die hochschullehre
Interdisziplinäre Zeitschrift für Studium und Lehre
Jahrgang 2, Nr. 2, 2016
www.hochschullehre.org

Dagmar Monett & Björn Kiehne

Interdisziplinäres Projektlernen in der agilen Softwareentwicklung

Zusammenfassung

Dieser Artikel präsentiert die neue Gestaltung bzw. Konzeption eines fachübergreifenden Labors zu agilen Software-Entwicklungsmethoden sowie ihre Planung, Durchführung und Evaluation als Informatik-Blockkurs. Das Lehrprojekt wurde über drei Jahre lang in der Fachrichtung Informatik des Fachbereichs Duales Studium an der Hochschule für Wirtschaft und Recht Berlin (HWR) durchgeführt. Nicht nur werden somit agile Softwareentwicklungstechniken von den Studierenden projektorientiert erlernt und erprobt, sondern die Lehre wird auch agil an die Bedürfnisse der Studierenden und der Lehrenden angepasst. Die Ergebnisse der Durchführung des Lehrprojekts zeigen, dass das projektorientierte Lernen in einer agilen und studierendenzentrierten Lehrumgebung vielfältige positive Chancen für Lernende und Lehrende mit sich bringt.

Schlüsselwörter

Projektorientiertes Lernen; Lernendenaktivierung; agile Lehre; agile Softwareentwicklung; Blockkurs

1 Einleitung

Befragt man den Duden nach der Bedeutung des Wortes agil, so gibt er folgende Umschreibungen an: betriebsam, beweglich, energiegeladen, geschäftig, geschickt, gewandt, lebhaft, quecksilbrig, rege, rührig, temperamentvoll, unruhig, vital, wendig. Genau das sind die Attribute, die agile Lehre (Chun, 2004; Berry, 2012; Briggs, 2014) ausmachen kann. All diese Qualitäten sind auch notwendig, um das Lernen im Kontext der agilen Softwareentwicklung erfolgreich zu machen.

Softwareentwicklung ist von Natur aus projektorientiert. Die agile Softwareentwicklung verlangt nach einer Herangehensweise, die multiperspektivisch und interdisziplinär ist, da auch die Produkte später von verschiedenen Gruppen genutzt werden. Damit bringt die agile Softwareentwicklung ganz natürlich Vorteile auch für das Lernen im Hochschulbereich mit sich, da es die Zusammenarbeit auf ein Produkt hin ausrichtet (Holcombe, Gheorghe & Macías, 2003; Hazzan & Dubinsky, 2007; Rico & Sayani, 2009). Durch die Übernahme von Elementen aus der realen Arbeitswelt in das Lernen an der Hochschule, können die Studierenden hier erleben, was es heißt, unter Zeitdruck miteinander zu einem Ergebnis zu kommen, das für den Kunden zufriedenstellend ist.

Der hier gewählte Zugang zum Lernen lehnt sich an das Projektlernen an (Frey, 2010; Apel & Knoll, 2001), das besonders durch seine Offenheit gekennzeichnet ist. Im Projektlernen geht es immer auch um „wirkliche" Anliegen und Fragen der Studierenden und es schafft damit eine Verbindung zu den eigenen Erfahrungen der Studierenden und der Arbeits- bzw. Lebenswelt. Die Studierenden werden aktiv – wenn man so möchte agil. Dabei richtet sich das Projektlernen klar auf ein Ziel aus – in diesem Fall auf die agile Entwicklung einer Software.

Das Lernen findet in Auseinandersetzung mit der Aufgabe statt, John Dewey (2002, Original 1910) spricht von „*reflective experience*" (etwa: „reflexive Erfahrung", „reflektierte Erfahrung"). Die Lernwiderstände, Verständnisprobleme und Herausforderungen der Zusammenarbeit werden zu Lernanreizen.

Im Zuge der Entwicklung eines fachübergreifenden Labors zu agilen Software-Entwicklungsmethoden wurden durch die Lehrende in Auseinandersetzung mit der eigenen Lehrveranstaltung folgende Fragen gestellt:

- Wie kann fachübergreifende Lehre im Erlernen von agiler Softwareentwicklung gestaltet werden/gelingen?
- Was ist der Gewinn dieses interdisziplinären Ansatzes?
- Was sind die Schwierigkeiten? Welche Tipps kann man Kolleginnen und Kollegengeben, die Ähnliches vorhaben?
- Was sind die Chancen und Risiken des Projektlernens vor dem Hintergrund der Erfahrungen mit dieser spezifischen Lehrveranstaltung?

Die vorliegende Arbeit[1] ist in folgende Abschnitte aufgeteilt: Zuerst werden die Grundlagen der konkreten Lehrveranstaltung beschrieben. Die Planung bzw. die Konzeption des Lehrprojekts erfolgt als vorbereitende Arbeit. Die Lehr- und Lernziele werden definiert sowie die Lehrdrehbücher zur Strukturierung der Lehre eingeführt. Anschließend wird die Durchführung des Lehrprojekts beschrieben und die konkrete Verwendung und Anpassung der Lehrdrehbücher untersucht.

Wie kann gemessen werden, ob die Studierenden die Lehrinhalte korrekt verstanden und eingesetzt haben? Entspricht die Qualität der Endsoftwareprodukte den Erwartungen? In welchen Bereichen kann das Lehrprojekt verbessert werden? Die Evaluation der Studierenden aber auch des Lehrprojekts ist Thema des darauf folgenden Abschnittes. Im Anschluss werden die Ergebnisse eines Fragebogens sowie der Evaluation der Studierenden präsentiert. Es folgt eine Diskussion und Bewertung der Auswertung der Ergebnisse. Der letzte Abschnitt enthält die wichtigsten Ergebnisse des Lehrprojekts. Sie bestätigen die Vorteile, die ein Informatik-Blockkurs für das Lehren sowie das Lernen mit sich mitbringt.

2 Das fachübergreifende Labor

Das fachübergreifende Labor (kurz FÜL) ist eine integrierte Lehrveranstaltung aus dem Modul IT4111 Projektmanagement im dualen Studiengang Informatik, Fachrichtung In-

1 Teil eines abschließenden Lehrprojekts zum *Berliner Zertifikat für Hochschullehre* des Berliner Zentrums für Hochschullehre (BZHL).

formatik, Studienbereich Technik der HWR. Es sind insgesamt 420 AE[2] für das ganze Modul vorgesehen, wobei das FÜL 44 AE umfasst.

Die Lehrveranstaltung findet derzeit im 3. Semester des Bachelorstudiengangs Informatik statt. Die Kursteilnehmenden haben Programmierkenntnisse aus vorherigen Kursen, sie haben aber bisher noch nicht in Teams gearbeitet. Die gesamte Gruppengröße liegt bei etwa 36 Studierenden.

Die Lehrform des FÜL ist eine Projektarbeit. Die Prüfungsleistung ist ein Programmentwurf, dessen Bewertung zu 30 Prozent in die Modulnote einfließt. Die Inhalte des FÜL können wie folgt zusammengefasst werden: Bearbeitung einer größeren fachübergreifenden Aufgabenstellung mit den Methoden des Projektmanagements und der IT[3]. Der Lehrende hat die Freiheit, die Aufgabenstellung und die Methoden des Projektmanagements und der IT auszuwählen.

2.1 Das Thema und die Techniken

Für das Projekt im FÜL werden ein fachübergreifendes, aktuelles Thema sowie agile Softwareentwicklungstechniken für die Realisierung empfohlen (Razmov & Anderson, 2006; Bleek & Wolf, 2008; Perera, 2009). Agile Techniken, wie die extreme Programmierung[4] (Wells, 2009), sind iterative Vorgehensweisen, die die Prozesse der Softwareentwicklung flexibler machen sollen. Schwerpunkte sind dabei die Menschen und Interaktionen, die funktionierende Software, die Zusammenarbeit mit den Kunden und das Eingehen auf Veränderungen, wie das Agile Manifesto (Beck et al., 2001) postuliert.

