Lecture Engineering

Thomas Lehmann und Bettina Buth, HAW Hamburg
thomas.lehmann|bettina.buth@haw-hamburg.de

Zusammenfassung

Für die Überarbeitung des Moduls Software Engineering im Studiengang Mechatronik der HAW Hamburg wurde ein kompetenzorientierter Ansatz verwendet. Dabei wurden Anleihen an die Vorgehensweisen und Prinzipien des Softwareengineerings genutzt, um systematisch zunächst die Kompetenzen und dann daraus die Inhalte des Moduls zu entwickeln. In diesem Artikel berichten wir über die bisherigen Erfahrungen bei der Konzeption und der Durchführung des Moduls anhand eines Fallbeispiels.

Ausgangssituation

An der HAW Hamburg wird der Studiengang Mechatronik gemeinsam von den Bereichen Maschinenbau, Elektrotechnik, Informatik und Fahrzeugtechnik durchgeführt. Das Department Informatik betreut entsprechend den Informatik-Anteil des Studiengangs. Durch personelle Änderungen ging die Verantwortung für das Modul Software Engineering zum Wintersemester 2013/14 an uns über. Die Übernahme sollte dazu genutzt werden, es von Grund auf neu zu konzipieren. Die Mechatronik vereint Mechanik, Elektrotechnik und Informatik, sodass man hier Studierende mit einer geringeren Affinität zu Software antrifft als in den Informatik-Studiengängen. In diesem Zusammenhang scheint es sinnvoll, den angehenden Mechatronikern nicht Softwareengineering im engeren Sinne, sondern die Perspektive des systematischen Vorgehens aus System- und Softwaresicht nahe zu bringen. Eine direkte Übernahme des Moduls Software Engineering aus der (Technischen) Informatik war deshalb nicht angebracht, da dieses nur die Softwareseite beleuchtet und von anderen Vorkenntnissen ausgeht. Das Modul musste auf die anderen Vorkenntnisse in Programmieren und die andere Art von Studierenden in diesem Studiengang angepasst sein.

Das Modul kann aufbauen auf Vorkenntnisse aus zwei Semestern Programmierung in C/C++. Insgesamt entspricht der Kenntnisstand hier ca. 40-50% dessen was in den Informatik-Studiengängen der HAW in den ersten beiden Semestern vermittelt wird. Die planerischen Rahmenbedingungen des Fachs gehen von einer Vorlesung mit 3 Vorlesungsstunden und 1 Praktikumsstunde pro Woche aus, üblich erfolgt eine Blockung der Praktika in 4 Termine a 4 Stunden.

> "Wo der Überblick fehlt, ist das Grauen nicht weit."(Lotter (2005))

Mit der Entscheidung zur Neugestaltung ergab sich die Frage, ob es einen systematischen Ansatz zur fachlichen und didaktischen Ausgestaltung eines Moduls gibt? Aus der Diskussion innerhalb der Hochschule bietet sich hier ein kompetenzorientiertes Vorgehen an. Wie behalte ich den Überblick bei der Entwicklung des Moduls? Da es hier um eine Art Produktentwicklung geht, entstand die Idee, Ideen und Methoden des Softwareengineerings auf die Entwicklung des Moduls Software Engineering anzuwenden. Dazu wurden Äquivalenzen zwischen den Entwicklungsschritten im V-Modell und der Entwicklung eines Moduls identifiziert. Im Folgenden wird beschrieben welche Anleihen gemacht, wie diese im kompetenzorientiertem Umfeld eingesetzt und welche Erfahrungen dabei gesammelt wurden.

Erste Schritte

Das Requirements Engineering (vgl. z. B. Hammerschall u. Beneken (2013)) sammelt die Erwartungen und Anforderungen an ein Produkt in seinen Facetten ein und gibt Leitlinien zu dieser Anforderungserhebung. In einer Vision wird ein Leitbild für das Produkt beschrieben, welche durch die Stakeholder in seinem Kontext näher definiert wird. Die Anforderungen werden systematisch dokumentiert und in einen Entwurf umgesetzt. Mit diesen Kernideen sind wir in das Erstellen von Requirements für das Modul eingestiegen.

In einem Textdokument haben wir zunächst unsere Vorstellung über ein Softwareengineering für einen Mechatroniker, somit das Ziel des Moduls im Kontext des Curriculums, dokumentiert. Bei der Betrachtung der Stakeholder lag der Fokus auf dem Hochschulkontext, also Studierende, Professoren/Dozenten, Labormitarbeiter, Stundenplaner und Studiengangskoordinatoren. Externe Stakeholder wie Personalverantwortliche wurden im ersten Iterationsschritt der Anwendung dieses Vorgehens nicht betrachtet. Diese hätten wie bei der Studie zu erforderlichen Kompetenzen (vgl. Hummel (2013)) für die Auswahl der Kompetenzen mit hinzugezogen werden müssen. Hier wurde zum Ausprobieren der Vorgehensweise auf die eigenen Erfahrungen aus der Praxis zur Definition der Kompetenzen vertraut.

