Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

Ein verbessertes Softwaretechnikpraktikum: zwischen grüner Wiese und Legacy-Systemen

Björn Axenath · Stefan Henkler

Universität Paderborn, Fachgebiet Softwaretechnik
Warburgerstraße 100, 33098 Paderborn
{axenath|shenkler}@uni-paderborn.de

Zusammenfassung

An der der Universität Paderborn soll Studenten im Grundstudium die Entwicklung von komplexen Softwaresystemen im Rahmen des Softwaretechnikpraktikums vermittelt werden. Dazu führen sie in kleinen Teams gemeinsam einen Entwicklungsprozess durch, um das geforderte Produkt zu erstellen. Aufgrund der geringen Dauer eines Praktikums werden die Schritte des Entwicklungsprozesses i. Allg. nur einmal durchgeführt.

In der Praxis wird Software selten auf der grünen Wiese entwickelt: In der Regel werden existierende Systeme, sog. Legacy-Systeme, erweitert. Für einen Studenten im Grundstudium birgt jedoch schon das Entwickeln von Software ohne die Erschwernis eines Legacy-Systems mannigfache Herausforderungen. Für heutige Projekte ist zusätzlich der Einsatz von Technologien erfolgsentscheidend; auch dieser Aspekt sollte daher im Rahmen eines Praktikums vermittelt werden.

Um Studenten dennoch bereits im Grundstudium an die Methoden heranzuführen, die in der Praxis verlangt werden, und durch Wiederholung der Tätigkeiten ihren Lernerfolg zu verbessern, wurde ein neues Konzept für Aufgabenstellungen erstellt und erprobt. Kern dieses Konzeptes ist es, jedes Team des Softwaretechnikpraktikums mehrere Teilaufgaben bearbeiten und durch die Übernahme von Software aus anderen Teams ein komplexes Softwaresystem entwickeln zu lassen.

Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

1 Einleitung

Das Softwaretechnikpraktikum an der Universität Paderborn, eine Veranstaltung für Studenten des Studiengangs Informatik oder des Studiengangs Ingenieurinformatik mit dem Schwerkunkt Informatik im ersten Studienabschnitt, hat das Ziel, die Entwicklung komplexer Systeme zu vermitteln. Hierbei sollen UML und Java verwendet werden. Diese Aufgabe soll in Teams mit jeweils ca. zehn Studenten durchgeführt werden. Da es sich bei industrieller Softwareentwicklung nur selten um Neuentwicklungen handelt, wird im Paderborner Softwaretechnikpraktikum seit mehreren Jahren dabei ein Reengineering eines Legacy-Systems durchgeführt (vgl. [Bot01]).

In den letzten Jahren wurde den Studenten der undokumentierte Code eines komplexen, verteilten Systems gegeben. Die Studenten mussten zunächst ein Reverse Engineering durchführen, um die Software zu verstehen. Erst danach konnte mit dem Forward Engineering begonnen werden.

Für einen Großteil der Studenten war dies die erste Begegnung mit einem großen Softwaresystem. Da es sich um ein Legacy-System handelte, war die Qualität bewusst niedrig gehalten. Die Erkenntnis für die Studenten beschränkte sich deshalb oft darauf, dass sie den Bedarf an Entwicklungsdokumentation für Softwaresysteme erkannten. Es war ihnen nicht möglich, an diesem System zu erkennen, ob es sich um gute oder schlechte Architektur bzw. gutes oder schlechtes Design handelte. Folglich übernahmen sie schechte Entwurfsmuster [Gam95] aus dem Legacy-System in ihre eigene Entwicklung.

Ein weiteres Problem der ursprünglichen Aufgabenstellung war das Vorgehensmodell. Wie viele andere Praktika [Dem99] lag dem Paderborner Softwaretechnikpraktikum das Wasserfallvorgehensmodell zugrunde, das es den Studenten nicht ermöglicht, aus Fehlern zu lernen und sofort das Gelernte erneut anzuwenden. Ziel war es, mit der neuen Konzeption des Praktikums unter den gegebenen Umständen ein iteratives Prozessmodell einzuführen, wie es auch bereits für die Universtität Dortmund beschrieben wurde (vgl. [Kop04]).

Bei dem bisherigen System wurde als einzige Technologie RMI [RMI06] eingesetzt. Unserer Meinung nach ist diese Technologie zu einfach im Vergleich zu den Technologien, die in den uns bekannten Industrieprojekten eingesetzt werden. Der effektive und effiziente Umgang mit Technologien, d.h. der Einsatz von Werkzeugen und Rahmenwerken zur Erstellung eines Produkts, ist unserer Meinung nach ein entscheidender Erfolgsfaktor für heutige Softwareprojekte. Um diesen Anfoderungen später gerecht werden zu können, müssen Studenten z.B. lernen, wie man sich zeitökonomisch in Technologien einarbeitet.

