Teamtraining für Software-Ingenieure

Andreas Fleischmann, Katharina Spies
Institut für Informatik, Technische Universität München
Boltzmannstraße 3, 85748 Garching b. München
fleischa@in.tum.de, spiesk@in.tum.de

Katharina Neumeyer
Institut für Psychologie, Technische Universität Darmstadt
Hochschulstraße 1, 64289 Darmstadt
training@neumeyer-kroeger.de

Zusammenfassung

Die Notwendigkeit, für die Ausbildung in Software-Engineering passende Unterrichtsformen zu entwickeln, die es den Studierenden ermöglichen, praktische Erfahrungen zu sammeln, ist eine mittlerweile anerkannte Herausforderung bei der aktuellen Konzeption der Lehre von Software Engineering an der Hochschule. Um den Studierenden solche Erfahrungen mitzugeben, werden an vielen Hochschulen Projekte durchgeführt, in denen studentische Teams weitgehend selbstständig einen Software-Engineering Prozess durchleben. Aufgrund unserer Erfahrungen bei der Durchführung solcher Projekte an den TU's Darmstadt und München ist festzustellen, dass die größten Herausforderungen für die beteiligten Studierenden dabei weniger in der Bewältigung technischer oder fachlicher Fragestellungen liegen: diese Problematik ist ihnen aufgrund ihrer Lernerfahrung im Studienverlauf bereits geläufig. Vielmehr stellt der Umgang mit organisatorischen und sozialen Problemen eine viel größere Herausforderung dar, und führt daher oft zu für die Studierenden unvorhergesehenen Problemen im Projekt.

In diesem Artikel stellen wir unsere Erfahrungen mit einem vorbereitenden Teamtraining vor, das wir vor diesem Hintergrund als Vorbereitung zu studentischen Software-Projekten durchgeführt haben. Die Zielsetzung dieses dreitägigen Blockseminars besteht darin, die Studierenden als Team zusammenfinden zu lassen, und ihnen bereits vor Beginn des Projekts grundlegende Techniken für die Zusammenarbeit im Team sowie eine Einführung in Projektmanagement mitzugeben.

1 Einführung

Die Wichtigkeit der praktischen Durchführung von Software-Projekten in der universitären Ausbildung zukünftiger Software-Ingenieure ist mittlerweile erkannt. Dementsprechend werden solche Projekte vielfach durchgeführt und sind bereits an vielen Universitäten fest im Curriculum verankert [Fle 02], beispielsweise an der TU Darmstadt als einjähriges „Software-Projekt“, an der TU München als einsemestriges „Softwaretechnik-Praktikum“, oder an der Uni Stuttgart gleich zweimal als „Softwarepraktikum“ und als „Studienprojekt“.

Solche Projekte zeichnen sich dadurch aus, dass studentische Teams in einen möglichst praxisnahen Setting über einen größeren Zeitraum hinweg weitgehend selbstständig einen Software-Engineering-Prozess durchlaufen, von der Anforderungsanalyse bis hin zur Abgabe eines Produkts. In München beispielsweise treten in dem Bemühen, eine möglichst praxisnahe Projektumgebung zu schaffen, die Veranstalter wie „echte“ Kunden auf. Hierzu gehören auch die Abgabe unvollständiger Aufgabenbeschreibungen, das Setzen von Deadlines und die Forderung nach zusätzlichen Funktionalitäten im fortgeschrittenen Verlauf des Projekts. Durch die Bündelung solcher Herausforderungen und die Größe der Aufgabe entsteht eine vergleichsweise realitätsnahe Situation (vgl. [Tau 01], [Gna 03]), in der die Studierenden anschaulich die Vielseitigkeit des Berufsbildes eines Software-Ingenieurs erleben: Verhandeln, Diskutieren, Organisieren, Design, Implementieren, Testen, Dokumentieren, Präsentieren. Folglich müssen die Studierenden neben technischen und fachlichen Herausforderungen auch organisatorische und soziale Probleme meistern. Beispiele für diese unterschiedlichen Probleme sind:

- Ein technisches Problem ist beispielsweise „Wie spielen die Entwicklungsumgebung Eclipse und die Versionsverwaltung CVS zusammen?“
- Ein fachliches Problem ist beispielsweise „Soll das Programm in der Situation X mit der Aktion A oder B reagieren?“
- Ein organisatorisches Problem ist beispielsweise „Ist es möglich, die anstehenden Aufgaben im Team sinnvoll aufzuteilen?“
- Ein soziales Problem ist beispielsweise „Wie gehen wir als Team damit um, dass Person X die ihm zugewiesene Teilaufgabe unbefriedigend gelöst hat?“