Die Hochschule gibt einen Rahmen für die Umsetzung vor. Dieser Rahmen aus Anzahl der Semesterwo-

chen, Semesterwochenstunden (3+1 SWS), Kohortengröße, verfügbaren Laborplätzen, Anwesenheitspflicht im Labor, typischen Gestaltung des Stundenplans, Verfügbarkeit von Labormitarbeitern, Laborinfrastruktur, Credit Points (5 CP) und ähnliche Rahmenbedingungen haben wir mit Architekturvorgaben gleichgesetzt. Auch wenn uns diese Rahmenbedingungen geläufig waren, haben wir sie zur Reflexion in unserem Dokument aufgeführt. Weiterhin liefern sie für Außenstehende (z. B. andere Departments/Studiengänge) Hinweise darauf, inwieweit unsere Umsetzungsideen übernommen werden können. Für den eigenen Überblick wurden auch Referenzen auf bestehendes Material aus anderen Modulen aufgeführt, sowie die Liste der Deliverables, d. h. Artefakte an Vorlesungsunterlagen, Übungsaufgaben, Laboraufgaben, Templates, usw. bis hin zur Klausur, erstellt. Eine Liste an Quellen wie Bücher oder Vod-/Podcasts ergänzt die Materialliste.

Requirements Engineering

Ziel eines Moduls ist, dass die Studierenden am Ende über einen Satz an (akademischen) Kompetenzen verfügen. Entsprechend wurden im Sinne von Requirements zu erreichende Kompetenzen formuliert. Es wurde hier das akademisches Kompetenzmodell nach Reis (2013) verwendet, welches keine weitere Unterteilung in fachliche, Methoden-, soziale und Selbstkompetenz (Böttcher u. a. (2011)) wie aus der Berufsbildungsforschung vorsieht. Die Kompetenzen wurden als Learning Outcomes (Biggs u. Tang (2007) oder Kennedy (2007)) formuliert, die ein Handlungsziel in einer komplexen Situation sichtbar machen. Im Sinne des Constructive Alignment (vgl. z. B. Biggs u. Tang (2007) oder Brabrand (2008)) lassen sich so leicht formative als auch summative Prüfungsfragen (Zürich (2008)) ableiten. Das Constructive Alignment sieht vor, dass die Intended Learning Outcomes mit den Prüfungen und die darauf vorbereitenden Lehreinheiten zueinander inhaltlich und auf den Niveaustufen (Bloom (1972)) kohärent sind. Es darf nicht erwartet werden, dass die Studierenden UML Modelle erstellen können, wenn in der Vorlesung nur die einzelnen Symbole durchgearbeitet werden und dann auf dieser Basis in der Prüfung Analyseaufgaben auf einem gegebenen Diagramm gestellt werden. Dieses wäre im Sinne des Constuctive Alignment inkohärent, alleine schon weil die drei Teile auf verschiedenen Niveaustufen liegen. Wird eine Modellierungskompetenz erwartet, dann ist eine Prüfungsaufgabe zur Modellierung zu stellen und diese Kompetenz (nicht die Musterlösung) im Rahmen der Lehreinheiten einzuüben. Das Design der Vorlesung arbeitet somit auf die Prüfung der Kompetenz hin.

Im Gegensatz zur Forderung von Oliver Reis wurden hier nicht nur Learning Outcomes im engeren akademischen Sinne formuliert, sondern auch Kompetenzen auf niedrigen Niveaustufen explizit aufgeführt. Es wurde so versucht, die zu erreichende Niveaustufe (Bloom (1972)) direkt in den verwendeten Verben abzubilden. Diese führten dann zu Unterkompetenzen analog zu Hierarchien von Requirements. Die Studierenden müssen zum Beispiel nicht in der Lage sein ein Lasten- oder Pflichtenheft zu erstellen; sie sollten aber die Bedeutung dieser Dokumente kennen. Dieses ”kennen” wurde durch die Unterkompetenzformulierung „Die Studierenden können den Unterschied zwischen einem Lasten- und einem Pflichtenheft erläutern.“[1] abgebildet. Im Gegensatz dazu wurde auf das Dokumentieren einzelner Requirements Wert gelegt, was sich im Rahmen des Themas Requirement Engineering in der Formulierung der Teilkompetenz „Die Studierenden können einzelne Requirements strukturiert dokumentieren und die Qualität der Dokumentation bewerten.“ widerspiegelt. Gerade die Bewertung der Qualität liegt auf einer hohen Niveaustufe. Dieses Vorgehen bei der Erstellung der Kompetenzliste führte zu Unterkompetenzen analog zu Hierarchien bei Requirements. Die Unterkompetenzen beschreiben hier detaillierter, was unter der übergeordneten Kompetenz verstanden wird oder führen auf, welcher Baustein zum Erlangen der Hauptkompetenz beitragen soll.

