Kompetenzorientierte Requirements Engineering Ausbildung im Rahmen eines SE-Moduls

Thomas Lehmann[1] Bettina Buth[2]

Abstract: Requirements Engineering ist ein wesentlicher Teilschritt eines Software Entwicklungsprozesses. Im Rahmen eines Moduls Software Engineering, das den gesamten Prozess abdecken soll, steht hierfür jedoch nur wenig Raum für die Ausbildung zur Verfügung. In diesem Paper wird beschreiben, wie - neben der erforderlichen Wissensvermittlung - in der Kombination von einem verpflichtenden Praktikum mit einer Vorlesung Fallanalyse, Dokumentation und Qualitätssicherung eingeübt werden. Ausgangspunkt der kompetenzorientierten Gestaltung ist dabei die Ermittlung eines geeigneten Learning Outcomes, aus dem die Lehrform im Sinne des Constructive Alignment abgeleitet wird.

Keywords: Kompetenzorientierte Lehrveranstaltung, Requirements Engineering, Learning Outcome, Constructive Alignment

1 Einleitung

Requirements Engineering ist ein wesentlicher Baustein im Software Entwicklungsprozess. Fehler, die in diesem Schritt gemacht werden, wirken sich extrem stark auf das Endprodukt aus. Dieses gilt für klassische Entwicklungsprozesse ebenso wie für agile Prozesse, da auch innerhalb eines kurzen Sprints die jeweiligen Anforderungen klar sein müssen.

Mit den Themen des Requirements Engineerings kann man inzwischen ein ganzes Modul füllen. Oftmals ist das Requirements Engineering allerdings nur Teil eines oder mehrerer Software Engineering Module innerhalb des Curriculums der Informatik-Ausbildung. Entsprechend wenig Zeit steht zur Verfügung. Was sind die wesentlichen Teile, die vermittelt werden sollen? Die Erfassung von Requirements; psychologische Aspekte; Kano-Modell; Lasten-/Pflichtenheft oder User Stories und Tasks; Requirement-Engineering Tools; formale Spezifikation; Dokumentation, Qualitätssicherung, Tracing, Maintenance der Requirements, ... sind nur einige der Aspekte.

Besser ist die Frage, was die Studierenden mitnehmen sollen? Was ist für sie an dieser Stelle der Ausbildung wichtig? Was sollen sie am Ende der Software Engineering Module

[1] HAW Hamburg, Department Informatik, Berliner Tor 7, 20146 Hamburg, T.Lehmann@haw-hamburg.de
[2] HAW Hamburg, Department Informatik, Berliner Tor 7, 20146 Hamburg, Bettina.Buth@haw-hamburg.de

wissen und vor allem können? Da der Requirements-Prozess dem Design-Prozess vorgelagert ist und sich das Design auf die Requirements bezieht, ergibt sich die Frage, was man für einen guten Design-Prozess benötigt? Offensichtlich gute Requirements! Man kann die Frage aus Sicht des Entwicklungsprozesses also reduzieren auf die Kernfrage: "Was sind gute Requirements?"

Aus Sicht der Studierenden bedeutet diese Kernfrage, sie sollen in der Lage sein, gute Requirements zu erstellen. Die erstellten Requirements sollten angemessen strukturiert, schriftlich erfasst und von entsprechend hoher Qualität sein. Wichtig dabei ist, dass die Studierende selber Requirements aufschreiben und deren Qualität prüfen und beurteilen können. Studierende haben oft Schwierigkeiten mit dem Formulieren entsprechender Texte. Gerade hierin besteht ein hohes Potential der Weiterentwicklung auf Seiten der Studierende. Ein weiterer didaktischer Aspekt ist, dass Mängel und Fehler bei anderen besser erkannt werden als bei einem selber, was bei der Ausbildung der Qualitätssicherung genutzt werden kann.

2 Konzept und Umsetzung

Aus diesen Überlegungen heraus wurde das Learning Outcome [Ke08, BT07] für den Teilbereich des Requirements Engineering innerhalb des Software Engineering Moduls im 3. Semester entwickelt. Aus diesem Learning Outcome können dann untergeordnete Learning Outcomes abgeleitet und schließlich in Lehreinheiten überführt werden (vgl. [LB15]).