Als fachübergreifendes Thema eigneten sich in besonderem Maße die Implementierung und die graphische Umsetzung einer Metaheuristik für die kombinatorische Optimierung praktischer Anwendungen. Der ACO-Algorithmus[5] (Dorigo & Di Caro, 1999), basierend auf dem Verhalten realer Ameisen bei der Futtersuche, ist ein schönes Beispiel dafür. Mit ihm lassen sich sehr schnell sehr gute Lösungen von komplexen Problemen berechnen, wie die des wissenschaftlich populären Problems des Handlungsreisenden, TSP[6] (Dorigo & Gambardella, 1997), bei dem der kürzeste Weg über mehrere Orte zu finden ist. Querverbindungen zu anderen Lehrmodulen konnten somit hergestellt werden: Das TSP-Problem wird in früheren Semestern eingeführt und verschiedene Optimierungsverfahren werden auch in anderen Modulen behandelt.

Auch herausfordernd und fachübergreifend erweist sich die agile Entwicklung von Software in Teams. Denn nicht nur funktionale, sondern auch nichtfunktionale Anforderungen von „echten“ Kunden sind zu erheben, zu dokumentieren und zu realisieren, wie z. B. Softwareprogramme für Kinder als Teil einer Kooperation von Hochschule und Grundschule.

2 AE: Arbeitseinheit, auch Unterrichtsstunde oder Unterrichtseinheit. 1 AE = 45 Minuten.

3 IT: Informationstechnologien.

4 auch als XP bekannt.

5 ACO: Ant Colony System.

6 TSP: Travelling Salesman Problem.

2.2 Die Aufgabenstellung

Die Aufgabenstellung für das Projekt im FÜL definiert die Hauptanforderungen an das Softwareprodukt, das von den Studierenden zu entwickeln ist. Zum Beispiel kann der ACO-Algorithmus anhand standardisierter TSP-Testinstanzen (d.h. bekannte Mengen von Städten und ihrer Positionen) analysiert, konzipiert, implementiert und getestet werden. Funktionalitäten, wie eine Visualisierung des Optimierungsverfahrens und das Anzeigen weiterer Statistiken von möglichst kurzen Touren zwischen den Städten, die der Handlungsreisende durchläuft, können dabei implementiert werden.

Eine weitere konkrete Aufgabenstellung, bei der externe Kund_innen einbezogen wurden, war die Implementierung eines Programms, womit Schüler_innen das bekannte Sudoku-Rätsel lernen und lösen können. Die Aufgabenstellung hatte die Besonderheit, dass ein tatsächliches „echtes" Problem bearbeitet und somit das Lernen durch Engagement (eng. „*Service Learning*") (Seifert & Zentner, 2010) angewendet wird: Es sollte ein Software-Programm mit graphischer Umsetzung entwickelt werden, das an der Kinder-Uni einer Grundschule verwendet werden sollte, um Schülern ab der 1. Klasse die Entwicklung von Software näherzubringen und sie für die Informatik zu begeistern. Die Kinder sollten anhand des Sudoku-Programms (kurz: SUD) mit Fragestellungen und Prinzipien der Softwareentwicklung spielerisch vertraut gemacht werden.

2.3 Warum eine neue Gestaltung des Kurses?

Die Arbeit an einem Kursprojekt ist sehr anspruchsvoll. Die wichtigsten Inhalte z. B. über agile Softwareentwicklung, XP, ACO und SUD müssen in den ersten Kursstunden vermittelt werden. Nur so haben die Studierenden alle theoretischen Grundlagen und Softwaretechniken zur Verfügung, bevor sie mit der konkreten Implementierung der Software beginnen. Die Erfahrungen aus vorherigen Jahren, in denen die Lehrveranstaltung wöchentlich während nur 4 AE stattgefunden hat, haben gezeigt, dass die Zeitspanne für die Entwicklung der Softwareprogramme und für die korrekte Umsetzung der agilen Techniken sehr knapp bemessen ist. Die fruchtbare Teamarbeit am Projekt überschritten die vorgesehenen Programmierzyklen, mehrere agile Techniken konnten nicht komplett umgesetzt werden, einige Projekte wurden am Ende des Kurses nicht fertig. Zudem litt die Konzentration der Studierenden, die Produktivität der Arbeitsgruppen, die Qualität des Endprodukts, die Zufriedenheit der Studierenden und die Anwendung von einigen didaktischen Unterrichtsmethoden unter den sehr kurzen Terminen.

Dennoch ist es sehr motivierend für die Studierenden, agil ein anwendungsbezogenes Optimierungsproblem lösen oder Software für Kinder implementieren zu können. Die Theorie dazu ist relativ einfach zu vermitteln und die agilen Techniken können dabei unkompliziert eingesetzt werden, sollten aber sorgfältig begleitet und im Rahmen des Kurses mit einem direkten Coaching unterstützt werden, was aber in vorigen Jahren nicht explizit eingeplant werden konnte.

All das waren Gründe für die neue Gestaltung bzw. Konzeption des agilen, fachübergreifenden Labors und damit auch für die Planung, Durchführung und Evaluation des neuen agilen Informatik-Blockkurses, der zehn Wochen lang projektbasiert stattfinden soll.

3 Das Lehrinnovationsprojekt

3.1 Lern- und Lehrziele des Lehrprojekts

Die Hauptlehr- und Lerninhalte des Lehrprojekts sind:

- Vermittlung der theoretischen Grundlagen (z. B. agile Software Entwicklungsmethoden, XP, ACO, Sudoku-Algorithmen).
- Umsetzung der Theorie mit begleitender Beratung bzw. Coaching (individuell, pro Arbeitsgruppe und pro Kurs).
- Bearbeitung eines Softwareprojekts in Teamarbeit.

Um diese Lehrinhalte zu vermitteln, die Planung des Kurses zu strukturieren und die Lernfortschritte der Studierenden zu evaluieren, werden in den nächsten Abschnitten allgemeine sowie detaillierte Lehr- und Lernziele des Kurses definiert.

3.1.1 Begründung der Auswahl von Lehr- und Lernzielen

Die Lernziele wurden mithilfe der Wissensdimensionen und der Lernzieltaxonomie formuliert. Diese sind im Abschlussbericht[7] eines BLK-Projekts[8] (Schobel & von Holdt, 2004) beschrieben worden. Sowohl die verschiedene Stufen des Lernens nach der Bloom'schen Lernzieltaxonomie, modifiziert nach Anderson und Krathwohl (Anderson & Krathwohl, 2000) als auch die Wissensdimensionen *Fakten- und konzeptuelles Wissen* (das „Was") ,*verfahrensorientiertes Wissen* (das „Wie") und die *kognitiven Lernzielkategorien* (Erinnern, Verstehen, Anwenden, Analysieren, Bewerten, Schaffen) haben dabei geholfen, Lernziele fürs Lehrprojekt zu beschreiben.

Eine konsequente Nutzung vorgeschlagener Aktivverben[9] nach Schobel & von Holdt (2004) für alle sechs Lernzielkategorien hat sich als sehr wertvoll erwiesen.

3.1.2 Grob- und Feinlehrziele des Lehrprojekts

Groblehrziele für die Blöcke B1 bis B4

Die Studierenden können nach Abschluss des Teilmoduls Fachübergreifendes Labor (d. h. des kompletten Kurses) …

G1: … die Grundlagen der Theorie über die agile Softwareentwicklung, über die extreme Programmierung und über Metaheuristiken (bzw. über Sudoku-Algorithmen) erklären, wiederaufrufen und anwenden.

G2: … verschiedene Algorithmen (wie ACO bzw. Sudoku-Methoden) analysieren, implementieren, testen und ausführen.

---

7 siehe Details ab Abschnitt 3.2.2, S. 17, BLK-Projekt „Entwicklung und Erprobung eines integrierten Leistungspunktesystems in der Weiterentwicklung modularisierter Studiengänge am Beispiel der Ingenieurwissenschaften".

8 BLK: Bund-Länder-Kommission für Bildungsplanung und Forschungsförderung.

9 Aus dem Anhang „Erläuterungen zur Beschreibung und Abstrahierung von intendierten Lernzielen" in Schobel & von Holdt, 2004.

G3: … kombinatorische Probleme (wie TSP bzw. Sudoku-Rätsel) lösen.

G4: … in Arbeitsgruppen Software nach den agilen Prinzipien entwickeln.

G5: … die entwickelte Software an einem Beispiel ausprobieren.

Feinlehrziele des ersten Blocks (Einführung)

Die Studierenden können nach Abschluss des ersten Blockes …

F1.1: … die Grundprinzipien der agilen Softwareentwicklung erklären.