Beispiel aus dem Bereich Requirements Engineering:

Die Studierenden ...

K-10 ... können Kundenwünsche identifizieren und als Requirements-Katalog dokumentieren und bewerten.

K-10.1 ... können einzelne Requirements im Sinne von strukturiertem Text formulieren und dokumentieren.

K-10.2 ... können Requirements geeignet attribuieren (Prioritäten, Stakeholder, usw.).

K-10.3 ... können Abnahmekriterien (Abnahmetests) für Requirements entwerfen.

...

K-10.7 ... können die Qualität von Requirements beurteilen.

K-10.8 ... können strukturiert ein Requirements Review durchführen.

Für die Identifikation der erforderlichen Kompetenzen wurde wie folgt vorgegangen: Aus bestehenden Modulen der Informatik und der persönlichen Erfahrung des Software und Systems Engineering wurden die Hauptkompetenzen formuliert. Unterkompetenzen ergänzten diese oder beschrieben genauer, was die Hauptkompetenz eigentlich aussagt. Hierzu wurden auch Modulbeschreibungen anderer Hochschulen herangezogen. Weiterhin wurden Lehrbücher und Vorlesungsunterlagen gesichtet. Bei den interessanten Themenfeldern wurde immer die Frage gestellt, was für sichtbare Handlungen können die Studierende in diesem Themenfeld ausführen, bzw. was ist ein mög-

[1] Die abgeleitete Prüfungsfrage im Sinne des Constructive Alignment ist dann „Erläutern Sie die Unterschiede zwischen einem Lasten- und einem Pflichtenheft!“

liche Handlung, die sich aus dem Wissen erschließt? Die Antworten führten dann zur Formulierung der Kompetenz mit einem geeigneten Verb des sichtbaren Handelns[2].

Nichtfunktionale Anforderungen ergeben sich aus den planerischen Randbedingungen; dies bezieht sich zum einen auf den generellen Zuschnitt als Modul mit 3 Stunden Vorlesung und 1 Stunde Praktikum und die praktische Umsetzung wie oben beschrieben. Andererseits sind aber auch die Vorgaben der Prüfungsordnung bezüglich der Prüfungsform (Klausur) und die Prüfungsvoraussetzung durch das erfolgreiche Absolvieren aller Praktikumstermine und -aufgaben als einschränkende Randbedingungen für die didaktische Ausgestaltung des Moduls zu sehen.

Übergang zum Design

Beim Erarbeiten der Kompetenzen fielen automatisch bereits Zusatzinformationen an, die den Übergang zum Design, dem Ausgestalten von Vorlesung, Laborpraktika und Übungen, vereinfachen. Analog ergeben sich beim Erfassen von Requirements bereits Ideen zu möglichen Realisierungen oder Designs. Diese (Design-)Informationen wurden strukturiert in einem Mindmapping-Tool (hier FreeMind) erfasst. Zu jeder Kompetenz wurde die Motivation für die Aufnahme der Kompetenz, Verweise auf vorhandenes Material, Querverweise, Notizen, Ideen für mögliche Prüfungsaufgaben sowie Ideen zur Vermittlung der Kompetenz annotiert (vgl. Abbildung 1). Bei den Vermittlungsideen wurde neben Vorlesungsvorträgen auch schon Verweise auf Methoden der aktivierenden Lehre (Macke u. a. (2012) bzw. Dallmeier u. a. (2013)) im Rahmen des seminaristischen Unterrichts mit Aufgabenstellungen vermerkt. Es wurde trotz der scheinbar gleichmäßigen Struktur der Annotationen ein Mindmapping-Tool verwendet, da es wesentlich flexibler bei der Ausbildung von hierarchischen Strukturen ist, bzw. für die Aufnahme von Ideen besser geeignet ist als die strenge Tabellenstruktur eines Requirement Tools.

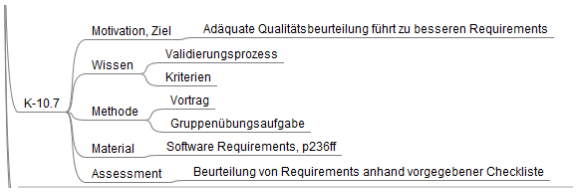


Abbildung 1: Zusatzinformationen zur Kompetenz 10.7 "... können die Qualität von Requirements beurteilen." im Mindmapping-Tool.

Im Sinne eines Architektur-Entwurfes wurde das Modul in thematischen Blöcke der beiden Lehrformen Vorlesung und Laborpraktikum aufgeteilt und grob der zeitliche Umfang pro Block festgelegt (Analogie: Assignment to Components). So wurden zum Beispiel für den Block „Requirements Engineering“ zwei Vorlesungstermine vorgesehen (siehe Tabelle 1).