Die Durchführung und Betreuung solcher Projekte an den TU’s Darmstadt und München hat gezeigt, dass die größten Herausforderungen für die beteiligten Studierenden dabei in der Bewältigung organisatorischer/sozialer Probleme liegen. Die Ursache hierfür sollte darin zu finden sein, dass einerseits das benötigte Informatik-Fachwissen in den Vorlesungen bereits hinreichend gut vermittelt wurde, und andererseits darin, dass es den Studierenden aufgrund ihrer eigenen Lernerfahrung deutlich leichter fällt, fachliche Defizite zu überwinden.

Aus unserer Einsicht heraus, dass Studierende früherer Projekte im Rückblick die sozialen und organisatorischen Probleme oft als entscheidend empfanden, und dass während des Projekts wenig Zeit und Bereitschaft besteht, auf dieses „unbequeme

Thema“ in Ruhe einzugehen, wurde an der TU München ein Teamtraining konzipiert , das die Studierenden vor Beginn des Projektes absolvieren, und das sie auf die Zusammenarbeit im Projekt-Team vorbereitet. Damit werden die Studierenden bereits vor dem Start des fachlichen Projekts ohne Termindruck auf den Umgang mit den mit Sicherheit auftretenden organisatorischen und sozialen Problemen vorbereitet, und es werden ihnen Wissen und Techniken für den Umgang und die Bewältigung dieser Probleme an die Hand gegeben.

Im vorliegenden Artikel skizzieren wir zunächst kurz den organisatorischen und inhaltlichen Rahmen eines vor kurzem abgeschlossenen Softwaretechnik-Praktikums an der TU München (Kapitel 2). Danach stellen wir Konzeption, Lerninhalte und Ablauf des Teamtrainings vor (Kapitel 3). Abschließend präsentieren wir unsere bisherigen Erfahrungen mit dem Teamtraining, geben einen Ausblick auf die weitere Entwicklung dieses Trainingskonzepts (Kapitel 4) und schließen mit einem kurzen Fazit (Kapitel 5).

2 Das Softwaretechnik-Praktikum „AutoRAID“

Abb. 1.: Das studentische Team von „AutoRAID“

Am Lehrstuhl Broy werden seit einigen Jahren regelmäßig Softwaretechnik-Praktika (STP) angeboten, in denen Studierende der Informatik im Hauptstudium ein Semester lang selbstständig ein Software-Produkt entwickeln. Aufgabe des STP im Sommersemester 2004 war die Entwicklung von „AutoRAID“, einem prototypischen Werkzeug zur modellbasierten Anforderungsanalyse für das Requirements-Engineering eingebetteter Systeme [Aut 04]. Mit AutoRAID sollen verschiedene Forschungsergebnisse des Lehrstuhls praktisch demonstriert werden, beispielsweise die Erfassung von Requirements durch direkte Eingabe oder Import aus bestehenden Dokumenten, die Klassifikation der Requirements in mehreren Dimensionen und deren anschauliche Darstellung

in einem Werkzeug, automatische Konsistenzanalyse, und das in Beziehung Setzen von Requirements untereinander und zu Design-Elementen (Tracing), inklusive der automatischen Erzeugung von Designelementen.

Das Projekt wurde von wissenschaftlichen Mitarbeitern und externen Firmenvertretern betreut, zum Teil in der Rolle der Kunden, mit denen die Studierenden die Aufgabenstellung verhandelten, zum Teil in der Rolle von Betreuern, an die sich die Studierenden ratsuchend wenden konnten.

Bereits die Ausschreibung der Aufgabenstellung führte zu einer ersten Herausforderung für die Studierenden: Sie mussten sich für die Teilnahme am Praktikum formal bewerben. Bei entsprechender Eignung wurden sie zu einem Auswahlgespräch geladen, das die Grundlage für die Auswahl der Teilnehmer darstellte. Kriterien für die Auswahl von den Studierenden waren dabei neben fachlichen und technischen Kenntnissen, beispielsweise Programmier-Erfahrung, insbesondere auch die Bereitschaft, ein überdurchschnittlich hohes Arbeitspensum zu bewältigen, und in einem selbstorganisierten Team zu arbeiten.