Das hier entwickelte Learning Outcome folgt der Form, die von Oliver Reis [ASD, Re13] vorgeschlagen wurde:
"Die Studierenden können als Ausgangspunkt weiterer Entwicklungen Requirements systematisch dokumentieren, indem sie
1.) die Wünsche des Kunden, die Abläufe und den Kontext identifizieren,
2.) strukturiert dokumentieren und
3.) geeignete Qualitätssicherungsmaßnahmen durchführen."

Um dieses Ziel zu erreichen, wäre eine ideale Ausbildungsform die Begleitung eines erfahrenen Requirements Engineers, die Verhandlung mit einem echten Kunden, sowie die Ausarbeitung von Requirements, die in einem industriellen Entwicklungsprozess eingesetzt werden. Die Qualität in der Ausführung dieser Prozessschritte würde dabei geprüft und benotet werden. Unter den Rahmenbedingungen der Hochschule ist dieser Aufwand einer solchen Individualbetreuung nicht realisierbar. Standardlehrmethoden (vgl. [MHV12]), wie Gruppenpuzzle oder ähnliches, greifen unserer Meinung nach auch nicht.

In der Informatik der HAW Hamburg sieht der Rahmen eines Moduls derzeit wie folgt aus: seminaristische Vorlesung an 12 Terminen mit nicht-verpflichtender Teilnahme, ca. 45 Studierende. Der Vorlesung sind vier Labortermine zugeordnet. Typischerweise besteht eine Laborgruppe aus 12-16 Studierende pro Termin. Die Teilnahme ist verpflichtend und ist Prüfungsvorleistung. Aufgaben im Praktikum werden in dieser Vorlesung in 4er Teams durchgeführt. Didaktisch wichtiger Aspekt des Laborpraktikums ist somit, dass alle Studierenden einer Praktikumsgruppe verpflichtend im gleichen Zeitraum am gleichen Ort

sind, zusammen mit den Betreuern. Hieraus ergibt sich eine personelle Planungssicherheit für die eingesetzte Lehrmethode.

Es stehen für das Thema Requirements Engineering und somit zum Erreichen des Learning Outcomes zwei Vorlesungstermine (insgesamt 4-6 h) und ein Labortermin (3 h) zur Verfügung. Das Lehrkonzept aus Vorlesung und Labor sieht deshalb wie folgt aus: Die Vorlesung wird mit seminaristischem Unterricht für die Wissensvermittlung, Diskussion und für integrierte Übungen verwendet. Der Labortermin wird zur abschließenden gemeinsamen Arbeit an einem Thema genutzt. Die Bearbeitung der Laboraufgabe ist wesentlich, da hier die Studierenden das vermittelte Wissen anwenden sollen und dieser Prozess durch die Betreuer beobachtbar ist.

Der Labortermin ist dabei folgendermaßen konzipiert: Die Studierenden erhalten die Fallbeschreibung eines technischen Systems im Umfang von ungefähr einer Seite. Im letzten Semester war es die Beschreibung des autonomen Fahrzeugs, das am Carolo Cup [FAU] teilgenommen hat. Die Fallbeschreibung muss den Kunden ersetzen, da das Führen eines Gespräches zur Requirements-Erhebung zu viel Zeit in Anspruch nehmen würde. Einzelne Aspekte in der Fallbeschreibung sind ausreichend schwammig oder interpretierbar formuliert, um die Problematik eines ”unwissenden” Kunden zu verdeutlichen. Auch wird in der Fallbeschreibung der Kontext - in diesem Fall das Fahrzeug und der Wettbewerb - thematisiert; Teile der Bearbeitung beziehen sich dagegen auf die Steuerungssoftware, die in ihrer Funktionalität nicht explizit beschrieben wird.

Als Vorarbeit muss der Text analysiert, der Systemkontext ermittelt werden, sowie zusammen mit einem Use Case, Requirements und ausgewählten Testfällen dokumentiert werden. Hierdurch werden die Punkte 1 und 2 des Learning Outcomes, die Erhebung und Dokumentation der Anforderungen, angesprochen.

