Projektbericht zur Fördermaßnahme *Innovative Lehr- und Lernkonzepte Innovation Plus* (2019/20, Projektnummer: R1-138)

Universität Oldenburg, Prof. Dr. Andreas Winter. Weitere Beteiligte Hochschullehrende: Prof. Dr. Ira Diethelm, Prof. Dr. Sebastian Lehnhoff, Dr. Christian Schönberg

Projektbeschreibung

Fächergruppe: Ingenieurwissenschaften

Studiengang/Studiengänge: Bachelorstudiengänge der Informatik und Wirtschaftsinformatik

Modul/Module: Programmierung, Datenstrukturen und Algorithmen (PDA); Objektorientierte Modellierung und Programmierung (OMP); Softwaretechnik I (ST1)

Kurze Beschreibung des Projekts aus dem Antrag:

In der Programmier- und Softwaretechnikausbildung ist neben der Vermittlung theoretischer Grundlagen ein wichtiger Aspekt die praktische Anwendung und Umsetzung der gelernten Konzepte durch die Studierenden. Dabei geht es um Programmierung und Modellierung von einfachen bis zu komplexen Aufgaben als Aspekte der Softwareentwicklung. Primärziel des Projektes ist es, den Studierenden eine Plattform zum Selbststudium bereitzustellen, auf der sie zeitnah Rückmeldungen zu Fehlern und Problemen ihrer Programmier- und Modellierungslösungen erhalten, die aber gleichzeitig durch Automatisierung für bis zu 500 Studierende einer Veranstaltung skaliert. Die Plattform soll sowohl für Einzelarbeit als auch für Team-basierte Entwicklungsmethoden wie Pair-Programming oder kollaborative Modellierung ausgelegt sein. Teilziele des Projekts sind die Evaluation bestehender Systeme, die Einführung und ggf. die Integration ausgewählter Systeme, die Integration der Systeme in das Campusmanagementsystem Stud.IP zu einer Gesamtplattform, ein Testlauf mit der Plattform zur Begleitung von zwei Vorlesungen für ein Semester und die Evaluation der erzielten Ergebnisse.

Fokus der Maßnahme:

selbstgesteuertes Lernen

forschendes Lernen

digitale Lehr- und Lernmethoden

neue Lehrformen für Massenveranstaltungen

Einführung neuer hochschuldidaktischer Konzepte und Maßnahmen

Projektverlauf

Konnte das Projekt in der geplanten Form durchgeführt werden?

ja

Die zeitlichen Verschiebungen durch personelle Umstellungen hatten auf die konzeptionelle Durchführung des Projekts keinen negativen Einfluss, konnten sogar zur Erweiterung des Evaluationszeitraums genutzt werden.

Wurde die Umsetzung des Projekts durch formale Prozesse in der Hochschule bei der Umsetzung des Projekts beeinträchtigt? nein

Anzahl der Durchläufe im Förderzeitraum: mehr als zwei Durchläufe

Anzahl der Studierende, die insgesamt an den Lehrveranstaltungen / am Modul teilgenommen haben: 2048

Welche Prüfungsformen wurden im Modul eingesetzt?

Klausur
elektronische Prüfung

Mussten die Studierenden neben den Prüfungen weitere Leistungsnachweise erbringen? (Alle verpflichtenden Leistungen, die erbracht werden müssen, um die ECTS-Punkte für das Modul zu erwerben.) Falls ja, welche?

nein

Wie gut passt die Prüfungsform zum Modulkonzept?

sehr gut

Wie gut passen die anderen Leistungsnachweise zum Modulkonzept?

es gab keine anderen Leistungsnachweise

Wie wurden die Expertinnen und Experten für Hochschuldidaktik ins Projekt einbezogen?

intensiv:

Neben Feedbackgesprächen gab es einen Austausch mit der Hochschuldidaktik zu Konzeption und Anwendungsformen der entwickelten Systeme. So war die Hochschuldidaktik u.A. auch bei der Zwischenpräsentation (04.05.2021) vertreten. Außerdem waren die beteiligten Module in die üblichen Evaluationsmechanismen der Hochschuldidaktik eingebunden.

