Ein Lehr- und Lernkonzept für die Softwareentwicklung im Team

Dennis Schmitz und Daniel Moldt
Universität Hamburg, Fachbereich Informatik
Vogt-Kölln-Str. 30
22527 Hamburg
schmitz@informatik.uni-hamburg.de
moldt@informatik.uni-hamburg.de

Abstract: Um beim Berufseinstieg erfolgreich als Informatiker wirken zu können, reicht es oft nicht aus nur separierte Kenntnisse über technische und theoretische Grundlagen, Programmiersprachen, Werkzeuge und Selbst- und Zeitmanagement zu besitzen. Vielmehr sollten Absolventen diese Kenntnisse praktisch miteinander verzahnt einsetzen können. An der Universität wird Studierenden leider selten die Möglichkeit geboten, diese verschiedenen Bereiche der Informatik miteinander integriert auszuüben. Dafür entwickeln wir seit über zwei Dekaden ein Lehr- und Lernkonzept zur Unterstützung praktischer Softwareentwicklungsveranstaltungen und setzen dieses um. Dadurch bieten wir angehenden SoftwareentwicklerInnen und ProjektmanagerInnen eine Umgebung, in der sie neues, praktisch relevantes Wissen erwerben können, sich selbst praktisch erproben und ihr Wissen konkret einsetzen können. Hier legen wir einen Schwerpunkt auf das Arbeiten im Team. Das hier vorgestellte Konzept kann auf ähnliche Lehrveranstaltungen übertragen und aufgrund seiner Modularisierung verändert und erweitert werden.

Keywords: Softwareentwicklung, Lehre, Teamarbeit, Projekte.

1 Einleitung

Eine Herausforderung an die universitäre Ausbildung besteht in der Einsetzbarkeit der Absolventen in Projekten in der Praxis ohne lange Einarbeitungsphasen. Vor dem Hintergrund der theoretischen, konzeptionellen Ausrichtung des Studiums erfordern Projekte praktisches Wissen und Erfahrungen in der

Anwendung von Konzepten, Techniken, Methoden und Werkzeugen. Somit sind organisatorische und persönliche Fähigkeiten bei der Interaktion mit anderen ein wichtiger Bestandteil der Ausbildung.

Bereits seit mehr als zwei Dekaden wird von unserem Arbeitsbereich systematisch an der Konzeption und Umsetzung eines Lehr- und Lernkonzepts in Praktika und Projekten gearbeitet. Neben den fachlichen Inhalten liegt der Schwerpunkt auf der Vermittlung einer ganzheitlichen Sicht auf ein Projekt. Zentraler Bestandteil ist, dass die Studierenden erlernen, wie sie gemeinsam an einer Aufgabenstellung arbeiten. Werkzeuge (wie Versionsverwaltung, integrierte Entwicklungsumgebungen, Continuous Integration, Testumgebungen, Kommunikations- und Projektmanagementwerkzeuge etc.) werden systematisch eingesetzt, um das Team bei der kollaborativen Arbeit zu unterstützen.

Als Lehrveranstaltungsformen werden Praktika und Projekte adressiert (4.–6. B. Sc. und 1.–3. M. Sc. Semester). Im Folgenden fassen wir die beiden Veranstaltungsformen unter dem Begriff *Lehrprojekt* zusammen. Dabei haben wir Erfahrungen mit Lehrprojektgrößen von sechs bis über 30 Teilnehmenden gesammelt.

Der in unserem Arbeitsbereich entwickelte Softwareentwicklungsansatz Paose (Petrinetz-, Agenten- und Organisationsorientierte Softwareentwicklung) basiert auf unseren Erfahrungen. In diesem Beitrag präsentieren wir didaktische Methoden und wie wir diese mittels Werkzeugeinsatz unterstützen. Nach der Benennung unserer erfahrenen Herausforderungen gehen wir detailliert auf die Durchführung des Lehrprojekts ein, diskutieren eingesetzte Lehr- und Lernkonzepte und erläutern deren Effekte.

2 Anforderungen, Inhalte und Herausforderungen

Um selbst eine geeignete Evaluation unserer Lehrprojekte vornehmen zu können, definieren wir zu Beginn Anforderungen, die wir im Laufe des Projekts erfüllen wollen. Daraus ergeben sich die Anforderungen, die von unserem hier präsentierten Lehr- und Lernkonzept berücksichtigt werden müssen. Die Anforderungen lassen sich grob in die folgenden Kategorien einteilen: (1) Anforderungen, die die Betreuenden selbst erfüllen müssen/wollen; (2) Anforderungen, die die Studierenden während und am Ende des Projekts erfüllen sollen; (3) Anforderungen, die die entwickelte Software erfüllen muss/soll. In diesem Beitrag konzentrieren wir uns auf die zweite Kategorie. Die erste und dritte werden hier lediglich exemplarisch umrissen.