Um den ausgewählten zehn Studierenden das in der Einleitung beschriebene Fundament zur Bewältigung der organisatorischen und sozialen Problematik an die Hand zu geben, und sie vor allem für diese Art der Probleme zu sensibilisieren, fand das Teamtraining noch vor Beginn des Praktikums statt, am ersten Semesterwochenende vom 23. bis 25.April 2004. Das Softwaretechnik-Praktikum begann einige Tage später mit einem Kick-Off am 27.April 2004; an diesem Kick-Off lernten die Studierenden ihre Betreuer und „Kunden“ kennen, bekamen die Aufgabenstellung in Form eines Lastenhefts und einen groben Projektplan, der drei Phasen vorsah: Analyse, Designs und Implementierung.

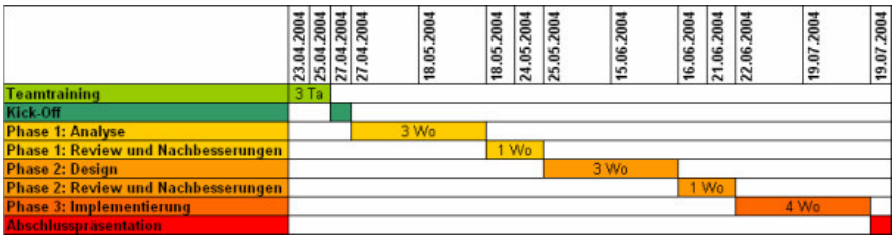

Abb. 2.: Projektplan für das Projekt "AutoRAID"

Abbildung 1 zeigt den Projektplan, der den Studierenden vorgeschlagen wurde. Die Studierenden orientierten sich an diesem Projektplan und schafften es, ihn einhalten, allerdings mussten sie nach der Abschlusspräsentation noch drei Wochen länger arbeiten, um die Dokumentation nachzureichen.

Das Team erstellte innerhalb des vorgegebenen Zeitrahmens das fertige Requirements-Engineering-Werkzeug „AutoRAID“ [Aut 04]. Das Werkzeug ist integriert in ein am Lehrstuhl entwickeltes Design- und Simulationswerkzeug „AutoFOCUS“ [Aut 02], das am Lehrstuhl verwendet und beständig weiterentwickelt wird; eine Fallstudie mit AutoRAID in der Industrie ist für Anfang 2005 geplant.

3 Konzeption, Lerninhalte und Ablauf des Teamtrainings

In diesem Abschnitt stellen wir die Konzeption, die Lerninhalte und den Ablauf des Teamtrainings vor.

3.1 Konzeption

Unsere Zielsetzung bei der Konzeption des Teamtrainings bestand vor allem darin, die Bewältigung der tatsächlichen fachlichen Aufgabe (Entwicklung von AutoRAID) mit dem vorgegebenen Projektplan von Beginn an zielgerichtet zu ermöglichen. Die Studierenden sollten mit Beginn der fachlichen Aufgabe bereits als Team agieren und über die nötige Zusatzqualifikation speziell zur gemeinsamen zielgerichteten Kooperation im Team verfügen.

Das von uns in München durchgeführte Teamtraining basiert auf Konzepten, die seit einigen Jahren von der Hochschuldidaktischen Arbeitsstelle der TU Darmstadt angewandt werden:

- Team-Coaching von Software-Projekten am Institut für Informatik der TU Darmstadt [Hen 02].
- Interdisziplinäres Teamtraining an der TU Darmstadt. Einzelpersonen, die sich vor dem Workshop nicht kannten, und die nach dem Workshop nicht als Team zusammenarbeiten müssen, bilden während des dreitägigen Workshops zu Trainingszwecken ein „ad-hoc“ Team [Kom 04].

Für München wurde das Darmstädter interdisziplinäre Teamtraining für „ad-hoc“ Teams überführt in ein Teamtraining für „echte“ Teams aus Studierenden, die an dem Softwaretechnik-Praktikum teilnehmen werden:

Da es sich nicht um ein „ad-hoc“ Team handelte, sondern um eine Gruppe von Studierenden, die sich im Anschluss an das Training als Team bewähren musste, musste das Training um spezielle Teambuilding-Aspekte ergänzt werden. Der bereits existierende interdisziplinäre Projektmanagement-Anteil wurde auf den spezifischen Kontext des Praktikums zugeschnitten. Im Gegensatz zu Darmstadt war das Teamtraining nicht optional, sondern als verpflichtender Baustein für alle Teilnehmer am STP vorgeschrieben, und wurde mit einem eigenen Seminarschein anerkannt.