Nachhaltigkeit

Wird das angepasste Modul auch nach Ende der Projektlaufzeit in der veränderten Form weitergeführt? ja, auf jeden Fall

Wird das geförderte Konzept auf andere Module übertragen? vermutlich ja

Wird das geförderte Konzept auf andere Studiengänge übertragen? vermutlich ja

Ggf. Erläuterung zur Nachhaltigkeit:

Die entwickelte Selbstlernplattform wird auch nach Projektende weiter eingesetzt und durch die Ergänzung neuer Aufgaben weiter ausgebaut. Konkret kommt die Plattform in den drei Modulen der Programmierausbildung PDA, OMP und ST1 zum Einsatz, wo die zugehörigen Aufgaben bereits in mehreren Semestern von jeweils über hundert Studierenden bearbeitet wurden (Quelle: Stud.IP Downloads und Server-Logs). Eine Übertragung auf andere Module und beispielsweise andere Programmier- oder Modellierungssprachen (außerhalb der UML-Familie) ist konzeptionell möglich. Ebenso ist der grundsätzliche Ansatz, Lösungen von Übungsaufgaben durch Anfragen zu überprüfen auch auf andere Aufgabentypen in anderen Disziplinen übertragbar. Diese Ideen werden im für 2022 bewilligten, Projekt „UOL digital.präsent“ im Rahmen der MWK-Förderung „Digitalisierung plus - Sofortmaßnahmen der Hochschulen in Niedersachsen im Bereich Digitalisierung“ aufgegriffen und vertieft. Konzeptionell ist eine Erweiterung um kollaborative Modellierung für das Selbststudium vorbereitet, die in einem Folgeprojekt umgesetzt werden soll. Technische Nachhaltigkeit und Übertragbarkeit: Der Client der Plattform ist direkt als Plugin in die Entwicklungsumgebung integriert, die auch von den Studierenden in der Informatik verwendet werden. Für die Programmierung kommt hier Eclipse zum Einsatz, für die Modellierung wird Papyrus verwendet, welches wiederum auf Eclipse aufbaut. Eclipse ist eine stark verbreitete Entwicklungsumgebung, die als Open Source von einer großen Community gepflegt wird. Eine Einstellung der Unterstützung für diese wichtige technische Basis ist also in nächster Zeit nicht zu befürchten. Da der Client unabhängig von konkreten Aufgaben gestaltet ist, können neue Aufgaben auf dem Server ergänzt werden, die dann ohne Änderungen am Client direkt dort zur Verfügung stehen. Die Überprüfung der eingereichten Lösungen findet ausschließlich auf dem Server statt. Daher können auch die Überprüfungen und das Feedback serverseitig ausgebaut werden, ohne dass die Clients geändert werden müssen. Gleichzeitig erlaubt es der integrierte Update-Mechanismus, einfach Updates des Clients auszurollen, falls dies notwendig werden sollte. Da die Schnittstelle für die Einreichung von Programmcode auf dem Server textbasiert und insbesondere unabhängig von Eclipse ist, können zukünftig auch Java-Dateien aus anderen Entwicklungsumgebungen eingereicht werden. Ebenso ist auch das Feedback plattformunabhängig gestaltet, so dass ein entsprechend, zu entwickelndes Plugin das Feedback vom Server dann auch z.B. in dem von den Studierenden gewünschte IntelliJ anzeigen kann. Mehrsprachigkeit: Der Entwurf der Software ist so konzipiert, dass eine Mehrsprachigkeit nachgerüstet werden kann. Die Sprache (sowohl die Benutzerführung als auch das Feedback betreffend) kann dann durch eine Einstellung im Client umgestellt werden. Die Spracheinstellung wird an angebundene Werkzeuge wie CheckStyle weitergereicht. Realisiert ist im Prototypen Englisch als Zielsprache der Programmierung und Modellierung, wobei zukünftig auch Deutsch als Lehrsprache der Informatik in Oldenburg ergänzt werden kann. Erweiterung um kollaborative Modellierung (CoMo): Außerdem konnte im Projektkontext die gemeinsame Modellierung durch mehrere Studierende in Echtzeit weiterentwickelt werden. Bisherige Aktivitäten für Diagramme im UML-Designer, der ebenso wie Papyrus auf Eclipse basiert, konnten erfolgreich zum Abschluss gebracht werden und die Übertragung dieser Ergebnisse auf Papyrus skizziert werden. Da sowohl das Feedback für Modellierungen in Einzelarbeit als auch die gemeinsame Kollaboration für