Zu den Anforderungen der ersten Kategorie zählen z. B. ein neues Lehrkonzept, neue Konzepte, Techniken, Methoden oder Werkzeuge zu erproben sowie seine eigenen Projektmanagementfähigkeiten zu verbessern. Die Anforderungen der dritten Kategorie setzen sich aus den drei Gruppen konzeptioneller, anwendungsbezogener und technischer Anforderungen zusammen [Ba10]. Die ersten beiden Gruppen sind abhängig von dem Anwendungsgebiet der Software. Die Anforderungen der dritten Gruppe ergeben sich sowohl aus dem Anwendungsgebiet als auch aus der technischen Umgebung, die eingesetzt werden soll.

In unseren Lehrprojekten lehren wir u. a. die Paose. Die Paose ist ein umfassender Softwareentwicklungsansatz für die verteilte agentenorientierte Softwareentwicklung. Dieser beinhaltet Vorgaben für die Softwaretechnik, das Vorgehen und die Kollaboration der Softwareentwickler. Die grundlegende Programmiersprache in der Paose basiert auf *Java Referenznetzen* [Ku02], die mit der integrierten Entwicklungsumgebung (IDE) Renew [CHM16] modelliert werden. Außerdem ermöglicht Renew die Modellierung mit weiteren Techniken. Die erstellten Modelle werden zur Generierung von Programmcode und zur Konfiguration eingesetzt. Eine detaillierte Beschreibung der Paose lässt sich in [Ca10] finden.

Die zweite Kategorie der Anforderungen fokussiert auf die Studierenden. Die Studierenden sollen möglichst viele Facetten der Softwareentwicklung im Team kennenlernen. In unseren Projekten gehen wir auf die technischen, die sozialen und die Managementbereiche ein, welche wir im Folgenden beschreiben. Außerdem heben wir die – basierend auf unserer Erfahrung – prägnantesten Herausforderungen hervor.

2.1 Techniken und Werkzeuge

Die Studierenden lernen in unseren Projekten eine neue Programmiersprache (*Java Referenznetze*). Sonstige bei uns auftretende Auszeichnungs-, Programmier-, Skript- und Spezifikationssprachen sind *Java, JavaScript, Java Server Pages, HTML, CSS, Bash, XML* und *LaTeX*. Die am meisten eingesetzte IDE ist Renew, hinzukommt – je nach eigener Präferenz – Eclipse[1] oder Intellij IDEA[2].

1 https://www.eclipse.org/, zugegriffen am 17. 04. 2018
2 https://www.jetbrains.com/idea/, zugegriffen am 17. 04. 2018

Zusätzlich lernen die Studierenden ein – für die meisten – neues Softwareentwicklungsparadigma, die agentenorientierte Softwareentwicklung sowie unseren Softwareentwicklungsansatz Paose. In diesem werden Techniken eingesetzt, die vielen Studierenden unbekannt sind bzw. speziell für die Paose entwickelt wurden. Zu diesen zählen z. B. vier spezielle agentenorientierte, aber UML-ähnliche Diagramm- und Modelltypen, auf die hier jedoch nicht weiter eingegangen wird. Deren Beschreibung ist in [Ca10] zu finden.

Darüber hinaus setzen wir auch weit verbreitete Techniken und Werkzeuge ein. So verwenden wir die Linux-Distribution Ubuntu und die Kommandozeile (Bash). Letzteres stellt für viele Studierende eine besondere Herausforderung dar. Außerdem nutzen wir *Apache Ant*[3], *Git*[4], *Jenkins*[5] und *Redmine*[6]. Diese Bandbreite an Techniken und Werkzeugen zu erlernen und effektiv einzusetzen kann die Studierenden bereits zu Beginn der Projekte überfordern und abschrecken.

2.2 Soziales

Häufig kennen die Studierenden sich untereinander zuvor nicht und können somit weder die Fähigkeiten und den Kenntnisstand der anderen einschätzen noch wie diese sich in einem Team verhalten. Diese Aspekte sind für ein erfolgreiches Softwareentwicklungsprojekt jedoch entscheidend.

Wir betonen bereits zu Beginn eines jeden Projekts wie wichtig es ist, dass die lose Gruppe der Studierenden zu einem Team zusammenwächst. Um dies zu fördern, verdeutlichen wir den Studierenden, welche sozialen Ängste und Vorurteile in einer solchen Situation die Teambildung negativ beeinflussen können. Wir weisen darauf hin, dass die Studierenden zum Lernen an dem Projekt teilnehmen und nicht, um sich gegen andere zu behaupten. Die Zusammenarbeit mit erfahreneren Softwareentwicklern stellt somit eine Chance für sie dar und keine Herausforderung. Außerdem heben wir hervor, dass, wenn jemand Schwächen in der Programmierung aufzeigt, er durchaus Stärken in der Entwicklung kreativer Lösungen auf konzeptioneller Ebene haben kann. Wir warnen also davor, dass die Studierenden sich gegenüber zu voreingenommen sind.