Im Jahr 2003 wurde dieses Teamtraining von Katharina Neumeyer und Andreas Fleischmann konzipiert, erstmalig durchgeführt und von Katharina Spies evaluiert. Für AutoRAID wurde das Training dann im Sommersemester 2004 zum zweiten Mal in der hier beschriebenen Form von Andreas Fleischmann durchgeführt.

3.2 Theoretische Grundlage

Unsere Erfahrungen mit Teamarbeit in studentischen Projekten lassen sich mit der „Teamkurve“ (Abbildung 3) vereinfachend zusammenfassen:

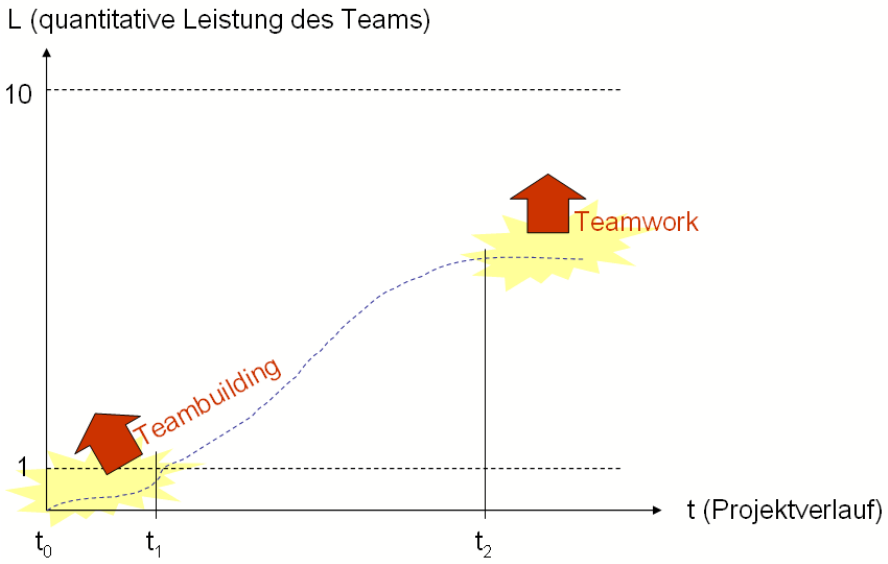

Abb. 3.: Die "Teamkurve"

Während man naiverweise erwarten könnte, dass die zehnfache Personenzahl in einem Team quantitativ zumindest annähernd die zehnfache Leistung erbringt, ist es in der Praxis oft so, dass in einer Anfangsphase (t_0 bis t_1) zehn Personen zusammen weniger leisten als eine Person alleine. In dieser Phase der Teamfindung bilden sich Rollen und Infrastrukturen im Team [Lan 00]. Auch wenn diese Anfangshürde überwunden ist, wird ein Team nie die volle Leistung erreichen (ab t_2), da auch im perfekten Team immer Abstimmung und Koordination nötig ist.

Bei diesen beiden Problempunkten setzen wir im Teamtraining an: Teambuilding soll die unproduktive Anfangsphase der Teamfindung (bis t_1) verkürzen und Frustration vermeiden, und Teamwork-Techniken sollen die Produktivität in den anschließenden Phasen erhöhen (ab t_1, insbesonders ab t_2).

Unserer Erfahrung nach dämpft schon alleine das Wissen um die Teamkurve die (ansonsten zum Teil erhebliche) Anfangsfrustration (Zitat eines Studierenden: „Teamarbeit, das ist, wie wenn sich zehn Leute in einen VW Käfer quetschen, um gemeinsam einzuparken“) erheblich, weil sie nun wissen, dass diese unproduktive Anfangsphase, die sie alle durchlaufen, normal, notwendig und vorübergehend ist.

Auch halten wir es für notwendig, Teamarbeit explizit zu motivieren und gezielt die Vorteile der Teamarbeit herauszustellen, damit sie nicht in den Alltagsproblemen unter-

gehen. Als Argumente für Teamarbeit nennen wir dabei vor allem die letztendlich höhere quantitative Leistung des Teams im Vergleich zu der einer einzelnen Person (in Abbildung 3 ist das ab t_1), und die höhere Qualität, die sich daraus ergibt, dass mehr Leute mehr Ideen haben, aber auch, dass man sich gegenseitig inspiriert, Fehler in der Arbeit der anderen entdeckt usw.