Modelle auf der gleichen technischen Grundlage aufsetzen, steht einer zukünftigen Integration von Feedback für Modelle in einer laufenden, gemeinsamen Modellierungssitzung nichts im Wege. Durch die Einbettung in Entwicklungsumgebungen können darüber hinaus bestehende Werkzeuge für die sequenzielle Kollaboration direkt weiter genutzt werden. Mittelfristig können hierdurch auch kollaborativ erstellte Lösungen von Modellierungsaufgaben in die Selbstlernumgebung übertragen werden.
Organisatorische Nachhaltigkeit: Da sich die entwickelte Software bewährt hat, wird sie einschließlich der entwickelten Aufgaben in weiteren Durchgängen der Module PDA (Programmierung, Datenstrukturen und Algorithmen), OMP (Objektorientierte Modellierung und Programmierung) und ST1 (Softwaretechnik I) genutzt werden. Die Selbstlernaufgaben sollen insbesondere die Klausurvorbereitung unterstützen, ergänzend zu den vorlesungsbegleitenden Aufgaben in den Tutorien. In den Folgejahren werden weitere Selbstlernaufgaben entwickelt und zur Verfügung gestellt werden. Darüber hinaus wird durch die Veröffentlichung des Sourcecodes eine Übertragung auf andere Module und andere Studiengänge (auch anderer Universitäten) ermöglicht. Der entwickelte Server ist unabhängig sowohl von der Lernplattform Stud.IP als auch von den konkret unterstützten Entwicklungswerkzeugen. Die Überprüfung, ob ein Nutzer in einer bestimmten Stud.IP-Veranstaltung eingetragen ist, wurde so gekapselt, dass sie leicht austauschbar ist.
Insgesamt ist die entwickelte Selbstlernplattform auf die Verwendung in anderen Veranstaltungen und die Nutzung an anderen Universitäten, auch mit anderen Campus-Managementsystemen, übertragbar. Das Datenmodell der Selbstlernplattform unterstützt beliebig viele Module, sodass auch hier die Übertragbarkeit und die Zukunftsfähigkeit gegeben sind. Außerdem wird die Weiterverbreitung der entwickelten Selbstlernplattform über OER/Twillo unterstützt. Über Twillo wird aus lizenzrechtlichen Gründen jedoch nur ein Verweis auf die Gitlab-Plattform der Universität Oldenburg bereitgestellt. Der Code-Zugang erfolgt dann über dies Open Source-Umgebung unter https://gitlab.uni-oldenburg.de/se/projects/innoplus/innoplus-prototype.

Im OER-Portal können Materialien, die im Rahmen des Projekts entstanden sind, hier heruntergeladen werden / Aus folgenden Gründen sind keine Materialien entstanden:

https://www.twillo.de/edu-sharing/components/collections?id=da3e1131-09d5-4624-b83e-4bcff43a8dc3 Aus lizenzrechtlichen Gründen verweist das hier hinterlegte Dokument auf alle im Projekt entstandenen frei verfügbaren Artefakte. https://uol.de/se?innoplus

Zielerreichung

Haben Sie die im Antrag beschriebenen Projektziele erreicht?

ja, die Erfolge waren sogar besser als erwartet:

Das Primärziel, nämlich die Bereitstellung einer Plattform zum Selbststudium, die Studierenden zeitnah Feedback zu Programmier- und Modellierungsaufgaben gibt, konnte im Projekt erreicht werden. Dafür wurden Plugins für das Entwicklungswerkzeug Eclipse entwickelt, mit denen direkt aus der Eclipse-Oberfläche Feedback für Programmier- und Modellierungsaufgaben angefordert werden kann. Die jeweiligen Lösungen werden automatisiert auf dem Server ausgewertet und Feedback in Form von Hinweisen auf Fehler

