

Einsatz von Standardprozessen bei der Gestaltung von Lehrveranstaltungen

Niko Kleiner, Stefan Sarstedt
Abteilung Programmiermethodik und Compilerbau
Fakultät für Informatik, Universität Ulm
89069 Ulm
email: {niko,stefan}@bach.informatik.uni-ulm.de

Zusammenfassung

Dieser Artikel beschreibt, wie mit Hilfe von Standardprozessen ein Softwarepraktikum im Hauptstudium an der Universität Ulm gestaltet wurde. Dieses Praktikum sollte nicht nur Entwicklungstätigkeiten schulen, sondern insbesondere auch auf begleitende Tätigkeiten wie Konfigurations- und Qualitätsmanagement eingehen. Der Leser erhält einen kurzen Überblick über den Aufbau des Praktikums, die Durchführung im SS2000 und einen Einblick in die dabei gewonnenen Erfahrungen.

1. Einleitung

1.1 Motivation und Lehrziele

Auf dem Markt befinden sich inzwischen eine Reihe von Standardprozessmodellen für die Softwareentwicklung [6,7]. Die Industrie verspricht sich von diesen Modellen kompaktes Erfahrungswissen großer Organisationen, das als Vorlage für die eigene Firma dienen kann. Dabei bieten diese Modelle in mehrerer Hinsicht Hilfe: Einerseits geben sie an, wann etwas zu tun ist (Prozess) und andererseits bieten sie Hilfestellung, wie man diese Einzelschritte am effizientesten ausführt (Methodik). Dabei unterscheiden sich Prozessmodelle in Art und Aufbau der Dokumentation. Das V-Modell 97 [1] trennt z.B. die Aspekte Prozess und Methodik völlig und ist damit methodenunabhängig; der Rational Unified Process [8] (kurz: RUP) hingegen ist methodenspezifisch, indem er direkt eine OO-Methodik integriert.

Die Anwendbarkeit dieser Modelle wird oft diskutiert [6]. In diesem Artikel beschreiben wir, wie wir an der Uni Ulm im SS2000 ein Softwarepraktikum im Hauptstudium mit Hilfe von Standardprozessen gestaltet haben und welche Erfolge und Probleme wir dabei im Hinblick auf die Lehre erfahren haben. Im Einzelnen haben wir dabei den Schwerpunkt auf folgende Fragen gelegt:

1. Sind Standardprozessmodelle als Vorlage für eine Lehrveranstaltung geeignet?
2. Ist die methodische Unterstützung ausreichend?
3. Inwieweit verbessert der Einsatz eines Standardprozesses die Teamarbeit und damit die Qualität der erstellten Dokumente?

Die letzte Frage ergab sich aus dem Gedanken, Teamaspekte in Prozessen mit zu berücksichtigen. Gerade bei verteilter Zusammenarbeit wird diesem Faktor zu wenig Rechnung getragen.

Neben diesen Fragestellungen soll in diesem Artikel auch darauf eingegangen werden, inwiefern diese Lehrform unseren Lehrzielen (siehe Ende des Abschnitts) zuträglich war. Den Studenten sollte eine Möglichkeit geboten werden, umfassend die sogenannten „best practices“ der Softwareentwicklung ohne Zeit- und Kostendruck auszuprobieren und in die Lage versetzt werden, sich eine auf eigenen Erfahrungen basierende Meinung über deren Anwendbarkeit bilden zu können. Wir formulierten drei Lehrziele:

1. Es sollte die Notwendigkeit entwicklungsbegleitender Tätigkeiten wie Qualitäts- und Konfigurationsmanagement verdeutlicht und verstanden werden.
2. OO-Modellbildung und Abstraktion sollten geschult werden.
3. Die Grundregeln von Reviews sollten erlernt werden. Insbesondere sollte hierbei erfahren werden, dass nicht nur die technischen Aspekte eines Reviews von Nutzen sind, sondern auch die Auswirkungen auf das Entwicklungsteam selbst.