Die erste Version an Requirements wird am Beginn des Labortermins abgegeben und einer Schnellprüfung durch den Dozenten unterzogen. Sie gilt als Zulassung zum Termin und soll vornehmlich prüfen, ob die nachfolgenden Schritte mit der eingereichten Lösung durchgeführt werden können.

Die restliche Zeit des Labortermins von ca. 3 h wird für die Durchführung eines Review-Prozesses genutzt. Dazu werden die Lösungen der 4er-Teams jeweils einem anderem Team zum Review und zur Erstellung einer Mängelliste gegeben. Hierdurch wird der 3. Punkt des Learning Outcomes, die Qualitätsprüfung, angesprochen.

Im Praktikumslabor werden Review-Stationen aufgebaut, d.h. es wird jedem Team ein Beamer oder Großbildschirm zur Verfügung gestellt sowie ein Computer für das Führen des Protokolls. Oftmals kann letzteres entfallen, da die Studierenden ihre eigenen Laptops verwenden. Wichtig ist, dass das Team und die Dozenten das zu prüfende Dokument gemeinsam betrachten können.

Das Review erfolgt strukturiert mit einem Moderator/Protokollant und Reviewern im Sinne eines moderierten Technical Reviews. Der Moderator wählt die zu betrachtenden Punkte aus, d.h. einen Abschnitt oder nur ein einzelnes Requirement. Nach einer kurzen ruhigen Denkphase, gesteuert durch den Moderator, werden die Reviewer um ihre Meinung gebe-

ten. Die Reviewer müssen zunächst alle ihre Meinung zu dem Punkt abgeben, erst danach wird die Diskussion eröffnet. Der Moderator soll explizit darauf achten, dass alle zunächst ihre Meinung ungestört äußern können. Das Ergebnis der Diskussion wird vom Moderator festgehalten. Dann geht es zum nächsten Punkt. Die Moderatoren sollen nicht mitdiskutieren und die Rolle des Moderators wechselt innerhalb des Teams nach ca. 20-30 min. Alle Studierende aus dem Team sollen mindestens ein mal die Rolle des Moderators inne haben.

Das strikte Vorgehen aus Auswahl, Ruhe, Meinungen äußern und Diskussion hat folgende Hintergründe, die den Studierenden auch erläutert werden: Erfolgt keine konsequente Einhaltung des Vorgehens, so werden zu viele über das Dokument gestreute Aspekte gleichzeitig diskutiert. Erfolgt sofort eine freie Diskussion, so setzen sich immer die Alpha-Tiere durch und das Gespräch konzentriert sich schnell auf einige wenige Aspekte des betrachteten Abschnitts. Durch die Ruhephase und das einzelne Abfragen der Reviewer haben die introvertierteren Studierenden Zeit zum Denken und können und müssen sich ungestört äußern. Die Moderatoren sind aufgefordert auch bei der Diskussion gezielt auf ruhige Kommillitonen zu achten. Nebenbei ist ein ”ich schließe mich der Meinung an” in der ersten Runde nicht zulässig, d.h. selbst schwache Studierende müssen mindestens den Vorredner in eigenen Worten wiederholen. Falls Zeit vorhanden ist, kann man den Unterschied in der unstrukturieren und der strukturierten Vorgehensweise im Rahmen der Vorlesung demonstrieren.

Die Reviews erfolgen unter Beobachtung des Dozenten und des Labormitarbeiters. Diese wandern zwischen den Teams und beobachten zunächst aus dem Hintergrund, da sich die Studierenden sonst sofort kontrolliert fühlen. Nach und nach wird von der Dozentenseite steuernd eingegriffen.