---

3 https://ant.apache.org/, zugegriffen am 17. 04. 2018
4 https://git-scm.com/, zugegriffen am 17. 04. 2018
5 https://jenkins.io/, zugegriffen am 17. 04. 2018
6 https://www.redmine.org/, zugegriffen am 17. 04. 2018

Eine weitere Herausforderung stellen Studierende dar, die partiell wenig Vorwissen besitzen (Studierenden belegen in der Regel verschiedene Studiengänge) oder weniger begabt sind und somit bereits früh im Projekt drohen von den anderen Studierenden abgehängt zu werden. Meistens ist es möglich, dass diese Personen durch andere Studierende unterstützt werden, so dass die Unterstützenden sich selbst in die Rolle der Betreuenden begeben. Wir erwarten von unseren Studierenden nämlich nicht nur, dass sie sich selbst in das Team integrieren, sondern auch dabei mitwirken andere in das Team zu integrieren und diese bei aufkommenden Schwierigkeiten zu unterstützen.

2.3 Management und Kommunikation

In unseren Softwareentwicklungsprojekten setzen wir auf eine flache Hierarchie. Somit fordern wir von den Studierenden ein adäquates Aufgaben-, Selbst- und Zeitmanagement. Außerdem beinhalten unsere Projekte Phasen in denen räumlich, zeitlich und organisatorisch voneinander getrennt gearbeitet wird (verteilte Phasen).

Da die Studierenden meistens keine Erfahrung in der Softwareentwicklung im Team besitzen, fehlt ihnen häufig die Fähigkeit, die anstehenden Aufgaben sinnvoll zu priorisieren und zu parallelisieren. Besonders in den verteilten Phasen ist das Verständnis über die Abhängigkeiten und die Zuständigkeiten der Aufgaben wichtig. Andernfalls können durch eine unzureichende Priorisierung Situationen entstehen, in denen andere nicht mit ihrer Arbeit fortfahren können, weil ihnen z. B. notwendige Informationen fehlen. Sind die Zuständigkeiten von Aufgaben nicht ersichtlich, kann es zu der Mehrfachbearbeitung einer Aufgabe kommen oder die Studierenden wissen nicht, an wen sie sich mit einer konkreten Frage wenden sollen. Unsere Herausforderungen bestehen darin, ihnen diese Sachverhalte zu vermitteln, sie rechtzeitig auf entsprechende Situationen hinzuweisen und ihnen zu zeigen, wie die Verwendung von Projektmanagementwerkzeugen Abhilfe schaffen kann. Die organisatorisch-fachliche Aufteilung erfolgt anhand der Matrixorganisation von Paose [Ca07].

Des Weiteren sollen die Studierenden lernen ihre verrichtete Arbeit zu präsentieren, um die restlichen Teammitglieder über ihren aktuellen Fortschritt, aufkommende Probleme und kreative Ideen zu unterrichten. Wie wir die in diesem Abschnitt hervorgehobenen Herausforderungen angehen, beschreiben wir in den beiden folgenden Abschnitten.

3 Durchführung der Lehrprojekte

Der Arbeitsaufwand für die Studierenden in den Lehrveranstaltungen beläuft sich auf sechs (Praktikum) bzw. neun (Projekt) ECTS Punkte. Die verpflichtende Anwesenheitszeit für die Studierenden beträgt wöchentlich vier (ein Termin) bzw. sechs Stunden (zwei Termine). M. Sc. Projekte haben zusätzlich ein Seminar mit drei ECTS Punkten.

Die Struktur unserer Lehrprojekte besteht aus vier Phasen, wobei die Studierenden i. d. R. nur an den zwei mittleren Phasen mitwirken. Vor Semesterbeginn befinden sich die Betreuenden in der *Vorbereitungsphase*. Mit Semesterbeginn startet das Projekt in die *Einarbeitungsphase,* um im späteren Verlauf des Semesters in die *Softwareentwicklungsphase* überzugehen. Nach dessen Abschluss beginnen die Betreuenden mit der *Nachbereitungsphase*. Wir beschränken uns in diesem Beitrag auf die Einarbeitungsphase und die Softwareentwicklungsphase. Anschließend wird außerdem auf die Beurteilung der Leistungen der Studierenden eingegangen.

3.1 Die Einarbeitungsphase

In der Einarbeitungsphase bekommen die Studierenden die Möglichkeit sich mit den Techniken, den Werkzeugen, der agentenorientierten Softwareentwicklung und dem Softwareentwicklungsansatz Paose vertraut zu machen. Die Studierenden lernen sich in dieser Phase kennen, indem sie in unterschiedlichen Konstellationen miteinander Aufgaben lösen und sich gemeinsam neues Wissen erarbeiten. Dadurch wird eine gemeinsame Wissensbasis etabliert, auf der wir anschließend ein Softwareentwicklungsprojekt aufsetzen. Aus diesem Grund nutzen wir bereits in der Einarbeitungsphase Werkzeuge, die in der Softwareentwicklungsphase verwendet werden.

Da die Betreuenden in dieser Phase die zentralen Ansprechpersonen darstellen, bildet sich das Team i. d. R. um sie herum. Die Projektkonzeption erfordert die durchgehende Anwesenheit der Betreuenden. Bei großen Gruppen ist eine Unterstützung durch weitere Personen, die mit der Umgebung vertraut sind, hilfreich.