In einer Matrix wurden dann die Kompetenzen den Blöcken zugeordnet. Es zeigt sich auch hier, dass einige Kompetenzen mehreren Vorlesungs- und Laborterminen zugeordnet werden können und müssen. Analog zum Verhalten, dass einige Requirements wie zum Beispiel Performance nur querschnittartig im System umgesetzt werden können, findet sich die Vermittlung einiger Kompetenzen als Querschnittsthemen wieder.

Die Zuordnung zu Vorlesungs- oder Laborterminen erfolgt je nach Kompetenz und erforderlichen Vermittlungs- als auch Prüfungsmethoden. Für die Kompetenz K-10.8 "Die Studierenden können strukturiert ein Requirements Review durchführen." ergibt sich durch das Verschieben des Verbs der Prüfungsauftrag "Führen Sie strukturiert ein Requirements Review durch." Durch die in der Prüfungsordnung vorgesehene Klausur (vgl. oben zu den nicht-funktionalen Anforderungen) lässt sich diese Kompetenz nur schlecht prüfen. Darüber hinaus sollten die zugehörigen Handlungen auch tatsächlich ausgeführt werden.

Wir hatten deshalb beschlossen, diese Kompetenz nicht im Sinne einer Notengebung zu prüfen. Diese Kompetenz wurde einem Labortermin zugeordnet, da hier Teams eine Aufgabe ausführen müssen und man als Dozent die Durchführung beobachten kann. Man kann die Teams bei der Ausführung beobachten und ihnen besser Feedback geben, als bei einer integrierten Übung in der Vorlesung. Weiterhin ist die Laborübung Teil der Prüfungsvorleistung und unterliegt somit auch einem Qualitätsanspruch. In den vorgelagerten Vorlesungen wird das Wissen um die Prozesse und die Qualitätsmerkmale von Reviews vermittelt. In dem eigentlichen Review im Praktikum sollen die Teams von vier Studierenden die Requirements eines anderen Teams des gleichen Labortermins überprüfen. Dies wiederum setzt voraus, dass vorher die Kompetenz zur Erstellung von Requirements vermittelt wurde. Aus dieser Abhängigkeit ergeben sich Randbedingungen für das Scheduling der Lehreinheiten und der Praktikumsinhalte.

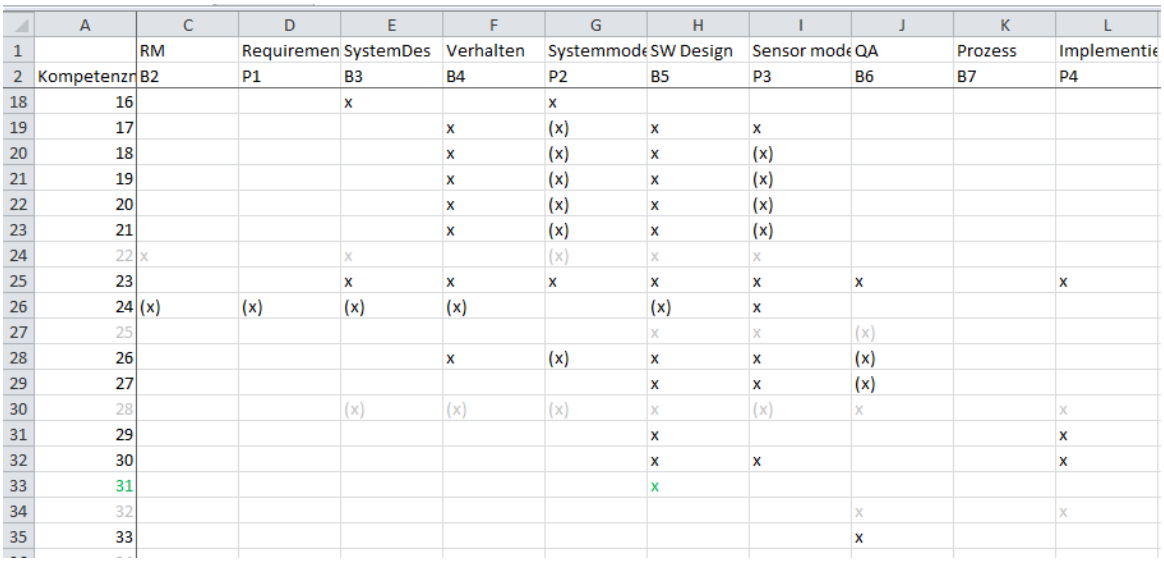
Insgesamt ergab sich für diesen Labortermin die Aufgabenstellung in das Erstellen von Requirements auf Basis einer Fallbeschreibung, welche eine kurzen Eingangsprüfung unterzogen wurden, und die Review-Sitzung unter Beobachtung mit abschließenden Feedbackrunden. Die Qualitätsbeurteilung und die überarbeiteten Requirements wurden dann geprüft und ergaben die Prüfungsvorleistung.