Von Dozentenseite wird auf die Einhaltung des Prozesses und der Rolleneinteilung geachtet, ebenso auf die Zeiteinteilung, da alle Teilbereiche des Dokuments begutachtet werden sollen und nicht in 3 h nur der Systemkontext intensiv bearbeitet wird. Weiterhin werden Hinweise zum Review gegeben. Stimmt der Kontext, werden Synonyme verwendet, wird hier nicht gerade ein Passivsatz verwendet? Diese Aspekte sind den Studierenden aus Vorlesung und einer beigelegten Checkliste bekannt, werden aber oft nicht angewendet oder wahrgenommen. Es erfolgt zunehmend die Aufforderung, Texte auf die sprachliche Goldwaage zu legen. Gerade hierbei wird dann der Unterschied zwischen dem, was man meint zu lesen und dem, was dort wirklich steht offenbar (vgl. [Sc01]). Gezielt wird hier eine Übersensibilisierung betrieben, da aus der Vergangenheit genügend Beispiel gibt, wo kleine Missinterpretationen zu Katastrophen führten, wie beispielsweise dem Mars Climate Orbiter [St99]. Auf solche Beispiele wird vorab im Rahmen der Vorlesunge eingegangen.

Am Ende werden die Mängellisten den Autoren übergeben und die Mängel sollen innerhalb von einer Woche als Nacharbeit behoben werden. Es erfolgt dann noch eine stichprobenartige Kontrolle der überarbeiteten Versionen durch den Dozenten.

In der Abschlussrunde des Praktikums als Stand-up Meeting wird sichtbar, dass nun die Studierenden auch bei ihren eigenen Versionen deutliche Mängel sehen. Insbesondere was sie meinten zu schreiben und was sie wirklich geschrieben haben. Durch das Spielen mit

Sprache entwickelten einige Studierende auch einen deutlichen Spaß während der Bearbeitung der Aufgabe. So sollte beispielsweise das Fahrzug an einer Stop-Markierung kurz anhalten. Daraus wurde das Requirement ”Die Steuersoftware soll mit Hilfe der Kamera-Daten Stoppmarkierung auf der Fahrbahn erkennen und kurz anhalten.” gemacht. Aus dem Kontext ist das Requirement verständlich, nimmt man den reinen Text, so muss die Software kurz anhalten.

3 Prüfungsform

Bis zum Wintersemester 2015 wurde im Modul Software Engineering eine Klausur als Prüfungsform gewählt. Im Themenbereich Requirements Engineering wurden dabei vorwiegend Wissens- und Verständnisfragen gestellt. Daneben wurde eine Liste von Requirements gegeben, die deutlich mängelbehaftet war. Die Mängel sollten lokalisiert und benannt werden. Das Schreiben von Requirements oder das Erstellen eines Use Cases wird (noch) nicht gefordert, da der Aufwand der Bewertung als zu hoch angesehen wird (Prüfungsökonomie). Hier muss im Rahmen der Möglichkeiten der Prüfungsvorleistung auf das Erreichen eines Mindestmaß geachtet werden.

Ab dem Wintersemester 2015 erfolgt eine mündliche Prüfung. Hier ist vorgesehen, dass die Studierenden auf Basis einer mündlichen Fallbeschreibung Requirements formulieren oder Teile eines Use Cases formulieren sollen. Hier kann dann direkt über das Vorgehen oder die erzielte Qualität als Prüfungsgespräch diskutiert werden.

Indirekt erfolgt eine weitere Prüfung im nachgelagerten Modul Embedded System Engineering, in dem die Vorlesung Software Engineering 2 eingebettet ist. Im Rahmen dieses Moduls müssen die Studierenden als Projekt eine Steuerungssoftware für eine Produktionsanlage erstellen. Hierbei müssen die Methoden und Techniken aus Software Engineering 1 angewendet werden und unter anderem aus der allgemeinen Aufgabenbeschreibung die Requirements erstellt und umgesetzt werden.

4 Fazit

Aus Dozentensicht ist diese Art der Laborgestaltung konsistent im Sinne eines Constructive Alignment [BT07] mit dem Learning Outcome. In der Klausur wird neben Wissensfragen auch die Beurteilung und Aufdeckung von Mängeln gefordert. Hierdurch wird auf verschiedenen Taxonomiestufen [Bl72] geprüft, was derzeit noch ein Mangel in der Prüfungsform bzw. der Aufgabenstellungen darstellt. In der mündlichen Prüfung kann direkt die Anwendug des Wissens gefordert und beobachtet werden.