F1.2: … den Lebenszyklus der extremen Programmierung erkennen und wiederholen.

F1.3: … die XP Werte, Regeln und Techniken beschreiben und klassifizieren.

F1.4: … die Theorie über Metaheuristiken und kombinatorische Probleme (bzw. Sudoku-Lösungsverfahren) erklären und zusammenfassen.

Diese Feinlehrziele konkretisieren das Groblehrziel **G1**, das die theoretischen Grundlagen des Kurses definiert.

Feinlehrziele des zweiten Blocks (1. XP-Iteration)

Die Studierenden können nach Abschluss des zweiten Blockes …

F2.1: … Softwareanforderungen in *Story Cards*[10] beschreiben, ihre Prioritäten einschätzen, die Umsetzung diskutieren und abstimmen sowie die bezogenen Aktivitäten für die erste Iteration planen.

F2.2: … in ihrer Arbeitsgruppen *Stand-Up Meetings*[11] durchführen und in Paaren Software programmieren.

F2.3: … in Arbeitsgruppen Regeln für die Arbeit mehrerer Teams in einem Raum formulieren, selbständig analysieren und präsentieren.

F2.4: … in Arbeitsgruppen selbständig Software analysieren, entwerfen, programmieren, testen, ausprobieren, präsentieren und abgeben.

Die Feinlehrziele **F2.1** und **F2.2** haben mit den agilen Prinzipien der Softwareentwicklung zu tun, die allgemein in den Grobzielen **G1** und **G4** erläutert werden. Das Feinlehrziel **F2.4** ist eine konkrete Spezifizierung der Groblehrziele **G4** und **G5**. Es ist auch eng mit den Groblehrzielen **G2** und **G3** verbunden.

Feinlehrziele des dritten Blocks (2. XP-Iteration)

Die Studierenden können nach Abschluss des dritten Blockes …

F3.1: … den ersten Prototyp der entwickelten Software präsentieren, durchführen, testen lassen und evaluieren.

F3.2: … neue Softwareanforderungen in *Story Cards* beschreiben, ihre Prioritäten einschätzen, die Umsetzung diskutieren und abstimmen sowie die bezogenen Aktivitäten für die zweite Iteration planen.

F3.3: … in Arbeitsgruppen selbständig Software mittels XP-Techniken und nach den neuen Anforderungen anpassen, optimieren und weiterentwickeln.

10 Auf *Story Cards* werden Software-Anforderungen formuliert.

11 Kurze tägliche Besprechung erfolgen in einem agilen Team.

Feinlehrziele des vierten Blocks (Vorträge)

Die Studierenden können nach Abschluss des vierten Blockes …

F4.1: … das Endsoftwareprodukt präsentieren, ausführen, testen lassen und „offiziell" abgeben.

F4.2: … die in den Projekten konkret angewandte Theorie und die eingesetzten agilen Software-Entwicklungstechniken beschreiben und mit der entwickelten Software in Beziehung setzen.

F4.3: … ihr eigenes entwickelten Projekt kritisch bewerten und auch von anderen Arbeitsgruppen bewerten lassen.

Die Feinlehrziele aller Blöcke wurden weiter verfeinert und als konkrete Aktivitäten in Lehrdrehbücher dargestellt. Einige Beispiele werden in den nächsten Abschnitten gezeigt.

3.2 Design des Kurses

Die Tabelle 1 zeigt den neuen Ablaufplan der Lehrveranstaltung in vier Blöcken und die dafür vorgesehene Zeit in AE. Der größte Anteil machen die Blöcke zwei und drei aus, in denen hauptsächlich die Gruppenarbeit und das Coaching im PC-Labor stattfinden sollen. Die Gruppenarbeit ist inkrementell konzipiert, d.h. die Studierenden haben im Verlauf des Kurses immer mehr Zeit zum Programmieren, da am Anfang der Lehrveranstaltung mehr Theorie vermittelt wird bzw. mehr Coaching notwendig ist. Die Lehre und das Lernen erfolgen projektbasiert.

Tab. 1: Ablaufplan der Lehrveranstaltung in Arbeitseinheiten.

<table>
<tr><th>Block 1</th><th colspan="2">Block 2</th><th colspan="2">Block 3</th><th>Block 4</th></tr>
<tr><td>Tag 1</td><td>Tag 2</td><td>Tag 3</td><td>Tag 4</td><td>Tag 5</td><td>Tag 6</td></tr>
<tr><td rowspan="2">6 AE</td><td>8 AE
(PC-Lab)</td><td>8 AE
(PC-Lab)</td><td>8 AE
(PC-Lab)</td><td>8 AE
(PC-Lab)</td><td rowspan="2">6 AE</td></tr>
<tr><td colspan="2">16 AE</td><td colspan="2">16 AE</td></tr>
<tr><td colspan="6">44 AE</td></tr>
</table>

Die Tabelle 2 zeigt eine ähnliche Verteilung der Blöcke, jetzt aber bezüglich der Entwicklung des Softwareprodukts in zwei Iterationen (oder Teamarbeitsphasen) und die dafür vorgesehenen Releases (d. h. zwei geplante Abgaben der entwickelten Softwareprototypen). Der zweite Block fängt mit der Planung der ersten Iteration an und endet am Tag des ersten Releases im dritten Block, dem wiederum eine neue Iteration vorangeht. So dauert eine ganze Iteration ca. vier Wochen. Am Ende bzw. im vierten Block sollen die Studierenden ihre Ergebnisse präsentieren.

Tab. 2: Ablaufplan der Lehrveranstaltung: Blöcke und Iterationen.

<table>
<tr><th>Block 1</th><th colspan="2">Block 2</th><th colspan="2">Block 3</th><th>Block 4</th></tr>
<tr><td>Tag 1</td><td>Tag 2</td><td>Tag 3</td><td>Tag 4</td><td>Tag 5</td><td>Tag 6</td></tr>
<tr><td rowspan="2">Organisatorisches

Intro I</td><td>Intro II
1. Planung</td><td rowspan="2"></td><td>1. Release
2. Planung</td><td rowspan="2"></td><td rowspan="2">Endvorträge

2. Release</td></tr>
<tr><td colspan="2">1. Iteration
(Teamarbeit)</td><td colspan="2">2. Iteration
(Teamarbeit)</td></tr>
</table>

In den folgenden Abschnitten wird die Planung der neuen Kurs-Blöcke dargestellt.

3.2.1 Teil I: Einführung

Die Einführung in die Theorie soll hauptsächlich in der ersten Kurswoche erfolgen. Im ersten Block (*B1*) bzw. am ersten Tag der Lehrveranstaltung wird eine Einführung in die Theorie agiler Softwareentwicklung und XP vermittelt. Außerdem werden die notwendigen anwendungsbezogenen Grundlagen über Metaheuristiken (wie ACO) und über die kombinatorische Optimierung für die Lösung eines realen Problems (wie TSP) beschrieben.

In dem Block sollen den Studierenden auch alle organisatorischen Einzelheiten über den Kurs, das XP-Projekt und die Evaluierung sowie über die begleitende eLearning-Plattform zur Unterstützung von Lernaktivitäten vermittelt werden. Insgesamt sind sechs AE für den Block *B1* vorgesehen (siehe Tabellen 1 und 2).

3.2.2 Teil II: „Das XP-Projekt“ in zwei Iterationen

Im zweiten und dritten Block (*B2* und *B3*) bzw. an den Tagen zwei bis fünf werden neue Inhalte (hauptsächlich über agile Methoden inkl. XP) vermittelt, direkt bezogen auf ihre praktischen Umsetzung im Kursprojekt. Als eine weitere Form der Verzahnung von Theorie und Praxis im dualen Studium wird in der Lehrveranstaltung simuliert, wie die Teamarbeit in einem Unternehmen (Zeitmanagement, Interaktion mit fiktiven oder realen Kunden, Meetings, Rahmenbedingungen usw.) funktionieren könnte.

Die Planung, Entwicklung und Fertigstellung des Softwareprodukts nehmen insgesamt zwei Monate in Anspruch – im Labor, aber auch als Selbststudium und Teamarbeit außerhalb der Hochschule. Die Tage 2 und 3 im Block *B2* sind für die erste Phase der ersten XP-Iteration und die Tage 4 und 5 im Block *B3* für die erste Phase der zweiten XP-Iteration gedacht. Insgesamt sind 32 AE (2 Blöcke je zwei Tage je acht AE) für die Planung der zwei Iterationen des XP-Projekts, die begleitende Arbeit an den Projekten und für das Release der zwei ersten Prototypen der Software im PC-Labor vorgesehen.