1.2 Einbettung in das Curriculum

Während des Informatik-Grundstudiums an der Universität Ulm besuchen alle Studenten eine „Einführung in die Konzepte der Objektorientierung und des Softwareentwurfs“. Im Anschluss daran wird während des 3. Fachsemesters ein „Softwaregrundpraktikum“ durchgeführt. Als Lehrziele stehen dabei die Vermittlung erster Kenntnisse und Erfahrungen bei der methodischen Entwicklung großer Software-Systeme im Team im Vordergrund [9]. Aspekte wie Qualitätssicherung und Konfigurationsmanagement sind zweitrangig.

Als Grundlage für weiterführende Veranstaltungen im Hauptstudium dient die Vorlesung „Softwaretechnik“. Sie soll einen möglichst umfassenden Überblick über Themen des Software Engineering geben. Darauf aufbauend gibt es zur Vertiefung Spezialvorlesungen zu den Themen „Requirements Engineering“, „Software Qualität“ sowie „Management von Softwareprojekten“. Zur praktischen Umsetzung des Erlernten dienen schließlich das in diesem Bericht angesprochene Praktikum „Objektorientierte Softwarekonstruktion“ sowie das Praktikum „Experimentelles Software Engineering“, bei dem es hauptsächlich um die Modellierung des Ablaufs und die Untersuchung von Prüfverfahren eingebetteter Systeme geht [2].

Alle Veranstaltungen stehen sowohl im Rahmen des traditionellen Studiengangs Informatik als auch des vor kurzem eingeführten, auf einem Leistungspunktesystem basierenden Bachelor-/Master-Studiengangs zur Verfügung [3].

2. Aufbau des Praktikums

In diesem Abschnitt wird der Hintergrund vermittelt, unter dem das Praktikum durchgeführt wurde. Dazu beschreiben wir kurz das Projekt, den von uns vorgegebenen Prozess und die tatsächliche Durchführung während des Semesters.

2.1 Projektbeschreibung LuxOR

Im Vorfeld des Praktikums wurde zunächst nach möglichen Themen für das Projekt gesucht. Das zu erstellende System sollte sich von den üblichen universitären Problemstellungen wie beispielsweise „Literaturdatenbank“ oder „Bibliotheksverwaltung“ abheben und real sein, um die Teilnehmer zu motivieren. Schließlich haben wir uns auf ein Reservierungs- und Verkaufssystem für Kinokarten, genannt LuxOR (Lux Online-Reservierungssystem¹), entschieden. Die Hauptmotivation für die Entwicklung eines solchen Systems liegt in der Verbesserung des Kundenservices des Kinos, der Möglichkeit der Sicherung eines festen Kundenstamms und einer wesentlichen Entlastung der Kassenmitarbeiter. Einige wichtige Features von LuxOR sind:

- Abrufen des Kinoprogramms über das Internet.
- Graphische Anzeige der Kinosäle und darauf interaktive Auswahl von Plätzen.
- Reservierung und Kauf von Kinokarten über das Internet oder an der Abendkasse.
- Verwaltungsfunktionen für Filme, Kinoprogramm, Kundenstamm etc.

2.2 Prozessübersicht

Aufgrund fehlender Erfahrungsberichte über ein Praktikum, in dem Standardprozesse zum Einsatz kamen, musste die Lehrveranstaltung von Grund auf neu gestaltet werden. Um die Risiken eines solchen Neustarts zu mildern, wurde eine Vorplanungsphase von drei Monaten und ein Probelauf von nochmals zwei Monaten eingeplant und durchgeführt. In der Vorplanung wurde aus dem RUP [8] und der Methode von Craig Larman [4] ein möglichst einfacher, aber vollständiger OO-Entwicklungsprozess extrahiert (genauerer dazu folgt im Abschnitt 3.4).