Ein Evaluierung des Konzeptes auf Basis von Klausurergebnissen hat nicht statt gefunden. Dieses ist auch schwierig, da mit dieser Neukonzipierung sich der Abschnitt Requirements Engineering deutlich geändert und damit auch die Prüfungsanforderungen/-aufgaben geändert haben. Als Indiz soll nun das Verhalten in der nachfolgenden Veranstaltung beobachtet werden, ob hier die Methoden und Werkzeuge des Software Engineerings im Rahmen des Projekts selbstverständlicher und konsequenter angewendet

werden. Parallel findet ein zweiter Durchgang des Moduls Software Engineering 1 unter Berücksichtigung des vorgestellten Konzepts statt.

Es kann daher bisher nur folgende Beobachtung für den Lernerfolg herangezogen werden: Die Studierenden kommen mit einer Lösung ins Praktikum, die sie zwar mit mittlerem Aufwand erstellt haben, aber doch ihrer Meinung nach eine solide Lösung darstellt. Am Ende des Praktikums sind diese Meinungen zum Teil deutlich revidiert und zwar nicht durch Rückmeldung der Betreuer, sondern auf Basis der eigenen Betrachtung.

Auch das übertrieben akribische Betrachten der Anforderungsvorschläge mit bewusst missverstandener und dabei oft komischer Interpretation motiviert und sensibilisiert die Studierenden. In einem anderen Fallbeispiel für die Software eines Garagentorantriebs wurde das vorgeschlagene Requirement "Das Garagentor wird mit Hilfe der Fernbedienung geöffnet." hinterfragt mit "Nutze ich die Fernbedienung als Hebel und hebel das Tor auf?". Diese mag zwar Quatsch sein, führt aber zu einem sportlichen Hinterfragen des geschriebenen. Wenn man als Dozent auch noch den Spaß am kritischen Arbeiten und Hinterfragen weckt - um so besser. Wenn dann die Studierenden bei nachfolgenden Laboraufgaben sofort die Eindeutigkeit der Aufgabenformulierung monieren, dann kann man diese Frage als Erfolg der Ausbildung werten.

Literaturverzeichnis

[ASD] Kompetenzorientiert Lehren, Lernen und Prüfen an der HAW Hamburg, http://www.haw-hamburg.de/qualitaet-in-der-lehre/asd/kompetenzorientierung.html.

[Bl72] Bloom, Benjamin S.: Taxonomie von Lernzielen im kognitiven Bereich. Beltz Verlag, Weinheim und Basel, 4. Auflage, 1972.

[BT07] Biggs, J.; Tang, C.: Teaching for Quality Learning. McGraw-Hill Companies,Inc., 2007.

[FAU] FAUST - Fahrerassistenz und autonome Systeme. http://faust.informatik.haw-hamburg.de.

[Ke08] Kennedy, Declan: Lernergebnisse (Learning Outcomes) in der Praxis. DAAD, 2008.

[LB15] Lehmann, Thomas; Buth, Bettina: Lecture Engineering. In: Software Engineering im Unterricht der Hochschulen (SEUH) 2015 CEUR Workshop Proceedings. Jgg. 1332, Dresden, 2015.

[MHV12] Macke, G.; Hanke, U.; Viehmann, P.: Hochschuldidaktik: Lehren – vortragen – prüfen – beraten. Beltz, 2012.

[Re13] Reis, Oliver: Kompetenzorientierte Prüfungen: Prüfungstheorie und Prüfungspraxis. In (Gutge-Wickert, Angelika; Kessler, Ulrike, Hrsg.): Die homöopathische Behandlung chronischer Krankheiten. 2013.

[Sc01] Schneider, Wolf: Deutsch für Profis: Wege zu gutem Stil. Goldmann, 11. Auflage, 2001.

[St99] Stephenson, Arthur G.; LaPiana, Lia S.; Mulville, Daniel R.; Rutledge, Peter J.; Bauer, Frank H.; Folta, David; Dukeman, Greg A.; Sackheim, Robert; Norvig, Peter; Weiler, Edward J.; Gregory, Frederick D.; Durnya, Louis; Isbell, Douglas M.: Mars Climate Orbiter Mishap Investigation Board Phase I Report. Bericht, NASA, November 1999.