Diese Missstände veranlassten uns, ein neues Konzept für die Aufgabenstellungen des Softwaretechnikpraktikums zu erarbeiten. Dabei war es vorteilhaft, dass das Softwaretechnikpraktukum in den letzten Jahren von unserem Lehrstuhl betreut wurde. In den folgenden Unterabschnitten beschreiben wir die Lernziele

des Praktikums, gefolgt von den Rahmenbedingungen, die wir für die Gestaltung des Softwaretechnikpraktikums haben. In Abschnitt 2 beschreiben wir das neue Konzept der Aufgabenstellung. Die von uns gemachten Beobachtungen werden in Abschnitt 3 evaluiert. Zuletzt, in Abschnitt 4, ziehen wir ein Fazit aus dem in diesem Jahr erstmals nach dem neuen Konzept verlaufenen Praktikum.

1.1 Lernziele des Paderborner Softwaretechnikpraktikums

Wie oben bereits ausgeführt, soll das Softwaretechnikpraktikum die Entwicklung komplexer Systeme vermitteln. Dazu sollte wie bereits in den letzten Jahren auch das Reengineering behandelt werden. In diesem Jahr sollen die Themen Entwicklungsprozesse, Projektmanagement und Technologieeinsatz besser vermittelt werden. Selbstverständlich werden auch die Themen Konfigurationsmanagement, Qualitätssicherung und Soft Skills behandelt.

Der Entwicklungsprozess soll ein geplantes Vorgehen bei der Softwareentwicklung vermitteln. Hierbei ist das affektive Lernziel die Erstellung von Lastenheft, Pflichtenheft, Analyse- und Entwurfsdokumentation sowie eines Testberichts. Im Rahmen des Projektmanagements sollen dazu Verantwortlichkeiten festlegt, Aktivitäten geplant sowie ein Controlling der Arbeitsstunden durchgeführt werden, sodass die Studenten ihr prozessbasiertes Vorgehen reflektieren können.

Im Umgang mit Technologien liegt der Schwerpunkt auf den kognitiven Lernzielen. Die Studenten sollen sich beim Einsatz von Technologien über die Vor- und Nachteile bewusst sein, die in Bezug zu den anderen Lernzielen stehen. Aus Sicht des Entwicklungsprozesses muss z.B. eine Einarbeitungsphase aufgenommen werden, die auch vom Projektmanagement berücksichtigt werden muss. Die Qualitätssicherung muss den Einfluss der Qualität der eingesetzten Technologie auf das gesamte zu entwickelnde Produkt berücksichtigen.

Im Bereich des Reengineering ist folgendes Lernziel zu erreichen: Die Studenten sollen aus dem Reverse Engineering für das folgende Forward Engineering Schlüsse ziehen. Im affektiven Bereich sollen sie den Nutzen der Reverse-Engineering-Modelle wahrnehmen und daraufhin die Qualität der eigenen Modellierung verbessern.

1.2 Voraussetzungen des Softwaretechnikpraktikums

Die Studenten haben u.a. die folgenden relevanten Vorkenntnisse im Programmieren im Kleinen am Beispiel von Java erworben sowie im Einsatz der UML. Eine detaillierte Beschreibung der vorausgesetzten Kenntnisse findet sich im Modulhandbuch des Studiengangs [Upb06].

Im Sommersemester 2006 hatten sich ca. 150 Studenten zum Softwaretechnikpraktikum angemeldet, sodass 16 Teams mit 8 bis 10 Teilnehmern gebildet werden konnten. Die Arbeitslast ist durch 10 ECTS Punkte definiert.

Für die Betreuung des Praktikums wurden vier halbe Stellen wissenschaftlicher Mitarbeiter und 14 halbe SHK-Stellen zur Verfügung gestellt. Jedem Team stand ein Betreuer zur Seite, der an den wöchentlichen Teamsitzungen teilnahm. Während der Woche waren die Betreuer per E-Mail zu erreichen und sollten auf E-Mails innerhalb von 24 Stunden antworten. Die Betreuer mussten zwei Rollen einnehmen. Zum einen sollten sie die Teams inhaltlich unterstützen, indem sie Fragen beantworteten, aber keine Vorgaben machten. Zum anderen sollten sie die Teams kontrollieren und die Organisatoren des Praktikums über mögliche Probleme informieren. Darüber hinaus stand allen Teams eine Programmierberatung zur Verfügung, die ebenfalls eine wöchentliche Sprechstunde hatte und per E-Mail kontaktiert werden konnte. Fragen zur Aufgabenstellung wurden von den Organisatoren stellvertretend für einen virtuellen Kunden per E-Mail beantwortet. Die Betreuer der Teams trafen sich mindestens einmal wöchentlich, um auf die nächste Woche vorbereitet zu werden und über den Stand ihrer Teams zu berichten.

2 Die Konzeption des Praktikums

Das Praktikum wird hauptsächlich durch die Aufgabenstellung geprägt. In diesem Jahr wurde beschlossen, neben der Vorlesung und den wöchentlichen Treffen der Teams weitere untestützende Veranstaltungen anzubieten.