¹ Lux ist der Name des Zusammenschlusses mehrerer lokaler Kinos

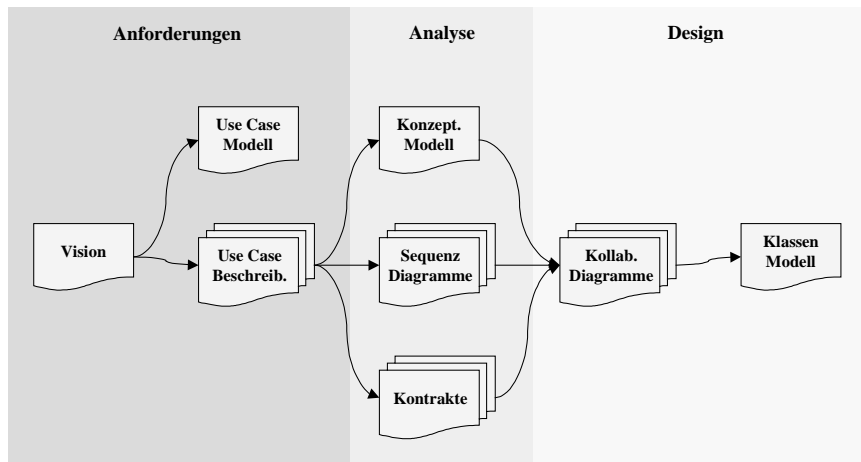


Abbildung 1: Dokumentenfluss

Abbildung 1 zeigt die Dokumente, die während des Prozesses zu erstellen waren (aus Platzgründen verzichten wir auf eine detaillierte Beschreibung des Prozesses). Pfeile geben Abhängigkeiten zwischen den Dokumenten an. Zuerst wurde ein Pflichtenheft (ein sogenanntes „Vision“-Dokument in der RUP Terminologie) erstellt, welches eine Problembeschreibung, die Beschreibung der Interessengruppen und Features des zu entwickelnden Systems enthielt. Daraus wurde ein Use Case Modell und Beschreibungen für die einzelnen Use Cases abgeleitet. Zur Analyse wurde ein konzeptuelles Modell erarbeitet, der Ablauf der Use Cases in Sequenzdiagrammen formalisiert und die dadurch gefundenen Systemoperationen mittels Kontrakten (Beschreibung der Vor- und Nachbedingungen) spezifiziert. Jede Systemoperation wurde schließlich durch eine Kollaboration realisiert, die angab, wie verschiedene Objekte zusammenarbeiten, um die Systemoperation durchzuführen. Aus den Kollaborationen konnte dann ein Klassenmodell abgeleitet werden. Die Kollaborationen und das Klassenmodell beschrieben zusammen das Design.

2.3 Organisation und Durchführung

Für die Durchführung des Praktikums standen uns im Sommersemester 13 Sitzungen zur Verfügung. Es wurden nur Modelle erstellt und auf die Implementierung verzichtet, da letztere erfahrungsgemäß zeitraubend ist und nichts zu den in diesem Praktikum angestrebten Lernzielen beigetragen hätte. Dieses Vorgehen wurde von Seiten der Studenten sehr honoriert.

Das Betreuerteam bestand aus drei Doktoranden und drei wissenschaftlichen Hilfskräften (kurz: Hiwis). Es nahmen neun Studenten teil, die in drei Dreiergruppen aufgeteilt wurden und einen Hiwi zur Seite gestellt bekamen. Alle arbeiteten an demselben Projekt.

Aus dem Vorlauf wurde eine Paketaufteilung des Projekts übernommen, so dass jedes Team an einem separaten Paket parallel arbeiten konnte.

Tabelle 1 zeigt die Abbildung des Prozesses auf die 13 Sitzungen (tatsächlicher Verlauf). Die Wochenaufgabe zu einer Sitzung gibt an, welche Aktivitäten mit welchen Ergebnissen im Laufe der Woche bis zur nächsten Sitzung durchzuführen bzw. zu erstellen waren.