Aufgabenblätter

Die Basis dieser Phase bilden Aufgabenblätter, anhand derer die Studierenden sukzessive die Entwicklung von Multiagentensystemen (MAS) mit einem Agentenrahmenwerk erlernen. Die Aufgabenblätter bauen jeweils auf einander auf und führen Schritt für Schritt zu der Entwicklung einer verteilten webbasierten Chat-Applikation.

Den Zeitraum für die Bearbeitung eines Aufgabenblatts passen wir individuell auf die jeweilige Form der Lehrveranstaltung und auf die Fähigkeiten der Studierenden an. Somit wird ein Aufgabenblatt im Praktikum in ein bis zwei Wochen bearbeitet und im Projekt i. d. R. in einer Woche.

Damit die Studierenden seltener die Hilfe von Betreuenden benötigen und somit schneller und intensiver Unterstützung erhalten, haben wir die Aufgabenblätter überarbeitet. Wir haben die Prozesse, die zur Lösung der einzelnen Aufgaben führen, extrahiert und die Struktur der Aufgabenblätter nach dem Vorbild der Prozesse entworfen. Das Ergebnis sind Aufgabenblätter, die explizit das Vorgehen zur Lösung einer Aufgabe zeigen, jedoch nicht die Lösung selbst vorwegnehmen. Die Studierenden müssen die Aufgaben somit weiterhin selbst lösen, werden dabei jedoch in ihrem strukturierten Vorgehen unterstützt. Durch den prozessorientierten Charakter bieten die Aufgabenblätter die benötigten Informationen zu den richtigen Zeitpunkten an. Außerdem verweisen die Aufgabenblätter auf Informationen, die als Hilfestellungen zur Lösung der Aufgaben genutzt werden können und einzelne Themengebiete vertiefen. Diese weiterführenden Informationen stellen wir in der Form eines Wikis bereit. Einen tieferen Einblick in die Gestaltung der Aufgabenblätter bieten wir in [Sc16].

Bearbeitung der Aufgabenblätter

Während der Anwesenheitszeit setzen wir bei der Bearbeitung der Aufgabenblätter auf ein erweitertes Konzept des *Pair-Programmings*. Um zu vermeiden, dass einzelne Paare abgehängt werden und um das Kennenlernen zwischen den Studierenden zu unterstützen, mischen wir die Paare etwa alle 30 Minuten neu. Ein neu geformtes Paar setzt seine Arbeit bei dem jeweils geringeren Bearbeitungsstand fort. Somit ist gewährleistet, dass die Studierenden sich bereits während der Bearbeitung intensiv über ihre Lösungen austauschen und miteinander auseinandersetzen.

Für die Heimarbeitszeit lassen wir die Studierenden kleine Gruppen bilden, in denen sie den Rest der Aufgabenblätter lösen und ihre gemeinsamen Lösungen anschließend in der Form einer kurzen Präsentation aufbereiten.

Wir wählen für jede Aufgabe ein bis zwei Lösungen aus, die dann im Plenum besprochen werden. Dabei heben wir die Inhalte hervor, die durch die Bearbeitung des Aufgabenblatts vermittelt werden sollen und greifen Themen auf, die Schwierigkeiten bereitet haben.

Präsentationen

Die Aufgabenblätter werden von Präsentationen unterstützt, die die jeweilige Thematik zu Beginn eines Aufgabenblatts aufbereitet darstellen. Dadurch wissen die Studierenden, worauf das kommende Aufgabenblatt abzielt und sie können sich leichter Zusammenhänge erschließen. Während einige Präsentationen von den Betreuenden durchgeführt werden, ist jede(r) Studierende dazu verpflichtet selbst mind. eine Präsentation zu halten. Die Studierenden fokussieren in diesen Präsentationen gebräuchliche Werkzeuge und Vorgehensweisen in der Softwareentwicklung. Jede(r) Studierende arbeitet sich also in eine Thematik ein, wächst dadurch im Projekt für diese zu einem Experten heran und steht somit den Kommilitonen für Nachfragen zur Verfügung. Dadurch nehmen die Studierenden selbst in einem Teilbereich des Projekts die Rolle einer/eines Betreuenden ein.

Wiki

Als zentrales Wiki für unsere Lehrprojekte nutzen wir das integrierte Wiki von Redmine. In diesem beschreiben wir die einzelnen Aspekte, die für die Entwicklung von MAS mit dem verwendeten Agentenrahmenwerk essentiell sind und wie diese miteinander integriert werden. Außerdem sind alle Werkzeuge, die wir einsetzen, ihre Konfiguration und wie sie ineinandergreifen beschrieben. Darüber hinaus gibt es zu jedem Aufgabenblatt eine Wiki-Seite, die die relevanten Themen verlinkt.