und Verbesserungsmöglichkeiten an Eclipse zurückgeschickt, die dort angezeigt werden. Durch diese Integration in die Entwicklungsumgebung kann auch beim Einsatz von Methoden wie Pair-Programming Feedback angefordert und angezeigt werden. Eine Anbindung an das Campus-Managementsystem Stud.IP erlaubt es, Nutzende der Selbstlernplattform existierenden Modulen zuzuordnen, so dass direkt die damit verknüpften relevanten Selbstlernaufgaben angezeigt werden können. Die Überprüfung der studentischen Lösungen erfolgt dennoch anonymisiert, so dass Studierende ohne Scheu auch unfertige oder fehlerhafte Lösungen einreichen können, um Feedback zu erhalten. Eingereichter Java-Programmcode wird hinsichtlich seiner Funktionalität durch die Ausführung von Akzeptanztests in Form von JUnit-Testfällen überprüft, die zusammen mit der Aufgabenstellung vom Dozenten entwickelt werden. Dabei erfolgt die Code-Ausführung in gesicherten und abgekapselten Containern (Sandboxen), um böswillige oder versehentliche Angriffe auf den Server zu verhindern. Neben den Testfällen wird der Code durch statische Anfragen überprüft. Damit sind Überprüfungen der Struktur und des konkreten algorithmischen Vorgehens der studentischen Lösungen möglich, sowie Prüfungen von Teillösungen, die zwar syntaktisch korrekt, aber nicht ausführbar sind. Dieser Anfrage-Mechanismus wurde im Rahmen der kostenneutralen Verlängerung entwickelt. Darüber hinaus wird die nicht-funktionale Qualität des Sourcecodes durch das serverseitig eingebundene Werkzeug CheckStyle überprüft, um beispielsweise auf Verstöße gegen Konventionen zur Benennung von Klassen und Methoden hinweisen zu können. CheckStyle wird außerdem genutzt, um ungeschickte oder ineffiziente Herangehensweisen zu erkennen und entsprechende Rückmeldungen an den Anwender zu geben. Auf diese Weise werden Programme nicht nur auf Korrektheit, sondern auch auf ihre Qualität überprüft. Dies ist ein wichtiger Aspekt der Programmierausbildung, der gerade beim Selbstlernen sonst oft vernachlässigt wird. Eingereichte UML-Diagramme werden, ebenfalls basierend auf einem Anfrage-Ansatz, auf Übereinstimmung mit einer von Lehrenden erstellten Musterlösung geprüft. Dadurch werden sowohl fehlende als auch überflüssige Teile studentischer Lösungen identifiziert. Zur Berücksichtigung unterschiedlicher Lösungsansätze können mehrere vollständige Musterlösungen durch die Lehrenden hinterlegt werden, mit denen sowohl leichte Unterschiede (bei z.B. Multiplizitäten) als auch größere Umstrukturierungen (z.B. unterschiedliche Auflösung von Mehrfachvererbung) berücksichtigt werden können. Zur Identifizierung von Synonymen und Übersetzungen bei Bezeichnern (z.B. Fahrrad, bike, bicycle) werden darüber hinaus verschiedensprachige Wörterbücher und Thesauri eingebunden, um beispielsweise sowohl deutsche als auch englische Bezeichner, sowie alternative Begriffe unterstützen zu können. Außerdem können darüber hinaus alternative Bezeichner (z.B. Pedelec) direkt durch die Lehrenden manuell in der Musterlösung hinterlegt werden. Dies funktioniert sowohl bei Programmcode als auch bei UML-Diagrammen. Die entwickelte Selbstlernplattform konnte erfolgreich evaluiert werden (Details siehe Frage 23), ermöglicht eine nachhaltige Weiterentwicklung und Übertragung (Details siehe Frage 18) und weist gleichzeitig auf weitere Ansätze und Forschungsideen zur besseren Nutzung in der Lehre hin (Details siehe Frage 25), für die dieses Projekt die konzeptionelle und technische Grundlage geschaffen hat, die durch den vorliegenden Prototyp validiert wurde. Ende März 2021 wurde das Projekt kostenneutral bis Ende Dezember 2021 verlängert, um verbliebene Gelder für studentische Hilfskräfte zur Härtung des bislang entwickelten Prototyps nutzen zu können. Die Evaluierung des Prototyps wurde in den Sommer- und Wintersemestern bis

Ende 2021 für die Programmierung und Modellierung fortgesetzt. Dazu wurde nicht nur die Funktionalität und Benutzbarkeit des Prototyps für Lehrende und Lernende verbessert, sondern insbesondere wurde auch der Pool an Selbstlernaufgaben erweitert.