An diesem Punkt der Planung muss man abwägen, ob der Aufwand in Lehre und Prüfung zum Erreichen der jeweiligen Kompetenz gerechtfertigt ist. Falls nicht ist es erforderlich die geforderte Kompetenz anzupassen, da man sonst das Constructive Alignment nicht erzielt. Auch hier findet sich die Analogie zu den Aufgaben eines Softwarearchitekten wieder, der das Grob-

[2] Die Sichtbarkeit ist für die Ableitung einer guten Prüfungsaufgabe erforderlich.

Tabelle 1: Zuordnung Themenbereiche zu Vorlesungsterminen

Thema	Themenbereich	Termin(e)
Organisatorisches	Organisatorisches, Übersicht, Einführung	1
Requirements	Requirements, Kano-Modell, Stakeholder, Requirements Prozess, Dokumentation, Qualitätssicherung	1-2
Systemdesign	Systemdesign, Modellierung allgemein, SysML, Block Diagramms, Internal Block Diagrams, Flows, Systemsicht	3-4
Verhaltensmodelle	Verhaltensmodellierung, Sequence Diagrams, Activity Diagrams, State Diagrams	5-7
Software	Modellierung von Software, Datenmodelle, technische Systeme steuern, Umsetzung in lauffähige Software	7-9
QA	Qualitätssicherung	10-11
Prozesse	Prozesse/Configuration Management	11
Projektmanagement	Configmanagement, Ticketsysteme, Projektmanagement	12
Safety (Optional)	Prinziples, Begriffe, Analysemethoden, Techniken	12



	A	C	D	E	F	G	H	I	J	K	L
1		RM	Requiremen	SystemDes	Verhalten	Systemmode	SW Design	Sensor mode	QA	Prozess	Implementie
2	Kompetenzn	B2	P1	B3	B4	P2	B5	P3	B6	B7	P4
18	16			x		x					
19	17				x	(x)	x	x			
20	18				x	(x)	x	(x)			
21	19				x	(x)	x	(x)			
22	20				x	(x)	x	(x)			
23	21				x	(x)	x	(x)			
24	22	x		x		(x)	x	x			
25	23			x	x	x	x	x	x		x
26	24	(x)	(x)	(x)	(x)		(x)	x			
27	25						x	x	(x)		
28	26				x	(x)	x	x	(x)		
29	27						x	x	(x)		
30	28			(x)	(x)	(x)	x	(x)	x		x
31	29						x				x
32	30						x	x			x
33	31						x				
34	32								x		x
35	33								x		

Abbildung 2: Ausschnitt der Zuordnungsmatrix von Kompetenz zu Lehreinheiten.

design auf Machbarkeit im Rahmen der zur Verfügung stehenden Ressourcen prüfen und ggf. Maßnahmen ergreifen muss.

Implementierung

Auf Basis der Blockzuordnung der Kompetenzen und der Zusatzinformationen wurden die einzelnen Termine bezüglich ihrer Inhalte, Vermittlungsmethoden und zeitlichem Umfang geplant. Dabei wurde Material (Folien, Übungsaufgaben, usw.) aus bestehenden Vorlesungen übernommen, angepasst oder neu erstellt. Als ergänzendes Material wurden den Studierenden die Liste der zu erwerbenden Kompetenzen, Kontrollfragen und weiterführende Quellen für jeden Themenblock zur Verfügung gestellt. Ein Tracing der Kompetenzen auf einzelne Folien oder Übungen wurde als zu feingranular angesehen.

Bei der Umsetzung wurde bei den im Sinne des seminaristischen Unterrichts in die Vorlesungen integrierte Gruppenübungen darauf geachtet, dass in der Leistungsstärke gemischte, immer neu zusammengestellte Teams aktiviert wurden. Die Übungen sind großteils nach Ansätzen des Peer Facilitated Learning konzipiert, d. h. jeder Studierender skizziert zunächst eine individuelle Lösung, diese wird im zusammengestellten Team diskutiert und zu einem Teamergebnis zusammengeführt. Die Ergebnisse der Teams wurden dann jeweils auf einem ”Marktplatz” präsentiert und diskutiert. Gerade bei Modellierungsaufgaben erweist sich die Diskussion auf dem Marktplatz als günstige formative Lernkontrolle, da es oftmals keine eindeutige (Muster-)Lösung gibt.

Die Entwicklung der Laboraufgaben gestaltete sich schwieriger, da hier mehr Randbedingungen, insbesondere die verfügbaren oder beschaffbaren technische Aufbauten mit berücksichtigt werden müssen. Weiterhin darf die Problemstellung eine gewissen Komplexität weder über- noch unterschreiten, da die Problemstellungen alle Teammitglieder auslasten sollen.

