-Prozesses mit diesem einen Tool erlernt werden. Insgesamt wird so der zusätzliche Lernaufwand minimiert, trotzdem können interessante und Zielgruppen-relevante Aufgaben über alle Phasen bearbeitet werden.

Das Fellowship unterstützt die Innovation entscheidend

Die Erfahrung und Ansätze von Kollegen und Kolleginnen kennen zu lernen, erweitert nach meiner Erfahrung ganz allgemein den Horizont und bringen immer neue Ideen, die auch dieses Projekt voranbringen können. Das Fellowship bedeutet weitere Kontakte in dieser Richtung, sowohl im Bereich Informatik als auch besonders aus anderen Disziplinen worauf ich mich sehr freue. Weiterhin hoffe ich, dabei Interessierte zu finden, die die hier beschriebene Lehrinnovation erproben möchten.

Die finanzielle Unterstützung ermöglicht es erstmals, dass nicht nur vereinzelt Studierende an diesem vielversprechenden Projekt mit einzigartigen Merkmalen arbeiten können, sondern konzentriert über ein Jahr eine Projektgruppe Ideen gemeinsam entwickeln, diskutieren und umsetzen kann.

Die Innovation ist organisatorisch stark eingebunden

Der Antragsteller ist innerhalb der Hochschule Düsseldorf organisatorisch stark vernetzt, eine interne Verstetigung wird damit sehr gut unterstützt:

- Rolle als Co-Koordinator des Studiengangs Medieninformatik
- Rolle als Studiendekan des Fachbereichs Medien
- Kontakt mit Informatik-Kollegen der Fachbereiche Elektrotechnik und Informationstechnik sowie Maschinenbau und Verfahrenstechnik.
- Kontakt mit dem Zentrum für Weiterbildung und Kompetenzentwicklung der HSD
- Kontakt mit den Informatik-Kollegen der Studiengänge Medieninformatik und Medientechnik, insbesondere der Module Webprogrammierung, Software-Engineering, Informatik für Ingenieure
- Kontakt mit der hochschuldidaktischen Weiterbildung im hdw nrw
- Zahlreiche eigene Pflicht- und Wahlveranstaltungen in der Lehre (siehe Anlage)