3.2.3 Teil III: Abschlusspräsentationen

Die Endvorträge finden in der letzten Kurswoche statt. Im vierten Block (*B4*) bzw. am letzten Tag der Lehrveranstaltung präsentieren die Studierenden ihre fertigen Softwareprodukte und ihre Erfahrungen in einer Reihe von Vorträgen, deren Struktur und Anforderungen rechtzeitig bekannt sind.

Insgesamt sind sechs AE für die Darstellungen inkl. anschließender Diskussionen und Feedback vorgesehen. Dazu wird ein Verlauf für die Präsentationen und die anschließenden Diskussionen – um eine wissenschaftliche Tagung zu simulieren – festgelegt. Eine Zusammenfassung des Kurses sowie Feedback zu den Vorträgen und eine Umfrage zur Evaluation des Lehrprojekts sind für die Zeit nach den Vorträgen geplant.

3.3 Lehrdrehbücher des Lehrprojekts

3.3.1 Konzeption der Lehrbücher

Ein Lehrdrehbuch ist eine konkrete Strukturierung der Lehre nach dem *Sandwich-Prinzip*. Dabei geht es um die Kombination der Vermittlung von Lehrinhalten (das passive Lernen, das *Einatmen*) mit Phasen, in denen die Studierenden am effektivsten üben oder wiedergeben können, was sie gerade gelernt haben (das aktive Lernen, das *Ausatmen*) (Döring, 2008). Die Lehr- und Lernprozesse können auf dieser Weise systematisch, methodisch und didaktisch gesteuert werden.

Ein allgemeines Schema eines Lehrdrehbuches lag als eigene Vorarbeit des Lehrprojekts bereits vor. Die Phasen von Ein- und Ausatmen sollten in 90-minütigen Zeitblöcken abwechselnd integriert werden. Neuer Stoff sollte in kurzen Teilblöcken vermittelt werden, um die Konzentration der Studierenden über längere Zeiträume in den Einatmen-Phasen unter Kontrolle zu halten. Die Stoffverarbeitung und die Interaktion mit den Studierenden sind für die aktiven Zeitblöcke geplant.

In der Lehrveranstaltung FÜL soll eine kontinuierliche und ständige Integration von XP-Konzepten und -Techniken in die Arbeitsweise der Studierenden erfolgen. Die Stoffverarbeitung und der Wissenstransfer sollen explizit in der Softwareentwicklung in Teams vorkommen. Der inkrementelle Unabhängigkeitsgrad der Teams ist zu erwarten. Die immer steigende Notwendigkeit, sich auf eine Programmieraufgabe oder auf eine konkrete Softwareimplementierung zu konzentrieren, speziell beim *Pair Programming*[12], stellt eine der Kompetenzen dar, die in der agilen Lehrveranstaltung verstärkt werden sollen (Astrachan, Duvall & Wallingford, 2001; Cliburn, 2003; Goldman, Kon & Silva, 2004; Xu & Rajlich, 2005).

Die Lehrdrehbücher sollen all diese Aspekte berücksichtigen. Sie sollen die direkte Betreuung und das Coaching der Lernenden in der Gruppe beinhalten. Sie sollen auch eine detaillierte Einteilung der Inhalte für alle Lernaktivitäten der Kursblöcke darstellen.

3.3.2 Beispiele von Lehrbüchern

Exemplarisch werden einige konkrete Lehrdrehbücher präsentiert, hauptsächlich für die Blöcke *B2* und *B3*, in denen die XP-Iterationen eingeführt, geplant und realisiert werden und auf die sich die Teamarbeit und das Coaching konzentrieren.

In der Tabelle 3 wird unter anderem das Feinlehrziel **F2.1** (siehe Abschnitt 3.1.2) mithilfe des Sandwich-Prinzips weiter verfeinert. Dabei werden konkrete Phasen von Ein- und Ausatmen abwechselnd konzipiert, um die Theorie und die Praxis zu agilen Methoden, die für die Planung des Kursprojekts wichtig sind, zu festigen. Die Art der Durchführung der Aktivitäten und die einzusetzenden Medien sind in Klammern eingegeben.

Als Ergebnis des Kursprojekts sollen die Studierenden ein Softwareprogramm abgeben, das sie Schritt für Schritt in Teams entwickeln haben. Dafür wird das Feinlehrziel **F2.4** auch mithilfe des Sandwich-Prinzips weiter verfeinert. Ein Lehrdrehbuch für die vierte Lehreinheit, *B2T3.4*, des Blocks *B2* am dritten Tag des Kurses wird erstellt.

12 Die Paarprogrammierung ist eine Technik der agilen Softwareentwicklung. Zwei Entwickler_innen codieren am gleichen Rechner.

Tab. 3: Lehrdrehbuch der ersten Lehreinheit, *B2T2.1*, des Blocks *B2* am zweiten Tag des Kurses.

90 Min.	**Einstieg**	5 Min.	**Einstieg** – *passive Einheit* Einladung, Begrüßung (oral) Lernziele, Agenda (oral, Tafel) Zeitverteilung für *B2-T2.1* (Flipchart)
	Arbeits-phase	20 Min.	**Stoffvermittlung A** – *passive Einheit* Motivation (oral) Ziele der Lehrveranstaltung (Folien) Inhalte der Theorie vermitteln: – Arten von Story Cards (Flipchart) – Informationen auf Story Cards (Tafel) – Planungsspiel (Flipchart, Tafel)
		3 Min.	**Zuruffrage** – *aktive Einheit* Beispiele sammeln (im Plenum)
		20 Min.	**Stoffvermittlung B** – *passive Einheit* Aufgabenstellung (Blätter verteilen) Ziele des XP-Projekts (Tafel) Erste Hauptanforderungen (Blatt, Tafel) Anforderungen fürs 1. Release (Blatt, Tafel)
		2 Min.	**Aufgabe erstellen** – *passive Einheit* Planungsspiel: Vorgehensweise (oral)
		35 Min.	**Wissenstransfer** – *aktive Einheit* Planungsspiel für 1. Iteration (Teamarbeit) – *Story Cards* produzieren – Prioritäten setzen – Mögliche Umsetzung diskutieren Coaching, individuelle Beratung (pro Team)
	Ausstieg	5 Min.	**Ausstieg** – *aktive/passive Einheit* Auf allgemeine Fragen eingehen (oral) Zusammenfassung (oral) Kurz über Agenda für Teil *B2-T2.2* (oral)

Mit dem ersten Prototyp des Softwareprodukts sollen die Studierenden diesen präsentieren, durchführen, testen lassen und evaluieren, wie es im Feinlehrziel **F3.1** vorgesehen ist. Die Tabelle 4 stellt eine mögliche Verfeinerung dieses Ziels dar. Gemeint sind dabei das erste Release des Programms und damit das Abschließen der ersten XP-Iteration in einer „geschlossenen" Diskussion innerhalb einer Arbeitsgruppe mit dem Lehrenden.

Die Anforderungen für die neue XP-Iteration sollen nun spezifiziert und weitere Vorgehensweise im Kursprojekt definiert werden. Das Feinlehrziel **F3.2** wird dadurch realisiert. Ein Lehrdrehbuch für die zweite Lehreinheit, *B3T4.2*, des Blocks *B3* am vierten Tag des Kurses wird erstellt.

Ähnlich dazu werden alle anderen Feinlehrziele verfeinert und in der Form von Lehrdrehbüchern geplant. Der folgende Abschnitt beschäftigt sich mit der Durchführung des Lehrprojekts und damit, wie die Lehrdrehbücher in den jeweiligen Lehreinheiten bzw. Kursblöcken eingesetzt wurden.

Tab. 4: Lehrdrehbuch der ersten Lehreinheit, *B3T4.1*, des Blocks *B3* am vierten Tag des Kurses.