Sitzung	Wochenaufgabe	Aufwand (in Std.)
1	Einarbeitung	7-9
2	Interviews, Vision erstellen	7-9
3	Use-Case Modell, Kurzbeschreibungen für Use-Cases	6-8
4	Ablaufbeschreibungen für Use-Cases	8-10
5	Konzeptuelles Modell pro Team, Sequenzdiagramme	6-8
6	Review vorbereiten	3-5
7	Konzeptuelle Modelle zusammenführen, Korrekturen aus Review	4-6
8	Kontrakte	8-10
9	Kollaborationen, Klassenmodell pro Team	10-12
10	Review vorbereiten	4-6
11	Klassenmodelle zusammenführen, Korrekturen aus Review	4-6
12	Abschlusspräsentation vorbereiten	3-5

Tabelle 1: Grober Projektplan (tatsächliche Durchführung)

Das konzeptuelle Modell und das Klassenmodell wurden zunächst von den Gruppen für ihre Pakete separat erstellt (in Wochenaufgabe 5 und 9) und mussten dann zu einem gemeinsamen Paket zusammengeführt werden (in Wochenaufgabe 7 und 11). Die Reviews wurden so platziert, dass jeder Teilnehmer mindestens ein Modell eines anderen Teams inspizieren musste und somit eine Übersicht über die Arbeit der Kommilitonen bekam. Dies sollte die Erstellung des Gesamtmodells erleichtern. So waren im ersten Review (7. Sitzung) das Use Case Modell, die Use Case Beschreibungen, die Formalisierungen davon in Form von Sequenzdiagrammen und die von den einzelnen Gruppen erstellten konzeptuellen Modelle zu inspizieren. Im zweiten Review (11. Sitzung) wurden Kontrakte, Kollaborationen und die einzelnen Klassenmodelle untersucht.

Als Konfigurationsmanagementwerkzeug setzten wir Rational ClearCase, für das Änderungsmanagement Rational ClearQuest und zur Modellierung Rational Rose ein.

3. Erfahrungen

3.1 Vorplanung und Probelauf

Sowohl die Vorplanung als auch der Probelauf haben sich in Bezug auf die resultierende Qualität der Lehre gerechnet. Rückblickend waren dabei folgende Erfahrungen am wichtigsten:

- Bei einem völlig neu aufzusetzenden Praktikum ist es zu Beginn beinahe unmöglich, die Aufwände für einzelne Aktivitäten abzuschätzen. Durch einen Probelauf erhält man erste Ausgangsdaten. Dadurch, dass wir im Probelauf den Hiwis den Prozess und die angewandten Methoden näher bringen mussten, erhielten wir auch Hinweise, an welchen Stellen besondere Verständnisschwierigkeiten auftreten und konnten dies bei der nachfolgenden Planung bereits berücksichtigen.
- Ein vom gesamten Betreuersteam durchgeführter Probelauf (ohne Vorplanung) bringt ein gemeinsames Verständnis des Projekts, des Prozesses und der Methoden. Widersprüchliche Aussagen und Meinungen von der Betreuerseite werden dadurch während der Durchführung des Praktikums von vornherein vermieden.
- Die Werkzeugumgebung ist vor dem Praktikum bereits getestet. Damit wird unnötige, durch Probleme mit einem Werkzeug verursachte Zeitverschwendung vermieden.

3.2 Durchführung

Bei der Durchführung hat sich die parallele Erstellung von Modellen als Hauptproblem herausgestellt (Konzeptuelles Modell, Klassenmodell, siehe Abschnitt 2.3). Obwohl Reviews vor einem Zusammenführen von Teilmodellen zu einem Ganzen so platziert und organisiert wurden, dass jeder Teilnehmer mindestens ein Modell eines anderen Teams inspizieren musste und damit genügend Wissen für die Erstellung des Gesamtmodells haben sollte, gelang es den Teilnehmern nicht, ein homogenes und verständliches Gesamtmodell zu erstellen. Wesentliche Mängel waren dabei:

- Es wurde kein gemeinsames Abstraktionsniveau gefunden.
- Die Teilnehmer haben zu wenig auf Verständlichkeit geachtet und waren zu sehr auf eine möglichst vollständige Beschreibung aus.
- Es wurden Teile aus den Ursprungsmodellen einfach weggelassen, weil sie bei der Zusammenführung hinderlich waren (diese wurden in späteren Phasen dann wieder vermisst).
- Die Teilnehmer haben bei Überlappungen um Positionen gefeilscht, anstatt die Vereinigung aller Teilmodelle als gemeinsames Ziel zu begreifen.