Stellen Sie kurz Ihre eigenen Evaluationsergebnisse zum Projekt dar, insbesondere zur Zufriedenheit der Studierenden und Lehrenden:

Die entwickelte Selbstlernplattform wurde in mehreren Lehrveranstaltungen evaluiert, wobei die Anonymisierung und Usability wichtige Aspekte der Evaluation waren. Lehrveranstaltungen: Eine erste frühe Evaluierung im Modul OMP (Objektorientierte Modellierung und Programmierung) im Sommersemester 2020 zeigte, dass grundsätzlich ein Werkzeug für automatisches Feedback im Selbststudium als sinnvoll erachtet wird. Dies zeigte sich nicht nur in der aktiven Nutzung des Prototyps für die Programmierung, sondern auch durch entsprechendes Feedback in einer Befragung. Insbesondere die zeitliche Unabhängigkeit von Tutorien oder Großübungen für Feedback entsprechend der eigenen Lerngeschwindigkeit wurde als hilfreich bewertet. Nur teilweise wurde ein solches Werkzeug als nicht notwendig erachtet, was möglicherweise auch mit den noch eingeschränkten Überprüfungs- und Rückmeldemöglichkeiten des Prototyps zusammenhing. Ein weiterer, zu dem Zeitpunkt noch separater, Prototyp für die Modellierung wurde ähnlich positiv aufgenommen. Dass automatisches Feedback für Selbstlernaufgaben insbesondere den eigenen Lernfortschritt, losgelöst vom sonstigen strukturierten Modulablauf unterstützen kann, wurde in der Lehrveranstaltungsevaluierung im Modul PDA (Programmierung, Datenstrukturen und Algorithmen) im Wintersemester 2020/2021 bestätigt. Ein umfangreicherer Prototyp für die Programmierung mit insgesamt vier Aufgaben unterschiedlicher Schwierigkeitsgrade wurde evaluiert: Die Tutoren und Tutorinnen in PDA identifizierten neben kleineren technischen Fehlern insbesondere die fehlende Sortierung der angezeigten Rückmeldung nach Art oder Ort des auslösenden Programm- oder Diagrammfragments als Schwachstelle. Dies konnte in einer neuen Version der Plattform verbessert werden und wurde den Studierenden in PDA bereits zur Vorbereitung der Nachklausur zur Verfügung gestellt. Leider gab es nur wenige Rückmeldungen direkt über den in die Plattform integrierten Fragebogen, aber über externe Befragungen und über die Auswertung von Server-Protokollen konnte festgestellt werden, dass das Werkzeug ohne technische Probleme verwendet werden konnte und dass die bereitgestellten Aufgaben erfolgreich bearbeitet werden konnten. Insgesamt wurde das Werkzeug als hilfreich eingestuft, dessen Verwendung anderen grundsätzlich empfohlen und eine Weiterentwicklung als sinnvoll angesehen. Gewünscht wurden für die Programmierung zusätzlich zur Unterstützung von Eclipse auch analoge Plugins für die Entwicklungsumgebung IntelliJ. Weitere Evaluationen in Folgesemestern führten grundsätzlich zu ähnlichen Ergebnissen. Mit wachsendem Funktionsumfang und verbesserter Benutzbarkeit stieg die Akzeptanz bei den Studierenden ebenfalls weiter an. Allerdings gibt es nach wie vor Teilnehmer an den Modulen, die für die Plattform grundsätzlich keine sinnvolle Einsatzmöglichkeit sehen. Dies liegt vermutlich an der individuellen Herangehensweise an das Selbststudium, die sich zwischen den Studierenden stark unterscheiden kann. Die, während er kostenneutralen Verlängerung durchgeführte Evaluation bestätigten grundsätzlich die bereits vorliegenden Ergebnisse. Ein weiterer Aspekt wurde allerdings deutlich: Eine nicht unerhebliche Zahl von Studierenden löst die Aufgaben, ohne auf das