Die Studierenden werden von uns dazu angehalten, ihnen fehlende Informationen selbstständig zu ergänzen. Dadurch entsteht eine gemeinsame Wissenssammlung, die Studierenden können die Lernumgebung selbst prägen und es entsteht ein Verantwortungsbewusstsein für den Lernerfolg aller derzeitigen und zukünftigen Projektteilnehmer.

Projektmanagement
Um die Studierenden bereits in dieser Phase an ein *Issue-System* zu gewöhnen, bilden wir für jede(n) Studierende(n) jedes Aufgabenblatt als eine hierarchische Sammlung von Tickets ab. Wir beschränken uns dabei auf folgende Granularität: Jede (Teil-)Aufgabe eines Aufgabenblatts ist ein Ticket. Die Tickets werden alle von uns inklusive einer Aufwandsschätzung erstellt. Die Studierenden müssen lediglich den Bearbeitungsstatus anpassen und ihre aufgewendete Zeit eintragen. Durch dieses Vorgehen erlernen die Studierenden den Umgang mit Tickets und können ihren Fortschritt im Vergleich mit ihren Kommilitonen selbst einschätzen.

Die Betreuenden erhalten so die Möglichkeit den Fortschritt der Studierenden auch außerhalb der Anwesenheitszeit einzusehen. Zusätzlich lassen sich die von den Studierenden gelieferten Ergebnisse in Relation zu der aufgewendeten Zeit setzen und so potentielle Schwierigkeiten identifizieren.

3.2 Die Softwareentwicklungsphase

Nachdem sich die Betreuenden und die Studierenden in der Einarbeitungsphase eine gemeinsame Wissensbasis geschaffen, sich auf Konventionen geeinigt sowie sich untereinander kennen gelernt haben, beginnt die Softwareentwicklungsphase. Damit die Studierenden sich im Laufe dieser Phase von ihrer eigenen Unsicherheit weniger gehemmt fühlen, machen wir ihnen bewusst, dass sie sich nicht in einer Prüfungssituation bzgl. ihrer softwaretechnischen Fähigkeiten befinden. Vielmehr bewerten wir in dieser Phase ihr Engagement, um die sozialen Kompetenzen der Studierenden zu stärken.

Die Betreuenden nehmen – neben ihren Softwareentwicklungsaufgaben, die jeweils im Paar mit einem Studierenden bearbeitet werden – in dieser Phase eine beratende und dirigierende Rolle ein. Sie stellen sicher, dass die Richtlinien des Softwareentwicklungsansatzes sowie Konventionen eingehalten werden. Zwar sollen die Studierenden eigenverantwortliches, selbstorganisiertes Arbeiten lernen, jedoch ist nicht zu unterschätzen, dass die meisten Studierenden unerfahren in der Softwareentwicklung im Team sind. Ohne ein zentrales Projektmanagement ist diese Phase nur selten von Erfolg gekrönt. Je nach Konstellation des Teams, können die Betreuenden ihre dirigierenden Aufgaben im Laufe dieser Phase auf die Studierenden verteilen. Die Studierenden nehmen somit außer der Rolle des Entwicklers auch organisatorische, konzeptionelle, beratende und lehrende Rollen ein.

Struktur
Wir folgen in der Softwareentwicklung zwar den mittlerweile gängigen Konzepten von Meilensteinen und Sprints, haben jedoch kein vollständiges Scrum implementiert. Zum Beispiel greifen wir nicht explizit auf alle Scrum-Rollen zurück. Zu Beginn eines Sprints definieren wir gemeinsam mit den Studierenden den neuen Meilenstein. Am Ende eines Sprints analysieren wir in der Retrospektive, welche Anforderungen erfüllt wurden und was die Gründe für das Nichterfüllen von Anforderungen waren. Wir verfolgen die gleiche Anwesenheitspflicht wie in der Einarbeitungsphase, um wenigstens ein bzw. zwei Mal in der Woche zentralisiert arbeiten zu können. Die restliche Zeit in der Woche wird meist räumlich und zeitlich verteilt gearbeitet. Wir haben die Erfahrung gemacht, dass längere verteilte Phasen besonders bei unerfahrenen Softwareentwicklern zu Problemen führen und das Projekt negativ beeinflussen.

Zu entwickelnde Softwareapplikation
In der Regel definieren die Betreuenden grobe Anforderungen an die zu entwickelnde Applikation. In den letzten Jahren haben wir dabei vermehrt auf die Unterstützung von Softwareentwicklern abgezielt, die in einem verteilten Team arbeiten. Dies bringt zwei Vorteile mit sich: die Studierenden kennen in etwa den Anwendungskontext und sie denken darüber nach, wie sie selbst von einer technischen Umgebung unterstützt werden wollen. Somit reflektieren sie ihre eigene Arbeitsweise im Team. Aus den Erfahrungen, die sie während der Entwicklung der Applikation machen, können sie außerdem weitere Anforderungen ableiten.