Zunächst beschreiben und erläutern wir die Konzeption der Aufgabenstellung, die sich in das zu entwickelnde Produkt und das Vorgehen aufteilt. Dabei begründen wir auch die konkrete Auswahl der eingesetzten Technologien, da diese zur Machbarkeit des Praktikums beigetragen haben. Abschließend geben wir an, welche Unterstützung die Studenten bei der Durchführung des Praktikums erfahren haben.

2.1 Eine integrierte Entwurfsumgebung

Jedes Team des Softwaretechnikpraktikums hatte die Aufgabe, eine Entwurfsumgebung für eingebettete Systeme zu entwickeln. Als Anwendungsbereich wählten wir automotive Systeme. Die Entwurfsumgebung sollte aus drei Werkzeugen bestehen: einem Komponenteneditor, einem Verteilungseditor und einem Diagnosewerkzeug. Mit dem Komponenteneditor sollten Software-Komponenten eines eingebetteten Systems durch ihre Schnittstelle und ihr Verhalten beschrieben werden. Mit dem Verteilungseditor sollte ein Netzwerk aus Rechnern erstellt werden können, auf dem die zuvor definierten Komponenten instanziiert werden können. Das Diagnosewerkzeug sollte das System ausführen und zur Diagnose über-

wachen können. Alle drei Komponenten sollten eine grafische Benutzeroberfläche haben. Die Teams mussten nur eine der drei Komponenten selber entwickeln, die anderen beiden Komponenten sollten von anderen Teams übernommen werden. Die vollständige Aufgabenstellung findet man unter [Swt06].

Als Datenmodell haben wir den Studenden Code zur Verfügung gestellt, der aus EMF-Modellen [Emf06] generiert wurde. Eine Besonderheit dieses Codes ist, dass die Daten durch Serialisierung in XMI persistent gemacht werden können. Dadurch reduziert sich der Implementierungsaufwand einer Applikation deutlich.

Das Modell enthielt keine Funktionalität zur Ausführung des Systems. Die Ausführungssemantik war textuell in der Aufgabenstellung beschrieben und musste von den Studenten implementiert werden. Im Folgenden bezeichnen wir diesen Code als Simulationsalgorithmus. In der Entwurfsumgebung ist er Teil des Diagnosewerkzeugs.

Durch den Anwendungsbereich werden den Studenten implizit elementare Fragestellungen aus dem Bereich des Software Engineering für eingebettete Systeme vermittelt, die Forschungsschwerpunkt der Universität Paderborn sind. Des Weiteren verfolgen wir, wie bereits in den Aufgabenstellungen der letzten Jahre, das Ziel, die Synergieeffekte aus kombinierter Forschung und Lehre zu nutzen [Geh02, Geh03].

Als Rahmenwerk für die Werkzeuge und deren Integration haben wir die Eclipse-Plattform vorgegeben [Ecl06]. Diese ermöglicht es, auf Grundlage einer Plug-in-basierten Architektur effizient eigene Werkzeuge zu erstellen, da sie Grundfunktionalitäten, wie z.B. das Verwalten von Konfigurationsdaten, bereits anbietet. Des Weiteren sind dort bereits viele Entwurfsmuster [Gam95], wie z.B. das Command Pattern für die Undo/Redo-Funktionalität vorhanden, sodass die Studenten einen qualitativ hochwertigen Code analysieren und daraus lernen können. Dies ermöglicht es den Betreuern der Teams, eine Binnendifferenzierung vorzunehmen, indem im Rahmen der Teamsitzung z.B. Entwurfsmuster behandelt werden können, falls das Team dieses Niveau erreicht hat. Dadurch wird ein iterativer Lernprozess ermöglicht, da diese Themen im folgenden Studienabschnitt vertieft werden können.

Die grafische Benutzeroberfläche sollte mit GEF (Graphical Editing Framework) erstellt werden [Gef06]. Dieses Rahmenwerk verfolgt die Model-View-Controller-Architektur und schafft somit ein Bewusstsein für Softwarearchitekturen. Die Kenntnis über Eclipse und GEF soll es den Studenten erleichtern, Software im Rahmen von Bachelor- und Masterarbeiten zu entwicklen.

2.2 Ablauf des Praktikums

Die wesentliche Neuerung des Softwaretechnikpraktikums bestand darin, dass die Aufgabe sich aus mehreren Teilaufgaben zusammensetzt. Für jede Teilaufgabe

wird ein vollständiger Entwicklungsprozess mit Lastenheft, Pflichtenheft, Analyse und Entwurf, Implementierung und Test durchgeführt (siehe Abb. 1). Diese Entwicklungsprozesse starten im Abstand von einigen Wochen. Zuerst wurde das eigene Werkzeug entwickelt, das erst später mit den beiden erworbenen Werkzeugen zu einer Entwurfsumgebung integriert wurde. Durch diese Aufteilung können wir unsere Ziele erfüllen.