Die Rahmenbedingungen des Departments sehen vier Laborterminen mit je 4 SWS vor. Dazu kommen Vor- und Nachbereitung. Typischerweise wird in unseren Modulen in Gruppen von 12-16 Studierenden in Zweierteams gearbeitet. Softwareengineering Methoden kommen aber erst dann zum Tragen, wenn eine Grundkomplexität überschritten wird. Somit wurde die Teamgröße im ersten Durchlauf auf 4-5 Studierende festgelegt und in nachfolgenden Durchläufen wieder auf 3-4 Studierende reduziert (vgl. Liebehenschel (2013)). In dieser Teamgröße können ausreichend komplexe Aufgaben gestellt werden, die in der Zeit bearbeitbar bleiben. Die Teamgröße erfordert eine andere Kommunikation als im eingespielten Zweierteam. Ein strukturiertes Vorgehen und ein Informationsaustausch über Diagramme (UML) wird nun erforderlich.

Die Labortermine eignen sich gut, die Studierenden beim Bearbeiten der Problemstellung und ihre Vorgehensweise zu beobachten. Deshalb wurden die Problemstellungen so ausgewählt, dass der Prozess im Vordergrund steht und das Ergebnis der Problembearbeitung durch nachzureichende Artefakte geprüft wurde. Im Einzelnen:

1. Labortermin: Strukturiertes Requirements Review

   Vorleistung: Kleines Requirements-Dokument erstellen

   Nacharbeiten: Einarbeiten der Mängelliste (erstellt durch anderes Team)

2. Labortermin: Modellierung eines Systems

   Vorleistung: Einarbeiten in die Diagrammtypen

   Nacharbeiten: Diagramme vollständig mit Kommentaren

3. Labortermin: Wartung Teil 1.

   Eine bestehende Steuerungssoftware soll erweitert werden. Aufarbeiten der Requirements und Analyse des Bestandssystems.

   Nacharbeiten: Dokumentation der Requirements und Analyseergebnisse

4. Labortermin: Wartung Teil 2.

   Design, Umsetzung und Test der Erweiterung.

   Nacharbeit: Dokumentation des Designs, der Umsetzung und der Abnahmetests

Gerade die Wartungsaufgabe erwies sich für die Studierenden als große Herausforderung, da sie sich in eine fremdes System eindenken müssen und sie mit dem üblichen Trial-and-Error nicht effektiv zum Ziel kommen. Trotz des Aufwandes und den Schwierigkeiten bei der Aufgabenstellung war das Feedback zu dieser Problemstellung sehr positiv, da es sich wie ein Problem aus dem realen Leben anfühlte.

Qualitätskontrolle

In der Qualitätskontrolle muss man zwischen der abschließenden Prüfung (weiter unten), Beurteilung des Lernfortschrittes im laufenden Semester und der Qualitätskontrolle in der Umsetzung unterscheiden.

Für die Beurteilung des Lernfortschrittes der Semestergruppe wurden die Diskussionen im seminaristischen Unterricht und die Bearbeitung der Übungsaufgaben herangezogen. Während der integrierten Gruppenübungen in den Vorlesungen hat man als Dozent die Möglichkeit zwischen den Teams zu wandern und die Arbeit der Teams zu beobachten. Hier kann man gut den Stand der Studierenden erkennen und auch direkt Feedback geben, falls die Studierenden kollektiv falsche Lösungsansätze verfolgen. Dies gilt ebenso bei den Diskussionen der Teamergebnisse. Formative Prüfungen durch Classroom Response Systeme, wie Klicker-Systeme, kamen hier noch nicht zum Einsatz, wären aber denkbar.

Im Bereich der Umsetzung wurde nach jedem Vorlesungstermin geprüft, ob die angesetzte Zeit für die Vermittlung der Teilkompetenz durch Vorlesungsteil und integrierten Übungen grob eingehalten wurde. Hier zeigte sich, dass der Zeitaufwand für die Gruppenübungen unterschätzt wurde, insbesondere die Zeit für die Diskussion der Ergebnisse. Letztere werden aber von uns als wesentlich für die Verankerung oder Vertiefung der Themen angesehen.

Fehler in den Unterlagen wurden direkt als Errata korrigiert. Weiterhin wurde ein Tagebuch geführt, in dem Beobachtungen aus den Veranstaltungen und Verbesserungsideen festgehalten wurden. Von Seiten der Hochschule werden regelmäßig Evaluationen der einzelnen Vorlesungen durchgeführt, so auch für dieses Modul. Hier waren insbesondere die Freitextrückmeldungen der Studierenden hilfreich, um Anpassungen vorzunehmen.

Maintenance

Die Vorlesung wurde nun bereits zwei Mal, unser Meinung nach erfolgreich, durchgeführt. Nach jedem Durchlauf wurden zunächst die Verbesserungsideen und die Beobachtungen aus dem Tagebuch ausgewertet und zusammengefasst. Analog zum prinzipiellen Vorgehen in der Softwareentwicklung müssen hier bei neuen Erkenntnissen aus dem Feld die Requirements überarbeitet werden, das Design angepasst werden, usw. Entsprechend führten die gewonnenen Erkenntnisse zur Überarbeitung der Kompetenzen, der Zusatzinformationen, der Laboraufgaben, bis hin zur Neugestaltung einzelner Folien oder Vorlesungsabschnitte. Die beobachteten Schwierigkeiten bei der Wartungsaufgabe führten zum Beispiel dazu, dass das Thema ”einarbeiten in ein fremdes System” nun explizit in der Vorlesung thematisiert wird.