Diese Schwierigkeiten traten im Probelauf nicht auf, sondern erst bei einer parallelen Arbeitsaufteilung. Auch Konfigurationsmanagement und ein Notationstreue

erzwingendes Werkzeug wie Rose können diese Probleme nicht lösen, da diese semantischer und nicht syntaktischer Natur sind.

3.3 Sitzungen

Während der wöchentlichen Sitzungen wurden mehrere Workshops durchgeführt. Wir haben festgestellt, dass die Diskussion in der Gruppe schon bei 9 Personen eher zäh verlief. Deshalb änderten wir im Laufe des Praktikums die Organisation solcher Workshops: Jede Gruppe musste einen temporären Teamleiter bestimmen, der jeweils alleine zur Sitzung kam. Die Teamleiter wurden dann geschult und mussten zusammen mit ihrem Hiwi das Gelernte an das eigene Team weitergeben. In jeder Sitzung wechselte die Teamleiterrolle. Von Seiten der Studenten wurde diese Form als motivierend und teamstärkend empfunden.

3.4 Prozessgestaltung

Rational Unified Process

Zunächst war geplant, als Prozessvorlage auf das V-Modell 97 zurückzugreifen, was sich jedoch schnell als nicht handhabbar herausstellte. Insbesondere ist eine Anpassung und Durchführung für ein konkretes Projekt unserer Meinung nach nur mit Werkzeugunterstützung effizient zu schaffen. Schließlich haben wir uns entschlossen, eine eigens angepasste Version des Rational Unified Process zu verwenden. Dabei haben uns die folgenden Punkte die Gestaltung des Praktikums erleichtert:

- Die HTML-Dokumentation erleichtert das Finden von Informationen wesentlich.
- Alle wichtigen Dokumente liegen in Form von Templates vor (z.B. die Use Case Beschreibungen), wobei neben der Struktur auch Hinweise auf die jeweiligen Inhalte der einzelnen Abschnitte vorhanden sind. Die Templates waren besonders hilfreich, da die Teilnehmer sich dadurch keine Gedanken über die Struktur machen mussten. Des weiteren wurden die Reviews (vgl. Abschnitt 3.5) durch gruppenübergreifend standardisierte Dokumente wesentlich vereinfacht.
- Es werden Arbeitshilfen in Form von Anleitungen für Workshops, Interviews sowie Review- und Brainstormingsitzungen gegeben. Darauf haben wir insbesondere beim Finden der Use Cases zurückgegriffen.

Während sich der RUP in den frühen Entwicklungsphasen (speziell bei der Anforderungserfassung und -dokumentation) als sehr hilfreich erwiesen hat, stellte sich die weitere Verwendung in der Analyse- und Designphase als zusehends schwierig heraus. Dies hatte u.a. folgende Gründe:

- Die Beschreibung der Analyse- und Designphase wurde als nicht mehr hinreichend praktikable Hilfestellung empfunden. Die Beispiele in der RUP-Dokumentation sind nicht durchgängig und größtenteils zu trivial.
- Als verwirrend hat sich im Verlauf von Vorbereitungs-Workshops die Unterscheidung der Analyse-Objekte in Boundary-, Controller- und Entity-Objekte herausgestellt. Diese Differenzierung ist unserer Meinung nach für den Aufbau eines Verständnisses über objektorientierte Analyse und Entwurf kontraproduktiv, da die Gefahr besteht, dass z.B. Controller-Objekte zu einer eher funktionsorientierten und Entity-Objekte zu einer datenorientierten Sichtweise des Systems verleiten. Hinweise auf die Wichtigkeit einer adäquaten Zuweisung von Verantwortlichkeiten werden im RUP nicht gegeben.

Nicht zu vernachlässigen ist auch die enorme Einarbeitungszeit für einen so komplexen Entwicklungsprozess. Aufgrund der Nachteile wurde im Verlauf der Planungsphase klar, dass eine Verwendung der Analyse- und Designkapitel des RUP für unsere Zwecke nicht in Frage kam.