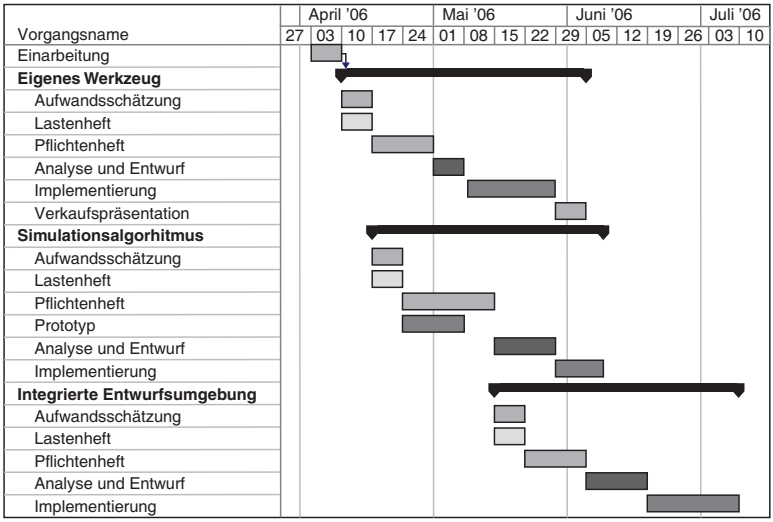

Abb. 1 *Ausschnitt aus der Projektplanskizze*

Die erste Teilaufgabe war das Erstellen eines eigenen Werkzeugs. Die zu erstellenden Werkzeuge wurden anhand der Teamgröße und Schwierigkeitsgrad des Werkzeugs auf die Teams verteilt. Den Teams wurden spezielle Rahmenwerke für die grafische Benutzeroberfläche und das Grundgerüst der Applikation vorgegeben. Folglich mussten sie sich zunächst in eine ausgereifte Technologie einarbeiten.

Da der Standardprozess mehrfach durchlaufen wurde, konnten die Studenten das Erstellen der Dokumente mehrfach üben. Bevor im nächsten Zyklus ein Dokument erstellt wurde, haben wir eine korrigierte Version des vorher erstellten Dokuments gleichen Typs zurückgegeben. Die Aufgabenstellung sah vor, dass neben der Erstellung des eigenen Werkzeugs und der Integration der Entwurfsumgebung auch die Ausführungssemantik durch einen Simulationsalgorithmus implementiert wurde. Die Dokumente zum eigenen Werkzeug und der Implementierung waren sich ähnlich, weil sie beide auf die Benutzungoberfläche ausgerichtet waren. Im Gegensatz hierzu war für den Simulationsalgorithmus keine Oberflächenbeschreibung notwendig. Diese unterschiedlichen Ausrichtungen der zu spezifizierenden Werkzeuge erforderten, dass die Dokumente nicht stereotyp geschrieben werden konnten und die Studenten sich über deren Zweck jedesmal erneut bewusst werden mussten.

Durch die Integration des eigenen Werkzeugs mit zwei fremden Werkzeugen zu einer einheitlichen Entwurfsumgebung sollen die Grundzüge des Reengineering vermittelt werden. Das Reengineering war für die Integration notwendig, weil trotz Aufgabenstellung mit einem Metamodell und vorgegebenem Rahmenwerk noch einige Anorderungen offen gelassen waren. Die Aufgabenstellung sah nicht explizit vor, eine Konsistenzprüfung für die Editoren zu erstellen. Beispielsweise erlaubte das Modell, eine Kompontente zu instanziieren, ohne ihr dabei einen Knoten zuzuweisen. Diese Eigenschaft konnte von dem Verteilungseditor oder dem Diagnosewerkzeug geprüft werden. Zudem hatten wir den Studenten empfohlen, dass die integrierte Entwurfsumgebung z.B. ein einheitliches Aussehen haben sollte. Dies bezog sich nicht nur auf Farben und Formen, sondern auch auf die funktionalen Aspekte des Layouts. Beispielsweise bietet das Applikationsrahmenwerk ein Fenster für Fehler oder Warnungen an, das anfänglich nur von einigen Teams genutzt wurde, während andere eine eigene Ausgabe für ihre Warnmeldungen entwickelten.

Wie bereits in der Einleitung erwähnt, ist das Reengineering durch die in der Praxis häufig vorkommenden Arbeiten mit Lagacy-Systemen motiviert. Die in diesem Praktikum konstruierte Situation deckt nicht alle Eigenschaften eines Legacy-Systems ab. Insbesondere entspricht die Größe der übernommenen Komponenten nicht der Größe realer Systeme, und die vorgebene Systemarchitektur ist nicht veraltet. Die Studenten mussten folglich nur den Umgang mit einem unzureichend dokumentierten System erfahren.

Für die Entwicklung des eigenen Werkzeugs und des Simulationsalgorithmus ist es notwendig, sich in die vorgegebenen Rahmenwerke und das Metamodell einzuarbeiten. Damit der Einarbeitungsaufwand in die Technologien rechtzeitig wahrgenommen wird, haben wir in der Projektplanskizze in der ersten Woche eine Einarbeitungszeit vorgesehen. Zur Unterstützung dieser Aufgabe haben wir auf der Homepage der Veranstaltung einen einfachen Editor veröffentlicht, der auf einem ähnlichen Modell basiert. Um die Semantik des Modells besser zu verstehen, haben wir verlangt, dass bereits während der Entwicklung des Pflichtenhefts für den Simulationsalgorithmus ein Prototyp erstellt wird.