130 Min.	**Einstieg**	2 Min.	**Einstieg** – *passive Einheit* Einladung, Begrüßung (oral) Lernziele, Agenda (oral, Tafel) Zeitverteilung für *B3-T4.1* (Flipchart)
	Arbeits-phase	120 Min.	**Evaluation** – *aktive Einheit* 1. Release und Diskussion (pro Team) – „geschlossener" Stand der Arbeit[13] – 20-minütige Vorstellung inkl. Fragen – Umsetzung Anforderungen 1. Release – Akzeptanz-Tests durch den Kunden – Anwendung von XP-Techniken
	Ausstieg	8 Min.	**Ausstieg** – *aktive/passive Einheit* Feedback über 1. Release (oral) Vergleich, Stand der Entwicklungen (oral) Reflexion (im Plenum) Zusammenfassung (oral) Kurz über Agenda für Teil *B3-T4.2* (oral)

3.4 Evaluation der Studierenden

Da das Kursprojekt im FÜL 30% der Modulnote entspricht, eignete sich eine adäquate Verteilung der Hauptthemen auf einer Skala von 1 bis 100 Leistungspunkte, die danach auf einer Skala von 1 bis 30 Punkte relativiert werden können. Folgende Kriterien wurden evaluiert:

- **1. Release:** Abgabe und Präsentation des ersten Softwareprototyps in einer „geschlossen" Diskussion: Gemeinsam mit dem Lehrenden wird in jeder Arbeitsgruppe der Prototyp anhand der formulierten Anforderungen evaluiert (10 Punkte).
- **2. Release:** Abgabe und Präsentation des Endsoftwareprototyps in einer „öffentlichen" Diskussion, d.h. alle Arbeitsgruppen im Rahmen einer Mini-Konferenz, bei der alle Studierenden vortragen und die Endsoftwareprodukte präsentiert werden. Die Evaluierung der Endvorträge berücksichtigt auch soziale Kompetenzen individuell und in Bezug auf die Teamarbeit der Vortragenden (20 Punkte).
- **Agile Techniken und Projektmanagement:** Dokumentation erstellter Anforderungen und nötiger Arbeiten am Projekt sowie deren Mitglieder, Meilensteine, Meetings, Phasen und Ergebnisse nach den agilen Kriterien (30 Punkte).
- **Lauffähiges Programm:** Softwareprodukt, das am Ende des Kurses entstehen soll. Es soll alle Anforderungen erfüllen. Die Arbeitsgruppen sollen eine lauffähige Version ohne Fehler einreichen. Die wichtigsten Funktionalitäten und ihre Funkti-

[13] Gruppen, die nicht evaluiert werden, können weiter an der Entwicklung arbeiten bzw. die Evaluation vorbereiten (davor) oder anschließend darüber reflektieren gemeinsam mit dem Lehrenden wird in jeder Arbeitsgruppe der Prototyp anhand der formulierten Anforderungen evaluiert.

onsweise sollen in der Abschlusspräsentation ausgeführt und evaluiert werden (20 Punkte).

- **Projektdokumentation:** Dokumentation zum Projekt in der Form eines wissenschaftlichen Papers von fünf bis sieben Seiten. Den Studierenden wird empfohlen, eine geeignete Vorlage für Beiträge für Informatik-Konferenzen, wie die *two-column IEEE style*[14], zu verwenden. Entsprechend werden auch die Papers evaluiert (20 Punkte).

Den Studierenden sind alle Kriterien bekannt, ebenso die Verteilung der Punkte und die Leistungen und Kompetenzen, die diesen Kriterien entsprechen. Außerdem wird den Studierenden ein Muster für die Inhalte der Endpräsentationen vorgegeben. Details und Hinweise für die Gestaltung des wissenschaftlichen Papers werden ebenso vorgeschlagen und während des Kurses mehrmals thematisiert. Die gesamte Evaluation ist damit für die Studierenden in hohem Maße transparent.

4 Durchführung des Lehrprojekts

Das Lehrprojekt wurde drei Jahre lang durchgeführt: beim Jahrgang Informatik-IT2010 (von Mitte August bis Mitte November 2011), beim Jahrgang InformatikIT2011 (von Mitte August bis Mitte November 2012) und beim Jahrgang Informatik-IT2012 (von Mitte August bis Mitte November 2013). Die Komposition der Kurse ist in der Tabelle 5 dargestellt. Dabei zeigen die ersten zwei Zeilen auch die Merkmale der Kurse (d.h. Angaben für die Jahre 2009 und 2010), bevor das Lehrprojekt gestartet wurde, da sie zum Vergleich in der anschließenden Diskussion benutzt werden.

Tab. 5: Merkmale der letzten fünf FÜL Kurse.

Jahr	Gruppen-größe	Anteil weiblich	Anzahl Teams	Ø Team-größe	wöch./ Block	Agile Iter.	Algo-rithm	Didaktik	Coaching
2009	30	1	6	5	w	3	ACO	–	–
2010	30	2	7	4,7	w	3	GA	–	–
2011	24	–	5	5	b	2	ACO	++	+
2012	28	2	7	4,1	b	2	ACO	++	++
2013	34	8	11	3,1	b	2	SUD	++	+++

In den Jahren 2009 und 2010 hatte die Lehrveranstaltung FÜL eine wöchentliche Frequenz („w“) und sie war für drei agile Iterationen, und damit für drei verschiedene Releases oder Zwischenabgaben des Softwareproduktes konzipiert. Das neue Lehrprojekt wurde als Block-Lehrveranstaltung („b“) durchgeführt und die Anzahl von Iterationen bzw. Releases wurde um eins reduziert.

Die zu implementierenden mathematischen Algorithmen waren ACO und GA[15] für TSP sowie fürs Generieren, Lösen und Verifizieren von Sudokus (im Jahr 2013). Vor dem Lehr-

14 siehe http://www.ieee.org/conferences_events/conferences/publishing/templates.html [Stand: 12.04.2015].

15 GA: genetischer Algorithmus, auch eine populationsbasierte Metaheuristik, wie ACO.

projekt (d.h. in den Jahren 2009 und 2010) wurden keine speziellen didaktischen Methoden umgesetzt, auch keine Team-orientierte Betreuung oder Coaching.

4.1 Konkrete Einsetzung und Anpassung der Lehrdrehbücher

Alle vorgesehenen Lehrdrehbücher[16] wurden konsequent und agil befolgt und eingesetzt. Die aktiven und passiven Lehr- und Lernaktivitäten wurde bei ihrer Ausführung zeitlich protokolliert, um später ähnliche oder zukünftige Aktivitäten anpassen zu können. Dafür wurde eine neue Spalte in den Lehrdrehbüchern mit Kommentaren zur besseren Kontrolle der Zeit eingeführt. Dadurch wurden Aktivitäten, die zum Beispiel 2011 bei der ersten Durchführung des Lehrprojekts wesentlich weniger bzw. mehr Zeit in Anspruch genommen haben, in ihrer Dauer erweitert oder gekürzt und so 2012 bzw. 2013 beim zweiten bzw. dritten Durchlauf des Lehrprojekts realistischer eingeplant.

Wie die Aktivitäten von den Studierenden aufgenommen wurden, wurde in einer eigenen Spalte der Lehrdrehbücher protokolliert, ebenso wurden die Probleme und schwierigen Situationen, die bei der Durchführung einiger Aufgaben aufgetreten sind, detailliert dokumentiert. Zum Beispiel zeigen sich einige Studierende von sämtlichen didaktischen Techniken nicht so begeistert. Sie möchten die Unterrichtsmethoden, die konkret mit der Lösung von Aufgaben oder dem Programmieren nichts zu tun haben, nicht wahrnehmen und vergleichen diese oft mit Aufgaben für „Schulkinder“. Bei der Einführung solcher Aufgaben wurden dann mehr die Ziele betont und die eingesetzten didaktischen Methoden begründet. Das alles konnte schwierige Situationen im zweiten und dritten Durchlauf des Lehrprojekts (ab 2012) voraussehen und sogar einige davon komplett vermeiden. Außerdem wurden einige Aufgaben komplett geändert oder auf andere Lehreinheiten verschoben.

4.2 Beratung und Coaching der Studierenden

In den Lehreinheiten, in denen mehr Zeit für die Programmierung und Entwicklung der Softwareprogramme eingeplant war, wurden einzelne Beratungs- und Coachingstunden explizit eingeführt. So hatte jeder Studierende, aber auch jedes Team, mehrere Gelegenheiten, über konkrete Situationen und Probleme bei der Entwicklung mit der Lehrenden zu diskutieren.