Kommunikation und Koordination
Unser grundlegendes Werkzeug für das Management des Softwareentwicklungsprojekts ist der Issue-Tracker von Redmine. Die Studierenden haben bereits in der Einarbeitungsphase gelernt, wie sie diesen dazu nutzen können, die aktuellen Tätigkeiten und den Fortschritt einzusehen. Außerdem wird die Entwicklung der einzelnen Komponenten und die Definition der Schnittstellen in dem in Redmine integrierten Wiki dokumentiert. Dadurch wird eine indirekte Kommunikation etabliert, die den Studierenden zeigen soll, wie wichtig diese Dokumentation ist. Studierende, die die Dokumentation vernachlässigen, erfahren, dass sie häufig auf gleiche Nachfragen reagieren müssen und somit in ihrer Arbeit unterbrochen werden.

Des Weiteren setzen wir auf eine klare Strukturierung der Applikation und leiten von dieser die Struktur des Softwareentwicklungsteams ab [Ca07]. Die Beteiligten werden jeweils einer oder mehreren Softwarekomponenten zugeordnet. Somit bilden sich kleine Gruppen, die jeweils für eine Softwarekomponente zuständig sind. Auf diese Weise bietet die strukturelle Übersicht der Applikation gleichzeitig die strukturelle Übersicht des Softwareentwicklungsteams. Die Interaktion zwischen Softwarekomponenten impliziert somit die Interaktion zwischen den einzelnen Gruppen, da diese gemeinsame Schnittstellen und auszutauschende Daten definieren müssen. Dies stärkt das Bewusstsein der Studierenden darüber, mit wem sie sich koordinieren müssen. Außerdem weisen wir die Studierenden darauf hin, zu Beginn eines Sprints die potentiellen Treffen mit anderen Gruppen zu koordinieren und auf diese Aufgaben ein besonderes Augenmerk zu legen.

Wir haben festgestellt, dass wir durch die Visualisierung dieser Beziehungen das Bewusstsein der Studierenden für die nötige Interaktion mit ihren Kommilitonen auf einfache Weise verstärken können. Dazu nutzen wir ein sog. *Coarse Design Diagram* (CDD, s. Abbildung 1). Da es die einzelnen Komponenten der zu entwickelnden Applikation darstellt (Agenten-Rollen werden als Akteure und Agenten-Interaktionen als Anwendungsfälle dargestellt), können die Studierenden die Schnittstellen ihrer Komponente sehen. Außerdem annotieren wir die einzelnen Komponenten mit den Namen der verantwortlichen Studierenden bzw. Betreuenden. Weitere Unterstützung für die Koordination von Teammitgliedern in verteilten Softwareentwicklungsprojekten ist in [Sc18] zu finden.

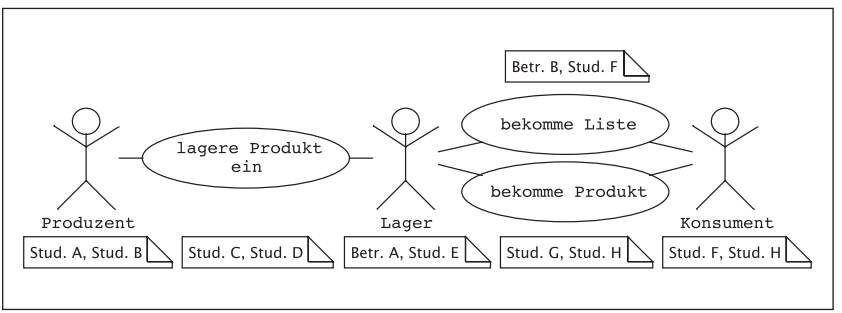

Abb. 1: Coarse Design Diagram (CDD) eines Produzent-Lager-Konsument-Beispiels

Retrospektive
Zusätzlich zu den applikationsbezogenen und softwaretechnischen Themen, nehmen wir uns entweder mit dem gesamten Team oder auch mit ausgewählten Teammitgliedern die Zeit, um Gründe zu erörtern, die zu einem reibungslosen Ablauf oder zu Schwierigkeiten geführt haben. Gemeinsam erarbeiten wir Lösungen, wie Schwierigkeiten in Zukunft vorgebeugt und reibungslose Abläufe hergestellt werden können. Diese Komponente bildet einen wichtigen Aspekt in dem iterativen Lernprozess der Studierenden während der Softwareentwicklungsphase und sollte nicht vernachlässigt werden.

3.3 Leistungsbeurteilung

Am Ende unserer Lehrprojekte wird keine einheitliche Prüfung durchgeführt. Hier gehen wir kurz darauf ein, wie wir die individuellen Leistungen der Studierenden beurteilen.

In der Einarbeitungsphase beurteilen wir die Studierenden danach, ob sie die Aufgabenblätter fristgerecht bearbeitet und die Ergebnisse in der geforderten Form eingereicht haben, sowie nach ihren eigenen Präsentationen. Da die Studierenden die einzelnen Aufgaben nicht alleine, sondern gemeinsam mit unterschiedlichen Studierenden bearbeiten, eignen sich die Aufgabenergebnisse schlecht als Bewertungsgrundlage. Wir beurteilen jedoch die individuelle Entwicklung der Studierenden und sprechen sie darauf an, wenn wir der Meinung sind, dass sie sich z. B. mit einem bestimmten Werkzeug intensiver beschäftigen sollten. Dies kontrollieren wir – mit Ankündigung – beim nächsten Termin.