Larman

Für die weiteren Phasen erwies sich das von Craig Larman vorgeschlagene Vorgehensmodell [4] als sinnvoll. Das Buch enthält eine umfangreiche und leicht verständliche Einführung in objektorientierte Analyse und Design. Alle Aktivitäten werden anhand eines durchgängigen Beispiels erläutert. Besonders nützlich für die Vorbereitung und Durchführung unserer Lehrveranstaltung waren folgende Punkte:

- Larman geht auf die unterschiedlichen Sichtweisen von Use Case Beschreibungen ein. Dies war insbesondere wichtig, weil anfangs nicht klar war, ob überhaupt Oberflächen-Aspekte in Use Cases beschrieben werden sollen. Den richtigen Abstraktionsgrad bei Use Cases zu finden war ein oft diskutiertes Problem während der Vorbereitung des Praktikums.
- Als ideal haben sich Larman's „GRASP Patterns“ (General Responsibility Assignment Software Patterns) erwiesen. Die Patterns geben Hinweise, wie Verantwortlichkeiten „richtig“ (im OO-Sinne) auf die Objekte verteilt werden.

Die Beschreibung der Anfangsphasen der Softwareentwicklung ist bei Larman relativ dünn, weshalb dieser Teil fast vollständig vom RUP übernommen wurde.

3.5 Reviews

Als Qualitätssicherungsmaßnahme wurden begleitend zum Praktikum Reviews durchgeführt. Dort mussten sich die Teilnehmer mit den Meinungen anderer zu ihren Dokumenten und Modellen auseinandersetzen, was auch im Feedback als besonders positiv empfunden wurde. Durch ein Rotationsprinzip war jeder Teilnehmer einmal Inspektor bzw. Dokumentenersteller, die Inspektoren waren grundsätzlich immer aus den anderen Praktikumsgruppen. Reviews waren aus den folgenden Gründen sehr hilfreich für die Teilnehmer und den Verlauf des Praktikums:

- Durch die Sitzungen konnte immer festgestellt werden, ob die Qualität der Dokumente ausreichend für die nächsten Prozessschritte war, oder ob eine Überarbeitung notwendig wurde.
- Die Reviews halfen den Inspektoren, ein Verständnis über die Arbeit der anderen Gruppen und somit über das Gesamtsystem aufzubauen.
- Das Feedback der Inspektoren wiederum gab dem Dokumentenersteller nützliche Hinweise auf Fehler und Optimierungsmöglichkeiten.
- Sie waren eines der wenigen Mittel zur Einzelbewertung der Teilnehmer.

Reviews sollten formal, organisiert und mit Moderatoren (Praktikumsbetreuer oder Hiwis) durchgeführt werden. Von den Studenten selbst durchgeführte Reviews verfehlen meist die inhaltlichen Ziele [2].

4. Fazit

Rückblickend können wir die zu Anfang aufgeworfenen Fragen nun folgendermaßen beantworten:

Der Rational Unified Process half einen Rahmen für das Praktikum zu stecken (was sind die Inhalte, welche Workshops etc.) und bot für einzelne Aktivitäten, wie z.B. den Use Case Workshop, gute Checklisten und Hinweise. Außerdem erleichtern die Dokumentenvorlagen (Templates) die Vorbereitung wesentlich. Allerdings ist der Aufwand, um einen ersten, übersichtlichen Prozess zu gestalten, zu hoch. Es werden auch keine Hinweise gegeben, wie man einen Minimalprozess aus dem generischen Modell gewinnen kann. Prozessmodelle würden vielleicht eine größere Akzeptanz erfahren, wenn sie gerade umgekehrt aufgebaut wären, d.h. ein Minimalprozess vorgegeben würde und an entsprechenden Stellen Hinweise auf problem-spezifische Erweiterungsmöglichkeiten gegeben würden.