Um die Lernziele der Lehrveranstaltung zu evaluieren, wurde auch nach Feedback von den Studierenden während des Kurses gefragt. Dazu wurden gezielte Fragen vorbereitet und diese aus vier verschiedenen Rollen der Lehrenden formuliert: als Lernbegleiter, als Coach oder XP-Experte, als Kunde und als Notengeber_innen. Am Anfang wurde diese klare Differenzierung der Rollen nicht deutlich wahrgenommen, aber mit der Zeit haben sich die Studierenden besser auf die Beratungs- und Coachingstunden vorbereitet. Zum Fragenkatalog beim Coaching gehören u.a. folgende Fragen:

a. Ist die Aufgabenstellung klar? Wurde der Auftrag verstanden?
b. Gibt es bisher irgendwelche Schwierigkeiten (organisatorisch, inhaltlich)?
c. Wie läuft die Kommunikation bzw. die Interaktion im Team?
d. Ist die Anzahl von Teammitgliedern zu gering/zu groß/angemessen?

16 Ein Lehrdrehbuch umfasst je zwei AE.

e. Wie sehen Sie die Trennung der Softwareentwicklung in zwei bzw. drei agilen Iterationen (zu wenig/zu viel/angemessen)?
f. Ist eine komplette Realisierung des Projektes (oder der Planung, des Releases usw.) am Ende des Kurses realistisch?
g. Haben Sie rollenbezogene Fragen (an die Lernbegleiterin, an den Coach, an die Kundin/den Kunden, an den/die Notengeber/in)?

Es erfolgte eine Diskussion mit jedem Team, wo die Antwort auf diese Fragen thematisiert wurde. Zum Beispiel wurden nicht nur Vorschläge zur Änderungen oder Verbesserung der Arbeitsweise im Team, sondern auch zu organisatorischen Aspekten der Lehrveranstaltung gesammelt und mit den Studierenden diskutiert. Damit wurde nicht nur die Software agil entwickelt, sondern auch die Lehrveranstaltung agil gestaltet und so realistisch wie möglich an die Bedürfnisse der Studierenden und der Lehrenden angepasst. Das direkte Feedback von den Studierenden spielte eine große Rolle dabei, die gestufte, explizite Evaluation der Studierenden aber auch.

5 Evaluation des Lehrprojekts

Das Lehrprojekt wurde ebenfalls evaluiert. Unabhängig von der formalen Kursevaluation der Hochschule wird dazu ein spezieller Fragebogen genutzt, der seit dem Jahr 2009 am Ende jedes FÜL Kurses verteilt und anschließend dokumentiert wird. Der Fragebogen ist in vier verschiedene Bereiche aufgeteilt: (i) Kurs- und Projektanforderungen, (ii) Lehre, (iii) Lernen und (iv) Umsetzung agiler Methoden. Insgesamt 16 Fragen berücksichtigen die Erfüllung mehrerer Lernziele und dienen der Evaluation des Lehrprojekts.

Die Studierenden sind mit den Fachtermini der agilen Entwicklung und der Programmiersprachen in Englisch vertraut. So wird die Umfrage in Englisch formuliert, auch um eine spätere Vorstellung der Ergebnisse auf internationalen Tagungen oder Publikationen zu vereinfachen. Die Studierenden konnten dazu auch eine Gesamtauswertung des Kurses angeben, einschließlich der Frage, was ihnen am besten und am wenigsten gefiel, sowie weitere Anregungen und Kommentare.

5.1 Ergebnisse der Evaluation des Kurses und Auswertung

Die empirischen Daten mit den Ergebnissen der Umfrage zur Untersuchung verschiedener Aspekte der FÜL Lehrveranstaltung wurden über einen Zeitraum von fünf Jahren erhoben. In Monett (2013) werden relative sowie globale Ergebnisse bis zum Jahr 2012 ausführlich dokumentiert. Im Jahr 2013 haben sich sogar die Ergebnisse einiger Bereiche des Fragenkatalogs, wie das Lernen und die Umsetzung agiler Methoden, weiter verbessert.

Das Feedback der Studierenden war bei allen Themen des Fragenkatalogs sehr positiv. Die größten Gewinnerinnen nach der Durchführung des Lehrprojekts sind allerdings die agilen Techniken und XP, die deutliche, stetige Verbesserungen der durchschnittlichen Werte in den letzten drei Jahren (gegenüber den Werten aus den Jahren 2008 und 2009) kumuliert haben: Die meisten Studierenden im Jahr 2012 und 2013 meinen, dass sie während des Kurses sehr viel gelernt hätten, der Kurs sehr motivierend gewesen sei, sich ihre

Programmierkenntnisse und die Qualität des Codes mit XP jeweils sehr verbessert hätten und dass das Arbeiten im Team und die Anwendung von XP um einen motivierenden Algorithmus zu implementieren sehr wichtig seien.

Da die agile Softwareentwicklung das zentrale Thema der FÜL Lehrveranstaltung ist, sind die Ergebnisse des Lehrprojekts im Allgemeinen mehr als zufriedenstellend. Für die Studierenden im Jahr 2013 waren die Zusammenarbeit mit echten Kunden aus einer Grundschule sowie die Entwicklung von Software für Kinder eine große Herausforderung, die sie sehr erfolgreich, aber auch mit viel Spaß gemeistert haben. Mehrere positive Kommentare dazu konnten der Umfrage entnommen werden.

Die Studierenden hatten auch die Möglichkeit zu erwähnen, was sie überhaupt nicht mochten und welche Empfehlungen und Änderungen berücksichtigt werden sollten. Einige typischen Reaktionen waren: es ist zu viel Arbeit für zu wenig Credits; der Zeitdruck ist zu hoch. Mehr Zeit sollte auch für die Programmierung und die Teamarbeit im Labor eingeplant werden. Es sei schwierig, in einem Raum zugleich mit vielen anderen Teams zu arbeiten. Entsprechende Maßnahmen zur Verbesserung der Lernsituation konnten in folgenden Kursen ergriffen werden.

Die Gesamtbeurteilung des Kurses in den fünf Jahren ist wie folgt: über 80% aller Studierenden bewerten den Kurs als positiv oder sehr positiv.

5.2 Reflexion

Die meisten Studierenden wollten bei einigen didaktischen Aufgaben nicht mitmachen, die im Frontalunterricht in Informatik-Kursen nicht üblich sind. Wenn sie damit konfrontiert und nach Feedback während und am Ende der Kursblöcke gefragt wurden, bestätigte sich, dass die Studierenden bei Aufgaben, in denen nicht direkt an Programmierungstätigkeiten gearbeitet wird, das Gefühl hatten, ihre Zeit zu „verschwenden“. Sie konnten die potenziellen Vorteile, die z.B. die didaktischen Spiele oder das Diskutieren in Paaren fürs langfristige Lernen haben, nicht wahrnehmen. In den Jahren 2012 und 2013 und unter Berücksichtigung von schwierigen Situationen in der Lehre im Jahr 2011 wurden der Zweck, die Ziele und die Vorteile solcher Art von Aufgaben im Voraus erklärt. Unterstützung kam dabei von einer Fachberaterin. Das Arbeitsumfeld und die Beziehungen Lehrende–Studierende waren dadurch viel entspannter und produktiver.

Nach direktem Feedback wurde auch mehrmals gefragt und zwar im Frontalunterricht und während der Coaching-Einheiten mit jeder Gruppe. Nie traten dabei Probleme auf. Nie gab es negative oder neutrale Kommentare, ganz im Gegenteil: Die Lehrende hat die Anmerkungen oder Wünsche der Studierenden umgehend im Rahmen der agilen Anpassung umgesetzt, wie zum Beispiel die Dauer der Pausen, die Suche nach anderen freien Räumen für die Teamarbeit, die Wiederholung von bestimmten Inhalten oder Informationen, die weitere Erklärung agiler Techniken usw.

Die Leistungsauswertung im Allgemeinen und speziell die erreichten Leistungspunkte sprechen eine klare Sprache. Die gemittelten Leistungspunkte[17] aus allen fünf Jahren sind in der Tabelle 6 aufgelistet. Alles in allem waren die erreichten Leistungspunkte hervorra-

17 Jedes Team besteht in der Regel aus drei bis fünf Studierenden, die die gleichen Leistungspunkte erhalten.

gend. Sie waren schon sehr gut vor der Durchführung des Lehrprojekts (Jahre 2008 und 2009) und haben sich mit diesem weiter verbessert.