automatisierte Feedback oder sonstige Hilfestellungen zurückzugreifen. Es sind in erster Linie die Studierenden, die sich mit Programmier- oder Modellierungsaufgaben schwerer tun als andere, die auf die Hilfestellung durch das automatisierte Feedback zurückgreifen. Diese Erkenntnis deckt sich mit dem Ziel, „schwächere“ Studierende besonders zu unterstützen. Aufgrund der Anonymisierung der Evaluationsergebnisse und der Nutzungsprotokolle war es nicht möglich, einen Zusammenhang zwischen der Nutzung der Plattform und den Ergebnissen in den Prüfungen zu etablieren. Die Selbstlernaufgaben wurden aufbauend auf den definierten Lernzielen und den bisherigen Erfahrungen in den betroffenen Modulen maßgeschneidert entwickelt. Während der kostenneutralen Verlängerung wurde der Mechanismus zur Definition und Bereitstellung der Selbstlernaufgaben verbessert, was die Handhabung für die Lehrenden deutlich vereinfacht hat. Anonymisierung: Zur Pseudo-Anonymisierung wird eine Validierungs-API von Stud.IP genutzt, die für den per MD5-gehashten Nutzernamen (im Format „abcd1234“) angibt, ob er in einer Veranstaltung eingetragen ist. Dadurch können anonyme Statistiken über mehrere Anwendungen der Plattform hinweg erstellt werden. Das Hashen passiert direkt lokal beim Anwendenden im Client, so dass intern und insbesondere zur Übertragung auf den Server ausschließlich der Hash verwendet wird, von dem nicht auf konkrete Anwendende zurückgeschlossen werden kann. Diese Vorgehensweise zur Anonymisierung wurde den Nutzenden offengelegt. Beim Export der Statistiken werden außerdem als zusätzliche Sicherung diese Hashes durch zufällige, aber lokal eindeutige Werte ersetzt. Der zentrale Server und der Ort zum Herunterladen der Clients wurden auf Rechnern der Informatik bzw. der Abteilung Softwaretechnik der Universität Oldenburg bereitgestellt. Der in die Plattform integrierte Fragebogen wurde unabhängig vom Werkzeug über eine von der Universität Oldenburg zur Verfügung gestellte Plattform für Umfragen realisiert, sodass keine Rückschlüsse auf die Befragten möglich sind. Usability: Das Anfordern von Feedback ist in Eclipse nicht nur in der Java-Perspektive möglich, sondern auch in anderen, wie beispielsweise der Debugging-Perspektive. Dies erleichtert insbesondere Programmieranfängern die Nutzung. Feedback für Java wird nicht nur in einer dafür neu erstellten Ansicht aufgelistet, sondern auch zusätzlich direkt im überprüften Java-Code angezeigt. Dadurch sehen Nutzende direkt die relevante Stelle in ihrem Kontext. Diese Funktionalität ist ein wesentlicher Faktor für die gute Benutzbarkeit. Sie ist ein wesentlicher Grund dafür, dass die entwickelte Software direkt in Eclipse integriert ist, anstatt beispielsweise ein Hochladen der entwickelten Programme über das Stud.IP-System zu ermöglichen. Auch im UML Designer bzw. in Papyrus kann Feedback zu studentischen Lösungen direkt in der Modellierungsumgebung angezeigt werden.

Fazit: Beschreiben Sie die wichtigsten Erkenntnisse aus dem Projekt:

Das Projekt konnte erfolgreich zeigen, dass Selbstlernplattformen für Programmierung und Modellierung auf Basis statischer, dynamischer und Anfragebasierter Analysen in den Modulen im Grundstudium der Informatik und Wirtschaftsinformatik praktikabel eingesetzt werden können. Der entwickelte Prototyp ist anwendbar auf Java-Sourcecode und auf UML-Klassendiagramme und gibt hilfreiches Feedback für Studierende. Zu bemerken ist, dass das Werkzeug für den praktischen Einsatz weniger in die Lernplattform Stud.IP als vielmehr direkt in die Entwicklungsumgebung eingebunden werden muss, um das Anfordern von Feedback in den Bearbeitungsprozess der Aufgaben zu integrieren und auch das Feedback