2.3 Vorlesung, Tutorien und Betreuung

Ziel der Vorlesung ist es, die Konzepte des Software Engineering, wie z.B. Qualitätssicherung oder Konfigurationsmanagement, zu vermitteln. Übungen, in denen diese Konzepte angewendet werden, sind nicht vorgesehen. Wir haben dieses Jahr Tutorien zu unterschiedlichen Fragestellungen, wie z.B. Qualitätssicherung, Konfigurationsmanagement, Eclipse/GEF etc., angeboten, in denen wir die Studenten in die praktische Anwendung der Konzepte eingeführt haben. Aus jedem Team durfte abwechselnd ein Mitglied teilnehmen. Jedes Teammitglied hatte ein Rolle, z.B. Qualitätsmanagementbeauftragter, die seine Verantwortlich-

keit definierte. Durch die Tutorien konnten sich die Teammitglieder schulen lassen und waren dann u.a. dafür verantwortlich, das Gelernte in dem Team zu verbreiten.

2.4 Motivation der Studenten

Bei der Konzeption des Praktikums spielte auch die Motivation eine zentrale Rolle. Wir diskutieren diesen Aspekt des Praktikums, weil durch die motivationssteigernden Maßnahmen den Studenten die Möglichkeit zur Selbstreflexion gegeben wurde.

Nach Fertigstellung des eigenen Werkzeugs haben wir eine Produktmesse veranstaltet. Die Teams präsentierten dazu ihre Produkte an kleinen Messeständen. Während der Produktmesse konnten die Teams ihre Arbeit evaluieren, um so eine Auswahl für ihre integrierte Entwurfsumgebung zu treffen. Die Betreuer des Softwaretechnikpraktikums bewerteten ebenfalls die Produkte. Auf Basis dieser Bewertung wurde entschieden, welches Team welche anderen Werkzeuge einkaufen durfte.

Bei einer Produktmesse können neben der technischen Begutachtung auch die Soft Skills erkannt und bewertet werden. Da wir keine Vorgaben über die Art der Messestände gemacht haben, waren die Studenten gezwungen, kreativ zu werden, und konnten auf der Produktmesse ihre Präsentationsideen direkt mit den Ideen der Konkurrenten vergleichen.

Am Ende der Vorlesungszeit wurde ein Modellierungswettbewerb veranstaltet, bei dem jeweils zwei Teams gegeneinander die Software eines mechatronischen Systems modellieren oder analysieren mussten.

Die Industrierelevanz der Aufgabe wurde durch zwei Maßnahmen verdeutlicht. Zum einen dürfen die besten drei Teams ihr Produkt vor Vertretern der Automobilindustrie präsentieren und zum zweiten wurde im Rahmen der Vorlesung ein Vortrag eines Mitarbeiters aus der Softwareindustrie gehalten.

3 Evaluation des Praktikums

Die Evaluation des Praktikums erfolgt durch die Ermittlung des Lernerfolgs der Studenten, den Vergleich der Ergebnisse des diesjährigen Praktikums mit vorherigen und die Reflexion des Organisationsaufwands. Im Folgenden werden wir zuerst auf den Lernerfolg der Studenten eingehen. Dazu spezifizieren wir, was wir unter Lernerfolg verstehen, und gehen hierbei auf Beobachtungspunkte ein. Dann werden wir in Abschnitt 3.1 bis Abschnitt 3.7 den Lernerfolg der Studenten bei den gesetzten Zielen erläutern. Abschließend werden wir in Abschnitt 3.8 auf einen Vergleich mit den vorherigen Softwaretechnikpraktika eingehen und hierbei konkret die Vor- und Nachteile reflektieren. Zudem werden wir auf den Organisationsaufwand eingehen.

Den Lernerfolg des Praktikums messen wir durch Beobachtung der gesetzten Ziele (siehe Abschnitt 1.1). Um dies zu ermöglichen haben wir diverse Instrumentierungen des Praktikums vorgenommen, um die Beobachtbarkeit zu erhöhen. Neben den klassischen Beobachtungspunkten, wie Klausur und Endabgabe der Projektarbeit, haben wir zusätzlich folgende Instrumentierungen vorgenommen:

- Feedbackschleifen: Jedes Team wurde aufgefordert, zu jedem Vorlesungstermin Fragen an den Veranstalter zu stellen.
- Tutorien: Neben den Feedbackschleifen wurden zu den Vorlesungsthemen Tutorien gehalten. Für die verschiedenen Themen wurden Verantwortliche in den Teams bestimmt, die entsprechend die Tutorien besuchen mussten.
- Wöchentliche Tutorentreffen: Feedback über den Stand der Teams durch die Tutoren.
- Abgaben: Dokumente wie Pflichtenheft oder Analyse und Entwurf wurden kontrolliert.
- Präsentationen: Neben der Abschlusspräsentation gab es noch bis zu vier weitere Präsentationen. Hier wurden gezielt unterschiedliche Rollen (Käufer, Lehrender, Fachmann) durchgespielt, um die verschiedenen Anforderungen, wie Soft Skills oder Technologieverständnis, zu kontrollieren.
- Feedbackbögen: Bei der Endabgabe zum Softwaretechnikpraktikum wurden zusätzlich zu den Dokumenten und der erstellten Software ein Feedbackbogen und ein eigenes Resümee eingefordert. Das Resümee teilt sich in ein Grupenresümee und ein Resümee einzelner Personen auf. Der Feedbackbogen besteht aus den Kategorien Teamarbeit, Projektmanagement, Unterstützung durch die Veranstaltung, allgemeiner Lernerfolg, Lernerfolg bei Tätigkeiten und Lernerfolg bei Technologien.

Eine interessante Beobachtung bei der Auswertung der Resümees und Feedbackbögen ist, dass die Selbsteinschätzung aus dem Team oder einer Person in den meisten Fällen mit der Einschätzung des Veranstalters und der Betreuer korrelieren.

Im Folgenden werden wir auf die verschiedenen Ziele des Praktikums eingehen und mit Hilfe der Beobachtungspunkte den Lernerfolg des Praktikums aufzeigen.

3.1 Entwicklungsprozesse

Um den Lernerfolg des Entwicklungsprozesses zu beschreiben, müssen die verschiedenen Phasen des Entwicklungsprozesses beobachtbar sein. Dies haben wir zum einen durch Abgaben der verschiedenen Dokumente und deren Bewertung sowie durch Präsentationen ermittelt. Durch die drei parallelen Prozesse, die versetzt zueinander durchlaufen wurden, hatten wir zudem die Möglichkeit, die Abgaben zueinander in Vergleich zu stellen. Dies hat uns ermöglicht, einen konti-

nuierlichen Lernerfolg zu beobachten. Der Vergleich der Abgaben der parallelen Prozesse lässt stark darauf schließen, dass die Studenten einen Lernerfolg durch die Iteration durchlebt haben. Während beim ersten Durchlauf häufig Abhängigkeiten zwischen unterschiedlichen, aufeinanderfolgenden Phasen nur schwach erkannt und entsprechend nur kaum erkennbar in den Dokumenten zu sehen waren, war in dem zuletzt beginnenden Prozess eine deutliche Verbesserung zu erkennen. Die Qualität der Dokumente hat sich wie erwartet verbessert.

3.2 Technologien

Den Lernerfolg bezüglich der Technologien lässt sich mit Hilfe des entwickelten Produktes feststellen.

Alle Teams haben eine integrierte Entwufsumgebung mit den geforderten Funktionen und Eigenschaften abgegeben. Mit der Vorgabe des Metamodells waren alle Teams in der Lage, ihr Werkzeug zu entwickeln und es in die geforderte Entwurfsumgebung zu integrieren. Die Mehrzahl der Produkte übertraf in Bezug auf den Funktionsumfang und die Stabilität unsere Erwartungen.

Letztendlich hatten zwei von 16 Teams zu der gegebenen Deadline für das eigene Werkzeug noch keine lauffähige Version. Eine Woche später konnte eines der Teams sein Werkzeug nachliefern. Damit konnte dieses Team rechtzeitig bis zur Produktmesse sein Werkzeug fertigstellen und dort präsentieren. Das andere Team musste mit einem fremden Werkzeug das Praktikum fortsetzen.

Weiterhin konnten wir den Lernerfolg in Bezug auf die eingesetzten Technologien durch Zwischenpräsentationen beobachten. Hierbei wurde ein sehr gutes Verständnis der eingesetzten Technologien festgestellt. Dies wurde letztendlich durch die Feedbackbögen am Ende der Veranstaltung bestätigt. Alle Teams hatten den Lernerfolg bezüglich der Technologien im Mittel mit gut bis sehr gut eingestuft. Eine positive Tendenz ließ sich ebenfalls durch die wöchentlichen Treffen mit den Tutoren ziehen.

3.3 Reengineering

Etwa nach der Hälfte des Praktikums mussten die Teams aus ihrem selbst entworfenen Plug-in und mit zwei weiteren Plug-ins eines anderen Typs eine integrierte Entwurfsumgebung entwickeln. Wir haben hierbei sehr hohen Wert auf gleiche Benutzbarkeit und Funktionalität (z.B. Konsistenzmanagement einheitlich durch restriktive Eingaben, einheitlich durch Fehlermeldungen etc.) der verschiedenen Editoren gelegt. Damit die Studenten diese Aufgabe erfolgreich bestehen, mussten sie mit den eingekauften Produkten ein Reengineering durchführen. Die entsprechenden Dokumentabgaben hierzu waren im Durchschnitt nur befriedigend. Da das Reengineering nur in einem Prozess durchgeführt wurde, kann eine Verbesserung über den Zeitraum des Praktikums nicht festgestellt werden.