Tab. 6: Erreichte Leistungspunkte aus allen fünf FÜL Kursen.

Jahr	Ø Leistungspunkte (von max. 30)
2009	27,82
2010	26,79
2011	28,92
2012	29,43
2013	28,76

Alle Studierenden haben die erforderlichen Leistungspunkte erreicht und trotz mangelnder Beteiligung und schwieriger Situationen in der Lehre im Jahr 2011 waren die erreichten Leistungspunkte dennoch sehr gut (28,92 von 30 Punkten). Meistens konnten die verlorenen Punkte dem wissenschaftlichen Schreiben zugewiesen werden. Die entwickelten Software-Programme waren aber am Ende der Kurse erfolgreiche, lauffähige Produkte, die die festgelegten Anforderungen erfüllen und rechtzeitig fertig waren. Darüber hinaus wurden die meisten XP-Werte und agile Praktiken sehr gut verstanden und während der Projektrealisierung korrekt umgesetzt.

6 Fazit

In dieser Arbeit wurde ein neues Lehrprojekt dokumentiert, das einen Informatik-Kurs über agile Softwareentwicklung in einen Blockkurs verwandelt. Die Planung, die agile Durchführung sowie die Evaluation des Lehrprojekts wurden entsprechend thematisiert.

Die Gestaltung des Kurses als eine Block-Veranstaltung hat sich als Idee herausgestellt, um die Arbeit an einer Lösung für reelle Kundenanforderungen an ein Unternehmen realistischer zu simulieren. Der „duale“ Charakter des Studiums am Fachbereich sowie die praxisnahe Erfahrung der Studierenden spielten dabei eine große Rolle. Die Planung der agilen Iterationen, die verschiedenen Releases der Softwareversionen, die Programmierung der Funktionalitäten und die Endprogrammierung selbst konnten bei einer Aufteilung des Kurses in Blöcke besser durchgeführt werden.

Nicht weniger wichtig sind die konsequente Wahl und die präzise Anwendung passender, didaktischer Lehr- und Lernformen, die die Lernenden, aber auch die Lehrenden in einer konzentrierten Block-Veranstaltung besser unterstützen. Weil die Interaktionen zwischen den beteiligten Stakeholdern eine zentrale Rolle spielen, die Lehrenden einfacher auf Veränderungen reagieren können, die Projekte und die Teamarbeit fokussiert bleiben und die Zusammenarbeit mit den Kunden in fachübergreifenden Projekten ermöglicht wird, konnte die Lehre dadurch auch agil gestaltet werden.

Im Abschnitt 2.3 wurden verschiedene Gründe für die neue Gestaltung bzw. Konzeption des Kurses thematisiert. Im Vergleich zu den Jahren, wo keine Blockveranstaltung durchgeführt wurde, haben die Ergebnisse des vorgestellten Lehrprojekts nicht nur die subjektive Wahrnehmung der Lehrenden, sondern auch das Lernen der Studierenden positiver beeinflusst. Insgesamt sind folgende Ergebnisse erreicht worden:

- Das Engagement der Studierenden in Blockkursen ist höher. Die erreichten Leistungspunkte auch.

- Die Studierenden haben in Blockkursen mehr Zeit, sich an aktiven Lernaufgaben produktiver zu beteiligen.
- Die Studierenden haben in Blockkursen mehr Zeit, sich besser auf die Entwicklung von Software-Programmen zu konzentrieren.
- Durch die Teamarbeit in Blockveranstaltungen verbessert sich die Fähigkeit der Studierenden, Software zu planen, zu analysieren, zu erstellen und auszuwerten.
- Die Studierenden können effektiver und souveräner ihre Projektergebnisse präsentieren und sich in der Diskussion aktiver beteiligen, wenn sie Software mit einem begleitenden Coaching qualitativer entwickeln.
- Die Studierenden verwenden die agilen Praktiken in größerem Maße, wenn sie über längere Zeiten ohne Unterbrechung im Unterricht projektbezogen lernen.
- Die Studierenden nutzen die agilen Praktiken effizienter, wenn die Lehre auch agil angepasst wird.
- Die Formulierung von Grob- und Feinlehrzielen mit Hilfe von Lehrdrehbüchern unterstützt den Lehrenden bei der Konzeption und Organisation der Lehre und auch bei ihrer (agilen) Durchführung.
- Die Lehrdrehbücher und die didaktischen Methoden, die die Lehre unterstützen, können leichter angepasst werden, wenn diese in der Form von Blockkursen stattfindet, da mehr Zeit für alternative Methoden und dazu passende Lernaktivitäten zur Verfügung steht.

Die Kursprojekte und das Lehrprojekt im Allgemeinen waren sehr erfolgreich und lehrreich. Sogar im Laufe des letzten Durchlaufs des Lehrprojektes haben sich mehrere Studierende bereit erklärt, weiter an dem Projektthema zu arbeiten bzw. forschen zu wollen. So entstanden darauf bezogene Themen für Studienarbeiten und es begann kurz danach eine begleitende Betreuung.

Für die im ersten Abschnitt gestellten Fragen können folgende Schlüsse gezogen werden:

Das in diesem Artikel vorgestellte Lehrprojekt wurde für die agile Softwareentwicklung in Informatik-Kursen konzipiert, kann aber relativ einfach verallgemeinert bzw. in andere Bereiche übertragen werden. Das Erlernen agiler Softwareentwicklung kann mit der fachübergreifenden Lehre vor allem agil und in Block-Veranstaltungen sehr gut gelingen.

Als bedeutender Gewinn des eingeführten interdisziplinären Ansatzes ist die agile Lehre als Ansatz zu benennen. In Annährung an das Agile Manifesto (siehe Abschnitt 2.1) werden in der agilen Lehre verschiedene Werte höher eingeschätzt: (i) Ein starker Fokus liegt auf den Individuen (Studierende, Lehrende, Kund_innen sowie andere Stakeholder) und den Interaktionen, wobei (ii) das studierendenzentrierte Lernen und das Tiefenlernen, d.h. die aktive Auseinandersetzung mit konkreten Inhalten, um diese zu verstehen (Marton & Säljö, 1976), im Vordergrund stehen. Außerdem können die Lehrenden besser und schneller auf Veränderungen reagieren, wenn sie zum Beispiel die Lehrdrehbücher agil einsetzen und anpassen. (iii) In einem Blockkurs nach dem agilen Einsatz werden Endprodukte (wie Softwareprogramme im hier vorgestellten fachübergreifenden Lehrprojekt) produktiver und qualitativer produziert, da zufriedener Beteiligte komplexe Pro-

jekte einfacher und erfolgreicher bewältigen können. (iv) Die Team- und die Zusammenarbeit mit Kund_innen, wie in realen Projekten, sind auch von Vorteil und unterstützen das Service Learning, wenn z. B. Softwarelösungen für die Community in Zusammenarbeit mit Schulen realisiert werden.

Einige Schwierigkeiten sind allerdings zu nennen: (a) Zeit fürs Coaching (individuell und teamorientiert) muss explizit eingeplant und von den Lehrenden ständig beachtet werden. (b) Aktive sowie passive didaktische und pädagogische Methoden müssen im Voraus reflektiert und in den Lehrdrehbüchern eingeplant werden. (c) Die Zusammenarbeit mit „echten" Kund_innen kann eine große Herausforderung für alle Beteiligten darstellen. (d) Der Perspektivwechsel vom Inhalts- zum Studierendenfokus kann sich als schwer erweisen, sowohl für Lehrende als auch für Studierende.