direkt an den relevanten Stellen anzeigen zu können. Dies konnte erfolgreich im Prototyp für die Java-Programmierung und die UML-Modellierung in Eclipse realisiert werden. Zentral war ebenfalls die erweiterbare Architektur der Selbstlernplattform, um nachträglich leicht Ergänzungen vornehmen zu können. So konnten beispielsweise leicht CheckStyle für die Überprüfung der Codequalität und eine Anfrage-basierte Überprüfung von Struktur und Vorgehensweise integriert werden. Allerdings führt die Integration zusätzlicher Überprüfungen und zusätzlicher Rückmeldungen an die Studierenden zu einem erheblich größeren Aufwand bei der Erstellung der Prüfkriterien und Musterlösungen durch die Lehrenden, was bei der Erstellung der Aufgaben berücksichtigt werden muss. Außerdem konnte das Projekt erfolgreich in das forschungsorientierte Lernen integriert werden. Hier wurde insbesondere die kollaborative Modellierung eingesetzt, um gemeinsam mit Lehrenden und Studierenden neue Systeme zu entwerfen und Forschungsprototypen zu entwickeln. Einbettung des Projekts in die Lehre: Im Sinne des forschungsorientierten Lernens wurde das Projekt auch in die Lehre einbezogen, indem einige aus dem Projekt abgeleitete Fragestellungen in Abschlussarbeiten untersucht wurden bzw. werden: - Eine Masterarbeit motivierte den Einsatz von automatisierten Werkzeugen für das Selbst-studium, u.a. durch Umfragen unter Studierenden, und liefert Anfragen als Grundidee zur Überprüfung studentischer Lösungen (Kimberley Hebig: Modellierung im Selbststudium, Masterarbeit, Universität Oldenburg, 2020). - Eine Bachelorarbeit hat die Machbarkeit der Überprüfung von UML-Klassendiagrammen gezeigt, in der rudimentäre EMF-basierte Umsetzungsideen gewonnen werden konnten (Jan Hofmann: Graph-Anfragesprache für EMF, Masterarbeit, Universität Oldenburg, 2021). - Eine Bachelorarbeit lieferte Anregungen zur Integration von CheckStyle zur ergänzenden Überprüfung der Code-Qualität des eingereichten Programmcodes (Jana Meyer: Statische Code-Analyse statt Testen, Bachelorarbeit, Universität Oldenburg, 2020). - Eine Masterarbeit zielte auf eine domänenspezifische Sprache zur Formulierung von Musterlösungsvarianten, für die bisher nur eine eingeschränkte Ausdrucksmächtigkeit realisiert werden konnte (Johannes Legler: Automated Evaluation of Modeling Tasks for Self-study, Masterarbeit, Universität Oldenburg, 2021). - Eine Bachelorarbeit hat die Verwendbarkeit von sehr einfachen UML-Werkzeugen für die automatisierte Korrektur durch Transformation untersucht (Maik Sanders: Umwandlung von UMLet-Zeichnungen in UML-Diagramme, Bachelorarbeit, Universität Oldenburg, 2021). - Eine Bachelorarbeit hat unterschiedliche Ansätze zur Überprüfung von UML-Modellen betrachtet (Tim Clausing: Analyse von UML-Klassendiagrammen, Bachelorarbeit, Universität Oldenburg, 2021). - Eine Masterarbeit hat sich mit erweiterten Konzepten zur kollaborativen Modellierung befasst (Maik Appeldorn: Interoperable Echtzeitkollaboration durch Difference Language, Masterarbeit, Universität Oldenburg, 2022). - Eine Bachelorarbeit soll die Verwendung von Gamification-Ansätzen untersuchen, um die Kurz- und Langzeitmotivation der Studierenden zum Selbststudium zu erhöhen (to appear). Die Vielzahl interessanter Themen zeigt deutlich, dass das Projekt nicht nur der direkten Unterstützung der Lehre in den jeweiligen Modulen dient, sondern auch selbst interessante Forschungsfragen ermöglicht. Insgesamt unterstützt das Projekt die Lehre also doppelt. Konzeptioneller Ausblick: Darüber hinaus konnten im Projekt wertvolle Hinweise für zukünftige Erweiterungen gewonnen werden. Für ST1 sollen weitere UML-Diagrammtypen unterstützt werden, wie beispielsweise Aktivitätsdiagramme, was durch die geeignete Wahl eines entsprechenden Datenmodells bereits vorbereitet wurde. Für die

Modellierung hat sich herausgestellt, dass die Angabe von Varianten in Musterlösungen durch die Lehrenden erheblich ausgebaut werden muss. Wo der entwickelte Prototyp anhand konfigurierbarer Bezeichner die prinzipielle Machbarkeit gezeigt hat, muss in Folgeaktivitäten ein umfangreiches Variantenmanagement entwickelt werden, um den Lehrenden die einfache Angabe von alternativen Lösungen zu ermöglichen, die dann zu präziserem Feedback für die Studierenden führen. Hier bieten sich insbesondere weitere studentische Forschungsvorhaben zur Bearbeitung an. Auch die kollaborative Modellierung als Teil des Selbststudiums, nicht nur gemeinsam mit Lehrenden, wurde als wünschenswerter Baustein identifiziert. Diese Erweiterungen sollen in Folgeaktivitäten realisiert werden.