Trotz der durchschnittlichen Abgaben wurden in dem Endprodukt bis auf wenige Ausnahmen die geforderten Eigenschaften umgesetzt. Der selbstbewertete Lernerfolg der Studenten wurde im Schnitt mit gut angegeben.

3.4 Projektmanagement

Beobachtungspunkte für das Projektmanagement waren die Abgaben der Projektpläne und die wöchentlichen Treffen mit den Tutoren. Aus beiden lässt sich durch die parallelen Prozesse erkennen, dass hier eine kontinuierliche Verbesserung stattgefunden hat. Während anfänglich überwiegend alle Teams nur rechtzeitig die Deadlines der Abgaben einhalten konnten, hatte sich im Lauf der Zeit ein verbessertes Management der zu erledigten Arbeiten eingestellt, wodurch sogar die Abgaben einige Tage vor der Deadline erfolgt sind. Die Aufgabe, dass jeder Teilnehmer während des Praktikums möglichst alle Tätigkeiten durchlaufen hat, wurde in der Regel erfüllt. Weiterhin ist positiv zu bewerten, dass die meisten Teams in der Lage waren, sich geeignet für mehrere parallele Aufgaben aufzuteilen, und hierbei auch einen Projektplan erstellt haben. Das Feedback der Teams war zum Projektmanagement gut: Die Teamarbeit war für die Studenten eine gute Erfahrung und ein guter Lernerfolg. Entsprechend war das Projektmanagement für die meisten Teams positiv und ein guter Lernerfolg. Einige Teams merkten in den Feedbackbögen zu knappe Deadlines an. Da diese Termine aber von anderen Teams unterschritten wurden, ist dieser Kritikpunkt eher weniger stark zu gewichten.

3.5 Inhalte und Anwendung der Vorlesung

Um den Lernerfolg bei diesem Thema festzustellen nutzen wir die Beobachtungspunkte Klausur, Feedbackschleifen, Tutorien und Feedbackbögen. Die Klausur wurde im Vergleich zum letzten Jahr besser abgeschlossen. Die Feedbackschleifen zur Vorlesung wurden von den Studenten noch nicht im gewünschten Rahmen genutzt. Hier muss offenbar noch vermittelt werden, welche Vorteile diese Feedbackschleifen für die Umsetzung der Vorlesungstheorie in die Praxis haben. In den Tutorien, wo Vorlesungsstoff praktisch angewandt wurde, sind überwiegend keine Diskussionen aufgekommen. Die Tutoren berichteten zum Teil, dass die Inhalte und Anwendung der Tutorien, eine Woche nach der dazugehörigen Vorlesung, nicht immer präsent bei den Studenten waren. Hier wird erneut deutlich, dass die Feedbackschleifen zur Vorlesung stärker ins Bewusstsein der Studenten gerückt werden müssen. Trotz dieser Schwierigkeiten ist in den Abgaben und Produkten sowie durch die Kontrollen der Teambetreuer deutlich geworden, dass die Inhalte von den Studenten umgesetzt werden konnten. Hierzu passen dann auch die überwiegend positiven Feedbackbögen.

Der allgemeine Lernerfolg wurde durch die Studenten mit gut bewertet. Durch die Resümees wurde auch deutlich, dass der Vorlesungsstoff interessant und hilfreich war.

3.6 Qualitätssicherung

Zur Qualitätssicherung wurden Reviews und Inspektionen für die Dokumenterstellung (Lastenheft, Pflichtenheft, Analyse und Entwurf) verlangt. Während der Implementierungsphase wurden Reviews und JUnit-Tests mit Überdeckungsbericht mit Hilfe von Emma zur Qualitätssicherung eingesetzt. Die Reviews und Inspektionen wurden innerhalb der Teambesprechung thematisiert bzw. durchgeführt. Hier lässt sich ähnlich wie bei den anderen Tätigkeiten, die mehrfach aufgrund der parallelen Prozesse durchgeführt wurden, eine Verbesserung erkennen.

3.7 Soft Skills

Hierunter betrachten wir, ob und inwiefern Studenten teamfähig sind und wie sie sich in verschiedenen Rollen, wie zum Beispiel in der eines Verkäufers, präsentieren können. Um die Teamfähigkeit zu bewerten, haben wir die Möglichkeit genutzt, entsprechende Informationen von den Tutoren zu erfragen. Weiterhin haben wir in den Fragebögen bei der Endabgabe diesen Punkt sowohl in dem Bereich des Team- als auch im Bereich des Einzelfeedbacks erhoben. Bis auf zwei Teams kann hierbei von einem positiven Lernerfolg gesprochen werden. Hierbei haben die Berichte der Tutoren mit den Fragebögen korreliert. Bei den zwei Teams, in denen die Fähigkeiten im Bereich der Soft Skills negativ beurteilt wurden, ist ebenfalls eine Korrelation mit dem erstellten Produkt zu erkennen.