Für Kolleginnen und Kollegen, die Ähnliches vorhaben, könnten folgende Tipps hilfreich sein:

- Eine frühzeitige Auseinandersetzung mit den Prinzipien des studierendenzentrierten und des projektorientierten Lernens bzw. mit dazu passenden Lehrtechniken ist zu empfehlen. Je besser der Lehrende sich vorbereitet, desto erfolgreicher kann die Lernendenaktivierung und damit das Lernen gelingen.
- Wenn man sich als Lehrende für ein Lehrkonzept wie das Projektlernen entscheidet, dann ist es sinnvoll, die Inhalte des Kurses schon bei der Konzeption danach auszuwählen. Das macht es einfacher, die einzelnen Lerneinheiten zu konzipieren.
- Je früher „echte" Kund_innen bzw. Projekte ausgesucht werden, desto besser kann das Service Learning im Unterricht integriert werden. Die/der Lehrende soll dabei eine besondere Rolle verkörpern und zwar als Koordinator/in zwischen den Studierenden und den Kund_innen sowie möglicherweise aber auch als Projektmanager/in oder/und Projektleiter/in.
- Eine agile Denkweise ist vor allem in der Lehre von Vorteil. Dies ist nicht auf einen Informatikkurs beschränkt, da die Prinzipien der agilen Lehre übertragbar sind. Die agile Gestaltung der Lehre und ihre agile Umsetzung sind empfehlenswert.
- Die Herangehensweise des agilen Lehrens kann auch auf andere Themen und Kontexte übertragen werden. Sie stellt eine Haltung dar, die es den Lehrenden erlaubt, flexibel auf die Bedürfnisse der Studierenden zu reagieren und verschiedene Fachdisziplinen zusammenzubringen. Sie stellt eine Schlüsselqualität für den beruflichen Alltag dar, die sich die Studierenden über das Lernen am Modell (Bandura, 1976) aneignen können.

Die Chancen des projektorientierten Lernens in einer agilen und studierendenzentrierten Lehrumgebung sind vielfältig, es bringt aber auch Risiken mit sich. Nichtsdestotrotz ist mit einem höheren Engagement der Studierenden zu rechnen, wenn sie projektorientiert lernen und in Teams in einer Block-Veranstaltung arbeiten, da sie das Tiefenlernen aktivieren und z. B. Kompetenzen, wie von Unternehmen verlangt, besser erwerben können. Die oder der Lehrende ist dabei nicht mehr das Zentrum des Unterrichts, sondern fungiert als Betreuer/in, Coach und Begleiter/in des Lernens, die/der agil auf Veränderungen (auch in der Lehre) reagieren kann.

Literatur

Anderson, L. W. & Krathwohl, D. R. (2000). *A taxonomy for learning, teaching, and assessing: a revision of Bloom's taxonomy of educational objectives* (1st ed.). Pearson.

Apel, H. J. & Knoll, M. (2001). *Aus Projekten lernen. Grundlegung und Anregungen*. München: Oldenbourg.

Astrachan, O., Duvall, R. C. & Wallingford, E. (2001). *Bringing Extreme Programming to the Classroom*. In proceedings of the XP Universe 2001. Raleigh, NC, USA: Addison-Wesley Professional.

Bandura, A. (1976). *Lernen am Modell: Ansätze zu einer sozial-kognitiven Lerntheorie*. (1. Auflage) Stuttgart: Klett.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., . . . Thomas, D. (2001). *Manifesto for Agile Software Development*. The Agile Alliance. Retrieved from http://agilemanifesto.org/

Bleek, W.-G. & Wolf, H. (2008). *Agile Softwareentwicklung: Werte, Konzepte und Methoden* (1st ed.). Heidelberg: Dpunkt Verlag.

Berry, M. (2012, May 16). *The case for agile pedagogy*. Retrieved from Teacher Network. *The Guardian. Retrieved from:* http://www.theguardian.com/teacher-network/teacher-blog/2012/may/16/agile-pedagogy-computer-programming-learning

Briggs, S. (2014, February 22). *Agile Based Learning: What Is It and How Can It Change Education?* Retrieved from informED: http://www.opencolleges.edu.au/informed/features/agile-based-learning-what-is-it-and-how-can-it-change-education/

Chun, H. W. (2004). The Agile Teaching/Learning Methodology and its e-Learning Platform. In W. Liu, Y. Shi, & L. Qing (Eds.) *Lecture Notes in Computer Science, Advances in Web-Based Learning* (Vol. 3143, pp. 11–18). Heidelberg: Springer Verlag.

Cliburn, D. C. (2003). Experiences with pair programming at a small college. *Journal of Computing Science in Colleges*, 19(1), 20–29.

Dewey, J. (2002). *Wie wir denken*. Mit einem Nachwort neu herausgegeben von R. Horlacher & J. Oelkers (Original 1910). Zürich: Verlag Pestalozzianum.

Döring, K. W. (2008). *Handbuch Lehren und Trainieren in der Weiterbildung*. Weinheim, Basel: Beltz Verlag.

Dorigo, M. & Di Caro, G. (1999). The Ant Colony Optimization Meta-Heuristic. In M. Dorigo & F. Glover (Eds.), *New Ideas in Optimization* (pp. 11–32). McGrawHill.

Dorigo, M. & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *BioSystems*, 43, 73–81.

Frey, K. (2010). *Die Projektmethode* (11. Auflage). Weinheim: Beltz.

Goldman, A., Kon, F. & Silva, P. J. S. (2004). Being Extreme in the Classroom: Experiences Teaching XP. *Journal of the Brazilian Computer Society*, 10(2), 5–21.

Hazzan, O. & Dubinsky, Y. (2007). Why software engineering programs should teach agile software development. *ACM SIGSOFT Software Engineering Notes*, 32(2), 1–3.

Holcombe, M., Gheorghe, M. & Macías, F. (2003). Teaching XP for Real: some initial observations and plans. In M. Marchesi, G. Succi, D. Wells & L. Williams (Eds.), *Extreme Programming Perspectives* (pp. 251-260). Addison-Wesley Professional.

Marton, F. & Säljö, R. (1976). On qualitative differences in learning – I: Outcomes and process. *British Journal of Educational Psychology*, 46, 4-11.

Monett, D. (2013). Agile Project-Based Teaching and Learning. In H. R. Arabnia, L. Deligiannidis & G. Jandieri (Eds.), *Proceedings of the 11th International Conference on Software Engineering Research and Practice, SERP'2013* (pp. 377–383). Las Vegas, NV, USA: CSREA Press.

Perera, G. (2009). Impact of using agile practice for student software projects in computer science education. *International Journal of Education and Development Using ICT*, 5(3), 85–100.

Razmov, V. & Anderson, R. J. (2006). Experiences with Agile Teaching in Project-Based Courses. In *Proceedings of the American Society for Engineering Education, ASEE Annual Conference & Exposition*. Chicago, IL, USA. Retrieved from https://peer.asee.org/experiences-with-agile-teaching-in-project-based-courses

Rico, D. F. & Sayani, H. H. (2009). Use of Agile Methods in Software Engineering Education. In Y. Dubinsky, T. Dyba, S. Adolph & A. Sidky (Eds.), *Proceedings of the Agile Development Conference, AGILE'2009* (pp. 174–179). Chicago, IL, USA.

Schobel, K. & von Holdt, U. (2004). *Arbeitspaket 2: Qualifikation und ihre Verifikation durch ein Leistungspunktesystem*. Universität Hannover. Retrieved from http://www2.tu-ilmenau.de/lps/hannover/Abschlussbericht_Hannover.pdf

Seifert, A. & Zentner, S. (2010). *Service-Learning – Lernen durch Engagement: Methode, Qualität, Beispiele und ausgewählte Schwerpunkte*. Eine Publikation des Netzwerks Lernen durch Engagement. Weinheim: Freudenberg Stiftung.

Wells, D. (2009). *Extreme Programming: A Gentle Introduction*. Retrieved from http://www.extremeprogramming.org/

Xu, S. & Rajlich, V. (2005). Pair Programming in Graduate Software Engineering Course Projects. In D. Budny & G. Bjedov (Eds.), *Proceedings of the 35th ASEE/IEEE Frontiers in Education Conference, ICSE'2005* (pp. 7–12). Indianapolis, IN, USA: IEEE Computer Society.

Autoren

Prof. Dr. Dagmar Monett. Hochschule für Wirtschaft und Recht Berlin (HWR Berlin), Fachbereich Duales Studium Wirtschaft Technik, Fachrichtung Informatik, Alt-Friedrichsfelde 60, 10315 Berlin; Email: Dagmar.Monett-Diaz@hwr-berlin.de

Björn Kiehne, M.A. Berliner Zentrum für Hochschullehre (BZHL), Fraunhoferstraße 33-36, 10587 Berlin; Email: bjoern.kiehne@tu-berlin.de

Zitiervorschlag: Monett, D. & Kiehne, B. (2016). Interdisziplinäres Projektlernen in der agilen Softwareentwicklung. *die hochschullehre*, Jahrgang 2/2016, online unter: www.hochschullehre.org