3.3 Ein Team formen (Teambuilding)

Ein Lernziel des Teamtrainings ist es, aus den Studierenden, die sich noch nicht kennen, ein Team zu formen. Der Schwerpunkt des Workshops liegt dabei nicht im expliziten Teambuilding (z.Bsp. Theorie, vielfältige Kennenlernübungen, Vertrauensübungen usw.), wir vertrauten stattdessen darauf, dass sich die Teammitglieder durch die gemeinsame Arbeit im Workshop (insbesondere im Miniprojekt, Abschnitt 3.6) „automatisch“ besser kennen und einschätzen lernen. Das explizite Teambuilding beschränkte sich daher auf folgende Elemente:

- Zu Beginn des Workshops machten wir eine ausführliche Kennenlernrunde, in der jeder Studierende sich den anderen anhand eines „Steckbriefes“ vorstellte (siehe Abbildung 4).

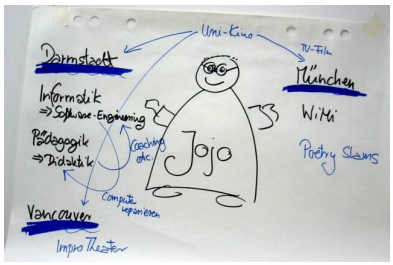

Abb. 4.: Ein typischer "Steckbrief"

- In Diskussionsrunden wurden die Studenten für die Unterschiedlichkeiten innerhalb des Teams und das damit verbundene Problempotential sensibilisiert Dazu diskutieren wir beispielsweise Fragen wie: „Euer Projekt steht unter großem Zeitdruck. Was ist dir wichtiger?“ a) „Die Deadline wird auf jeden Fall eingehalten, ggf. werden Abstriche bei der Qualität zugelassen.“ b) „Das Abliefern eines Qualitätsprodukts hat Vorrang, ggf. muss die Deadline überschritten werden.“
- Bei ganztägigen Blockveranstaltungen empfiehlt es sich, gerade nach der Mittagspause und am Nachmittag hin und wieder auflockernde Übungen zu machen; wir haben dabei Übungen eingesetzt, bei denen die Kooperation im Team im Vordergrund stand (z.Bsp. Reise nach Solidarien [Sen 04], Deckendrehen [Sen 04], Stuhlskulptur [Hen 02]).

Im Rahmen des Miniprojekts (siehe Abschnitt 3.6) lernten die Studierenden verschiedene Elemente einer Teaminfrastruktur kennen (z.Bsp. Kommunikation über Mailinglisten, Jour Fix) und entschieden sich gemeinsam, welche sie einsetzen wollen; sie verschafften sich einen Übersicht über ihre Stundenpläne, um herauszufinden, wann sie in der Woche Zeit haben, um gemeinsam zu arbeiten.