3.8 Vergleich mit vorherigen Praktika

Ziel der Umstellung ist, das bisherige Praktikum zu verbessern. Berücksichtigt man die Vorgaben im Bereich der Lernziele und die gewählten Evaluationsmaßnahmen, ist festzustellen, dass tatsächlich eine Verbesserung erzielt werden konnte. Die parallelen Prozesse haben sich sehr positiv auf den Lernerfolg für zentrale Themen der Softwaretechnik, wie beispielsweise systematisches Vorgehen vom Lastenheft bis hin zur Implementierung, ausgewirkt.

Der Organisationsaufwand im Vergleich zu den vorherigen Jahren ist gestiegen. Durch die parallelen Prozesse sind zusätzliche Abgaben angefallen, die kontrolliert werden mussten, und zudem ist der Aufwand, die parallelen Prozesse zu koordinieren, größer geworden.

Der Korrekturaufwand ist gestiegen, da nun die dreifache Anzahl von Dokumenten geprüft werden musste. Da der Umfang der Dokumente deutlich geringer war als im vorherigen Jahr, relativierte sich dieser Aufwand.

4 Fazit

Um die Ziele des diesjährigen Softwaretechnikpraktikums zu erreichen haben wir drei parallele Entwurfsprozesse eingeführt, die versetzt zueinander gestartet wurden. Wie in den vorherigen Kapiteln gezeigt, ist dies eine gute Entscheidung, da hierdurch ein größerer Lernerfolg bei den Studenten erreicht wird. Die zusätzliche Herausforderung, eine komplexe Technologie zu erlernen, gelingt ebenfalls. Es hat sich allerdings herausgestellt, dass beim Reengineering ein geringerer Lernerfolg erreicht wurde, der aber noch akzeptabel ist. Aufgrund der neuen Aufgabenstellung war es nicht möglich, von Anfang an ein geeignetes Legacy-System bereitzustellen. Für die folgenden Praktika stellen die diesjährigen Entwurfsumgebungen eine gute Grundlage für ein anspruchsvolleres Reverse Engineering dar.

Bezüglich des Arbeitsaufwands der Betreuung lässt sich abschließend folgern, dass dieser nicht proportional mit den parallelen Prozessen gestiegen ist. Im Verhältnis zu den erreichten Zielen ist der erhöhte Aufwand vertretbar.

Der Erfolg des diesjährigen Praktikums hat uns dazu bewogen, auf Basis dieser Konzeption das Folgepraktikum zu gestalten.

Literatur

[Bot01] Klaus Bothe, Ulrich Sacklowski: Praxisnähe durch Reverse Engineering-Projekte: Erfahrungen und Verallgemeinerungen. Software Engineering in den Hochschulen, SEUH 7, dpunkt, 2001.

[Dem99] Birgit Demuth, Heinrich Hussmann, Lothar Schmitz, Steffen Zschaler: Erfahrungen mit einem frameworkbasierten Softwarepraktikum. In: Tagungsband des 6. Workshops Software-Engineering im Unterricht der Hochschulen, Teubner-Verlag, 1999.

[Gam95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[Ecl06] Eclipse Projekt Homepage, *http://www.eclipse.org*.

[Emf06] Eclipse Modeling Framework Project Homepage, *http://www.eclipse.org/emf*.

[Gef06] GEF Projekt Homepage, *http://www.eclipse.org/gef*.

[Geh02] M. Gehrke, H. Giese, U.A. Nickel, J. Niere, M. Tichy, J.P. Wadsack, A. Zündorf: Reporting about Industrial Strength Software Engineering Courses for Undergraduates. In: Proc. of the 4th International Conference on Software Engineering (ICSE), Orlando, Florida, USA, 95–405, 2002.

[Geh03] Matthias Gehrke, Holger Giese, Ekkart Kindler, Jörg Niere, Wilhelm Schäfer, Jörg P. Wadsack, Robert Wagner, Lothar Wendehals: Software Engineering Education: The Synergy of Combined Research and Teaching, Tech. Rep. tr-ri-03-237, University of Paderborn, Paderborn, January 2003.

[Kop04] Corina Kopka, Doris Schmedding, Jens Schröder: Der Unified Process im Grundstudium – Didaktische Konzeption, von Lernmodulen und Erfahrungen. DeLFI 2004: 127-138.

[RMI06] Remote Method Invocation, *http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp*.

[Upb06] Modulhandbuch Bachelor-Master-Studienprogrammm Informatik. Fakultät für Elektrotechnik, Informatik und Mathematik an der Universität Paderborn, Juni 2006, *http://wwwcs.uni-paderborn.de/cs/studium/material/modulhandbuch.pdf*, 2006.

[Swt06] Homepage Softwaretechnikpraktikum, Universität Paderborn, Fachgebiet Softwaretechnik: *http://ag-schaefer.upb.de/Lehre/Lehrveranstaltungen/Praktika/Softwaretechnikpraktikum/SS06*.