Prüfung

Abgeschlossen wird das Modul mit einer schriftlichen Prüfung am Ende der Vorlesungen, wie es die Prüfungsordnung vorschreibt. Ein erfolgreicher Abschluss aller Labortermine ist Voraussetzung für die Zulassung zur Klausur. Eine individuelle Benotung ist im Laborpraktikum nicht erforderlich, nur eine individuelle Zulassung zur Klausur. Für das erfolgreiche Abschließen eines Labortermins wurden die abgelieferten Artefakte als Teamleistung bewertet. Auch wenn nur eine Zulassung erreicht werden muss, somit ein Mindestmaß an Leistung, so wurden auch hier beobachtet, dass es innerhalb einzelner Teams Ärger bezüglich der Zuarbeit einzelner Teammitglieder gab.

Die abschließende Klausur ist weiterhin klassisch mit kleinen Aufgaben aufgebaut und akkumuliert Teilpunkte, die wiederum auf eine Note abgebildet werden. Bei der Entwicklung der Klausuraufgaben konnten Aufgaben leicht aus den Kompetenzen abgeleitet werden. Aus der Kompetenz K-10.7 ”Die Studierenden können die Qualität von Requirements beurteilen.” wurde die Prüfungsaufgabe ”Beurteilen Sie die nachfolgenden Requirements auf ihre Qualität.” Dazu war ein Block mit Requirements gegeben, die in der Vorlesung angesprochene Mängel unterschiedlichster Art enthielten.

Bei Aufgaben höherer Niveaustufen, wie zum Beispiel die Qualitätsbewertung von gegebenen Requirements oder Modellierung, wurde in den Teilaufgaben ein Niveaustufenmodell angewandt. Es wurde nicht die Anzahl der gefundenen Problemstellen (ein Ding, ein Punkt), sondern die Komplexität und Tiefe der Analyse bewertet und auf eine Punktzahl abgebildet (Qualität der Antwort). Wurden zum Beispiel bei der Qualitätskontrolle der Requirements ausschließlich die Rechtschreibfehler bemängelt, aber nicht erkannt, dass immer wieder Synonyme verwendet wurden oder das es Widersprüche gab, so führte dieses zu einem schlechteren Ergebnis.

Ein vollständiger Übergang zur kompetenzorientierten Prüfung würde zwei bis drei komplexe Learning Outcomes/Kompetenzen prüfen, welche die notwendigen Teilkompetenzen mit abdecken. Ein derartiger Umstieg der Prüfungsmodalitäten wurde als zu großer Bruch betrachtet und wird nach und nach durch Abkehr von kleinen Teilaufgaben zu wenigen komplexeren Aufgaben vollzogen. Als Zwischenstufe wird eine Klassifikation der Teilaufgaben, wie von Waffenschmidt (Waffenschmidt (2013)) vorgeschlagen, mit entsprechendem Bewertungsschema angestrebt.

Fazit

Die hier gezeigte Übernahme von Vorgehensmodellen des Softwareengineerings erscheint vielleicht ungewöhnlich, hat uns aber einen guten Überblick verschafft und Hinweise für das jeweils weitere systematische Vorgehen gegeben.

Durch die gute Formulierung von (Teil-)Learning Outcomes war es zum Teil sehr leicht, Prüfungsaufgaben zu entwickeln. Auch konnte man den Studierenden Hinweise auf die Prüfung geben, insbesondere da die zur Verfügung stehenden Klausuren der Vergangenheit nicht mehr zur aktuellen Ausgestaltung des Moduls passten. Weiterhin hat die Kompetenzorientierung mit dem Hinterfragen der erwarteten Handlungsfähigkeit dazu geführt, dass sich weniger „sinnloses“ Wissen in der Vorlesung anhäuft als bei einer themenorientierten Vorgehensweise.

Der Ansatz zur Übernahme von Methoden aus dem Softwareengineering wurde noch nicht konsequent umgesetzt. So fehlt zum Beispiel eine Betrachtung des Kontextes der Vorlesung, d. h. die genaue Betrachtung aller anderen Vorlesungen des Curriculums. Auch wurde kein konsequentes Controlling bezüglich der eingeflossenen Aufwände durchgeführt[3]. Ein wesentlicher nächster Schritt wäre nun auch das sich Lösen von den gegebenen Rahmenbedingungen und zeitlichen Taktungen. Diese schränken das Denken in möglichen Vermittlungsmethoden noch zu sehr ein.