In der methodischen Unterstützung ist der RUP eher dünn. Insbesondere kommt man nach der Use Case Modellierung kaum mehr ohne weiterführende Dokumentation voran. An dieser Stelle schließt Larman sehr gut an den RUP an.

Ein Standardprozessmodell hilft vorab nicht unbedingt, Teamarbeit zu unterstützen und besser zu koordinieren. Erst wenn ein schlanker, konkreter Prozess erstellt und an alle in einer verständlichen Form kommuniziert wird, hilft er den Beteiligten,

eine bessere Übersicht zu bekommen und verbessert dadurch auch die Zusammenarbeit im Ganzen. Allerdings bleiben Probleme der Art, wie sie in Abschnitt 3.2 und 3.3 diskutiert wurden, bestehen.

Die gewählte Lehrform war für die Erreichung der Lehrziele erfolgreich. Durch den Einsatz von Konfigurationsmanagement und Reviews wurde den Studenten hinreichend klar, dass Softwareentwicklung im Großen ohne entwicklungsbegleitende Tätigkeiten nicht zu schaffen ist. Für die Schulung von OO-Modellbildung und Abstraktion hat der RUP selbst keine Hilfe geleistet. Larman's GRASP-Patterns schlagen unserer Meinung nach für diesen Zweck die richtige Richtung ein. Durch den Einsatz von Reviews und die damit verbundene Beschäftigung der Teilnehmer mit der Arbeit der Anderen werden oft erst die Grundlagen für Diskussionen gelegt, die die wesentlichen Probleme an den Tag befördern.

Die Organisation des Praktikums hat sich im Großen und Ganzen also bewährt und ausreichend Antworten für unsere Fragen abgeworfen. Mit dem nun vorhandenen Prozess, den Vorlagen und den erhobenen Aufwandsdaten können wir für folgende Semester relativ schnell und genau ein Folgepraktikum planen. Auch wenn die Aufspaltung eines Projekts und die Bearbeitung von Teilsystemen mit je einer Gruppe (vgl. auch Abschnitt 2.3) wesentlich mehr Koordinationsaufwand und Probleme mit sich bringt (siehe insbesondere Abschnitt 3.2), würden wir in Folgeveranstaltungen wieder diese Form wählen. Zwar leidet die Qualität des Ergebnisses darunter (z.B. inhomogene Modelle), jedoch erkennen die Studenten unserer Meinung nach nur so diese Probleme und lernen daraus nachhaltig. Allerdings würden wir weniger Betreuer einsetzen, da 6 Betreuer auf 9 Studenten mehr Koordinationsaufwand als Nutzen ist. Als bleibende Schwierigkeiten sehen wir den Umfang und die Detailliertheit der begleitenden Literatur und insbesondere einen objektiven Maßstab für eine Einzelbewertung.

Literatur

1. Dröschel, W.; Wiemers, M.: Das V-Modell 97. Oldenbourg Verlag, München, 1999.
2. Ernst, D.; Schulte, W.; Houdek, F.; Schwinn, T.: Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme. Ulmer Informatik Berichte Nr. 97-13, 1997.
3. Gehring, W.: Ein Rahmenwerk zur Einführung von Leistungspunktesystemen. Ulmer Informatik Berichte Nr. 2000-04, 2000.
4. Larman, C.: Applying UML and Patterns. Prentice Hall, 1997.
5. Latinen, M.; Boddie, J.: Scaling Down Is Hard to Do / Do We Ever Really Scale Down?. IEEE Software, Band 17, Nummer 5, S.78-81, 2000.
6. Müller-Ettrich, G.: Objektorientierte Prozessmodelle. Addison-Wesley, 1999.
7. Noack, J.; Schienmann, B.: Objektorientierte Vorgehensmodelle im Vergleich. Informatik Spektrum 22, S.166-180, 1999.
8. Rational Unified Process. <http://www.rational.com>
9. Schwarz, M.; Schulte, W.: Realistische Aufgabenstellungen für das Softwaregrundpraktikum. Berichte des German Chapter of the ACM, Band 48, S.94-104, B.G. Teubner, Stuttgart, 1997.