Die Beurteilung in der Softwareentwicklungsphase ist schwieriger. Zwar lassen sich aus Redmine und Git normative Werte – wie z. B. die aufgewendete Zeit oder die Anzahl eingepflegter Code-Zeilen – extrahieren, jedoch lässt sich z. B. kein Wert für das aufgebrachte Engagement in der Integration ins Team und in Diskussionen aus diesen Werkzeugen gewinnen. Die Betreuenden stützen sich dafür auf ihre Erfahrungen und Beobachtungen, die sie regelmäßig untereinander diskutieren. Dies wird dadurch unterstützt, dass die Betreuenden Teil des Softwareentwicklungsteams sind, eng mit den Studierenden zusammenarbeiten und mit ihnen in kontinuierlichem Austausch stehen.

4 Diskussion der eingesetzten Lehr- und Lernkonzepte

Im Folgenden benennen wir konkrete Lehr- und Lernkonzepte bzgl. des vorherigen Abschnitts und diskutieren, warum wir diese Konzepte einsetzen. Diese Betrachtung dient außerdem als Grundlage für Lehrende, um ähnliche praktische Lehrprojekte zu gestalten.

Gemeinsame Einarbeitungsphase In unserem Kontext ist die gemeinsame Einarbeitungsphase bereits aufgrund der für die Studierenden unbekannten Paose und den zugehörigen Werkzeugen notwendig. Jedoch plädieren wir darauf, mit Studierenden auch in bekannteren Kontexten eine gemeinsame Einarbeitungsphase vor einem Softwareentwicklungsprojekt durchzuführen. Diese Phase dient besonders dem Kennenlernen, dem Einüben von Arbeitsprozessen und der Einigung auf Konventionen.

Kooperation beim Lernen Durch das Mischen der Paare in der Einarbeitungsphase wird die Kooperation beim Lernen unter den Studierenden gestärkt. Dies stärkt wiederum das Bewusstsein dafür, dass sie sich alle zusammen mit den Herausforderungen auseinandersetzen. Da der Fortschritt in einem neuen Paar meistens nicht übereinstimmt, ergibt sich daraus eine Situation, in der ein(e) weiter fortgeschrittene(r) Studierende(r) als Lehrende(r) auftritt und so das zuvor Gelernte festigen kann. Dies gleicht den Fortschritt der Studierenden an und verhindert das Abhängen einzelner. Das eingesetzte und gemeinsam gepflegte Wiki verstärkt den kooperativen Faktor beim Lernen zusätzlich. In der Softwareentwicklungsphase bilden alle Studierenden ein Team, welches gemeinsam eine Applikation entwickelt. Da wir die Studierenden auf die einzelnen Softwarekomponenten aufteilen, die nur in Zusammenarbeit die Funktionen der Applikation erfüllen, müssen die Studierenden miteinander kooperieren, um das Gesamtziel des Projekts zu erreichen. Dadurch sind die Studierenden stets dazu bereit, sich auch Schwierigkeiten bei der Entwicklung anderer Komponenten anzuhören und Lösungen zu entwickeln.

Expertenbildung Die Studierenden müssen sich in der Einarbeitungsphase in eine Technik oder ein Werkzeug einarbeiten und diese/dieses ihren Kommilitonen präsentieren. Im weiteren Verlauf des Lehrprojekts fungieren sie als Experten dafür und stehen ihren Kommilitonen als Ansprechpartner zur Verfügung. Dies spiegelt nicht nur die Gegebenheiten eines Projekts in der Wirtschaft wieder, sondern rückt die Studierenden in die Rolle eines Lehren-

den und gibt ihnen somit ein stärkeres Verantwortungsbewusstsein für den Erfolg des gesamten Projekts.

Arbeitsprozessorientiertes Lernen Ein gemeinsames Verständnis über das Vorgehen in der Softwareentwicklungsphase ist wichtig für eine erfolgreiche Zusammenarbeit. Deshalb legen wir bereits in der Einarbeitungsphase großen Wert darauf, dass die Studierenden einheitliche Arbeitsprozesse verinnerlichen. Dazu tragen zum einen die prozessorientierten Aufgabenblätter und zum anderen die Verwendung des Issue-Systems bei. Das arbeitsprozessorientierte Lernen führt außerdem dazu, dass die Studierenden die Werkzeuge nicht getrennt voneinander erlernen, sondern wie diese ineinandergreifen und somit ihren Platz in der sog. Tool-Chain kennen lernen.