Die mit der Kompetenzorientierung einhergehende Problemorientierung war gerade in den Laboren eine Herausforderung für die schwächeren Studierenden. Sie erwarten klar umrissene (Teil-)Aufgaben, die abzuarbeiten sind. Die gefühlte Akzeptanz von realistischen Problemstellungen statt Aufgaben war bei den leistungsstärkeren, bzw. softwareaffineren Studierenden hoch und wurde begrüßt. Die Laboraufgaben waren so konzipiert, dass diese nur im 4er-Team bei strukturiertem Vorgehen in der Zeit zu lösen waren. Ein ”Gebastel”, wie man es oftmals im dritten Semester noch beobachtet, führte nicht zum Erfolg und ließ einige Teams dann doch auf systematisches Arbeiten umschwenken.

Die an das Softwareengineering angelegte systematisch konstruktive Vorgehensweise kann unserer Meinung nach auch für andere Module angewandt werden. Das Lecture Engineering gibt Dozenten aus den Ingenieurwissenschaften/der Informatik Mittel aus Ihrer Denkwelt an die Hand, um ihre Module durchzuplanen und auszugestalten. Die Qualität der Module erhöht sich, da eine Transparenz existiert und ein Abgleich im Sinne des Constructive Alignment zwischen Learning Outcome, Vorlesungs- und Übungsinhalten und der Prüfung vereinfacht wird.

Literatur

[Biggs u. Tang 2007] BIGGS, J. ; TANG, C.: *Teaching for Quality Learning*. McGraw-Hill Companies,Inc., 2007

[Bloom 1972] BLOOM, Benjamin S. ; BLOOM, Benjamin S. (Hrsg.): *Taxonomie von Lernzielen im kogni-*

[3] Falls Leser sich auf einen ähnlichen Weg begeben, wären wir an entsprechenden Daten interessiert.

tiven Bereich. 4. Beltz Verlag, Weinheim und Basel, 1972

[Böttcher u. a. 2011] BÖTTCHER, Axel ; THURNER, Veronika ; MÜLLER, Gerhard: Kompetenzorientierte Lehre im Software Engineering. In: LUDEWIG, Jochen (Hrsg.) ; BÖTTCHER, Axel (Hrsg.): *Software Engineering im Unterricht der Hochschulen (SEUH 2011)*. München : dpunkt.verlag, Heidelberg, 2011, S. 33–39

[Brabrand 2008] BRABRAND, Claus: *Teaching Teaching & Understanding Understanding*. DVD, 2008

[Dallmeier u. a. 2013] DALLMEIER, Valentin ; GROSS, Florian ; HAMMACHER, Clemens ; HÖSCHELE, Matthias ; JAMROZIK, Konrad ; STREIT, Kevin ; ZELLER, Andreas: Muster der Softwaretechnik-Lehre. In: SPILLNER, Andreas (Hrsg.) ; LICHTER, Horst (Hrsg.): *Software Engineering im Unterricht der Hochschulen (SEUH 2013)*. Aachen : dpunkt.verlag, Heidelberg, 2013, S. 101–102

[Hammerschall u. Beneken 2013] HAMMERSCHALL, U. ; BENEKEN, G.H.: *Software Requirements*. Pearson Studium, 2013. – ISBN 9783868941517

[Hummel 2013] HUMMEL, Oliver: Transparente Bewertung von Softwaretechnik-Projekten in der Hochschullehre. In: SPILLNER, Andreas (Hrsg.) ; LICHTER, Horst (Hrsg.): *Software Engineering im Unterricht der Hochschulen (SEUH 2013)*. Aachen : dpunkt.verlag, Heidelberg, 2013, S. 103–114

[Kennedy 2007] KENNEDY, D.: *Writing and Using Learning Outcomes: A Practical Guide*. University College Cork, 2007. – ISBN 9780955222962

[Liebehenschel 2013] LIEBEHENSCHEL, Jens: Software-Engineering Projekte in der Ausbildung an Hochschulen - Konzept, Erfahrungen und Ideen. In: *Software Engineering im Unterricht der Hochschulen (SEUH 2013)*. Heidelberg : .verlag, 2013, S. 27–34

[Lotter 2005] LOTTER, Wolf: DER ROTE FADEN. In: *Brandeins* 02 (2005)

[Macke u. a. 2012] MACKE, G. ; HANKE, U. ; VIEHMANN, P.: *Hochschuldidaktik: Lehren – vortragen – prüfen – beraten.* Beltz, 2012

[Reis 2013] REIS, Oliver: Kompetenzorientierte Prüfungen: Prüfungstheorie und Prüfungspraxis. In: GUTGE-WICKERT, Angelika (Hrsg.) ; KESSLER, Ulrike (Hrsg.): *Die homöopathische Behandlung chronischer Krankheiten*, 2013

[Waffenschmidt 2013] WAFFENSCHMIDT, Eberhard: Kompetenzorientierte schriftliche Prüfungen. In: *Handbuch Qualität in Studium und Lehre* (2013)

[Zürich 2008] ZÜRICH ; HOCHSCHULDIDAKTIK, Universität Zürich. A. (Hrsg.): *Leistungsnachweise in modularisierten Studiengängen: Dossier*. Universität Zürich, Arbeitsstelle für Hochschuldidaktik, 2008