Projektlernen Das Lernen im Rahmen eines konkreten Auftrags, der in der Form eines Projekts umgesetzt wird, ist in unseren Lehrprojekten inhärent. Die Anforderungen an das Projekt kommen dabei jedoch nicht von einem reellen Kunden, sondern von den Betreuenden und Studierenden. Diese Anforderungen beschränken sich nicht nur auf Funktionalitäten der Applikation, sondern beinhalten auch softwaretechnische, organisatorische und soziale Anforderungen. Dadurch, dass wir die Studierenden laufend auf kritische Aufgaben und Zeitpunkte hinweisen, lernen sie ihre Aufgaben sinnvoll zu priorisieren und welche Aufgaben voneinander abhängen bzw. parallelisiert werden können. Durch den verteilten Projektcharakter lernen sie außerdem wie bedeutend es ist, dass Zuständigkeiten und Entscheidungen explizit dokumentiert werden.

Lernen mit erfahrenen Softwareentwicklern Wir (als erfahrene Paose-Softwareentwickler) haben sehr gute Erfahrungen damit gemacht, gemeinsam mit den Studierenden eine Applikation zu entwickeln. Die Studierenden können Arbeitsweisen übernehmen, lernen Tricks, die man nur lernen kann, wenn man eng (z. B. im Paar) zusammenarbeitet, und haben jederzeit Ansprechpartner, die sich mit der in der Entwicklung befindlichen Applikation auskennen. Betreuende, die nicht an der Entwicklung teilnehmen, können zwar bei konzeptionellen und konkreten technischen Herausforderungen unterstützen, jedoch kaum Fragen beantworten, die auf Details der Applikation abzielen.

Retrospektive Die individuellen Gespräche zwischen Studierenden und Betreuenden, in denen Zeit für die Retrospektive eingeräumt wird, tragen

erheblich zum Lernerfolg der Studierenden bei. Es ist wichtig, dass in diesen Gesprächen keine Vorwürfe oder Schuldzuweisungen gemacht werden, sondern an das Verständnis für Entscheidungen appelliert wird und lediglich Vorschläge gemacht werden, wie bestimmte Situationen hätten besser gelöst werden können.

Diese Lehr- und Lernkonzepte können gemeinsam oder auch einzeln auf andere Lehrprojekte übertragen und in diesen eingesetzt werden. Von zentraler Bedeutung sind die Verankerung des Teamgedankens und das tiefgehende Verständnis der Betreuenden für nebenläufige, verteilte Abläufe in sozialen und organisatorischen Kontexten, die sich wechselseitig stark beeinflussen.

5 Schlussbetrachtung

Anhand der von uns veranstalteten praktischen Lehrprojekte in der Universität präsentieren wir in diesem Beitrag Herausforderungen und Lösungsvorschläge in der Form eines Lehr- und Lernkonzepts. Wir schildern, wie wir unsere Lehrprojekte strukturieren und die auftretenden Herausforderungen mit gezieltem Vorgehen und Werkzeugeinsatz lösen. In diesem Zuge bieten wir eine Übersicht einiger eingesetzter Lehr- und Lernkonzepte und beschreiben deren Effekte.

Wir versuchen unsere Lehrprojekte kontinuierlich zu verbessern und fokussieren die Entwicklung und den Einsatz weiterer Werkzeuge, die den Projektablauf ganzheitlichen unterstützen. Dazu zählt z. B. eine prozessorientierte Managementumgebung, die die Studierenden in der Priorisierung ihrer Aufgaben unterstützt, indem sie die Abhängigkeiten zwischen Studierenden auf einer detaillierteren Ebene visuell darstellt.

Literaturverzeichnis

[Ba10] Balzert, H.; Koschke, R.; Lämmel, U.; Liggesmeyer, P.; Quante, J.: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. Spektrum Verlag, 2010.

[Ca07] Cabac, L.: Multi-Agent System: A Guiding Metaphor for the Organization of Software Development Projects. In (Petta, P. Ed.): MATES 2007. Band 4687 in LNCS, Springer-Verlag, Leipzig, Germany, S. 1–12, 2007.

[Ca10] Cabac, L.: Modeling Petri Net-Based Multi-Agent Applications. Logos Verlag, Berlin, 2010.

[CHM16] Cabac, L.; Haustermann, M.; Mosteller, D.: Renew 2.5 – Towards a Comprehensive Integrated Development Environment for Petri Net-Based Applications. In (Kordon, F.; Moldt, D. Eds.): PETRI NETS 2016. Proceedings. Band 9698 in LNCS. Springer-Verlag, S. 101–112, 2016.

[Ku02] Kummer, O.: Referenznetze. Logos Verlag, Berlin, 2002.

[Sc16] Schmitz, D.; Moldt, D.; Cabac, L.; Mosteller, D.; Haustermann, M.: Utilizing Petri Nets for Teaching in Practical Courses on Collaborative Software Engineering. In Desel, J.; Yakovlev, A. Ed.): ACSD 2016. IEEE Computer Society, S. 74–83, 2016.

[Sc18] Schmitz, D.; Moldt, D.; Haustermann, M.; Mosteller, D.; Röder, C.: Team Coordination Based on Causal Nets with Synchronous Channels. In (Chatain, T.; Grosu, R. Eds.): ACSD 2018, IEEE Computer Society, S. 60–69, 2018.