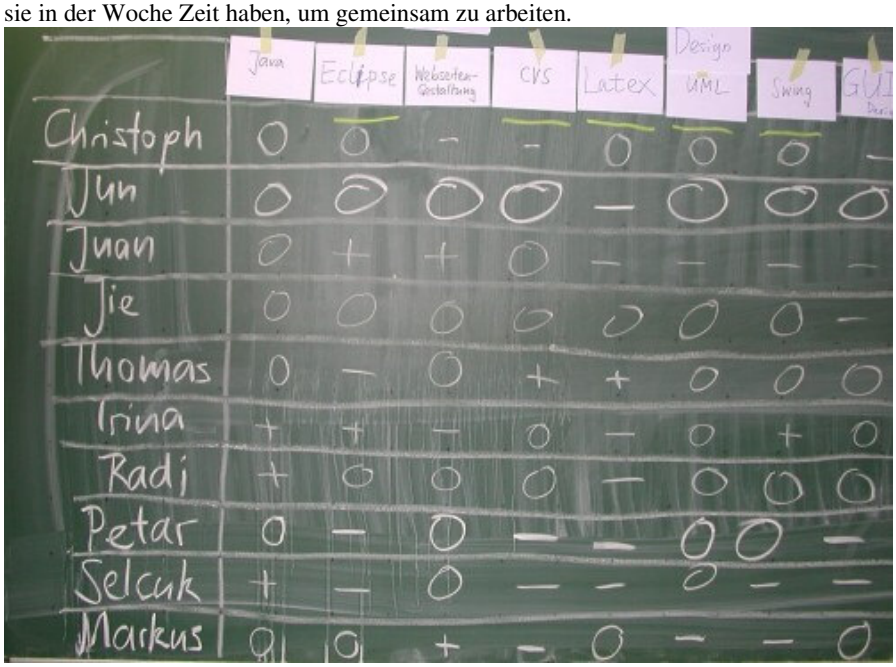

Abb. 5.: Zwischenergebnis auf dem Weg zum Teamprofil

Das Team diskutierte auch, welche Rollen man innerhalb des Teams vergeben will (Projektleiter, Verantwortliche für verschiedene Tools, Chefentwickler); sie verschafften sie sich eine Übersicht über die fachlichen Fähigkeiten und Interessen der einzelnen Teammitglieder, u.a. um einschätzen zu können, wer für welche Rolle am besten geeignet wäre und einigten sich darauf, wer welche Rolle übernehmen würde.

3.4 Als Team zusammenarbeiten (Teamwork)

Im Vordergrund des Workshops stand die Vermittlung von Arbeitstechniken, die das gemeinsame Arbeiten unterstützen: Zum einen als ganzes Team effizient und kooperativ zusammenzuarbeiten, zum anderen eine effektive Arbeitsteilung.

Dazu wurden den Studierenden verschiedene Techniken kurz vorgestellt, die sie dann in einer Übung selbst erprobten; anschließend wurden die Vor- und Nachteile der Technik reflektiert. Im Miniprojekt haben die Studierenden dann selbstständig Techniken ihrer Wahl eingesetzt, und bekamen zu Wahl und Durchführung einer Technik Feedback von den Trainern.

Sitzungen strukturieren

Hier wurden folgende Grundlagen vermittelt:

- Für die Struktur und Durchführung eines Meetings sollte eine Person verantwortlich sein; diese Person bereitet das Meeting vor (Raum reservieren, Erinnerungsmail verschicken, Tagesordnungspunkte sammeln) und moderiert das Meeting.
- Zu Beginn des Meetings sollten gemeinsam die Tagesordnungspunkte gesammelt werden, die besprochen werden müssen; diese werden dann entsprechend ihrer Wichtigkeit und Dringlichkeit in eine Reihenfolge gebracht und der Reihe nach abgearbeitet.
- Verschiedene Arten der Ergebnissicherung; zum Beispiel Sofort-Protokoll am Laptop, das mit einem Beamer projiziert wird.

Arbeitstechniken

Hier wurden verschiedene Arbeitstechniken zum gemeinsamen Sammeln, Ordnen/Strukturieren und Bewerten/Entscheiden vorgestellt:

- Brainstorming und Kartentechniken (wie zum Beispiel in Abbildung 7 dargestellt).
- Regeln für das effektive und kooperative Diskutieren, zum Beispiel Herstellen einer gemeinsamen Ausgangsbasis, Zieldefinition und Strukturierung von Diskussionen, Ergebnissicherung. Abbildung 7 zeigt Tipps zum Diskutieren, die die Studierenden selbst formuliert haben, betreffend das Einzelverhalten („ich“), das Gruppenverhalten („Gruppe“) und die gesamte Organisation der Diskussion.

Abb. 7.: Von Studierenden erarbeitete Tipps für Diskussionen

- Verschiedene Methoden, um gemeinsam Entscheidungen zu treffen, zum Beispiel Abstimmungen, Entscheidungsmatrizen oder Konsensdiskussion.

Uns war in diesem Zusammenhang besonders wichtig, nur solche Medien einzusetzen, die die Studierenden auch während des Praktikums einsetzen können; daher haben wir zum Beispiel auf Metaplanwände und Moderationskarten größtenteils verzichtet und stattdessen Wände, Beamer, Laptop und Flipcharts eingesetzt.

Arbeitsteilung

Etliche Aufgaben müssen nicht von allen Teammitgliedern gemeinsam bearbeitet werden, sondern können auf Teilgruppen verteilt werden. Dies erhöht die Effizienz des Teams erheblich - wenn einige Regeln beachtet werden, beispielsweise:

- Für jede Aufgabe sollte es eine verantwortliche Person und eine Deadline geben.
- Zu Beginn der Aufteilung sollte sichergestellt werden, dass die einzelnen Teilgruppen genau wissen, was sie bearbeiten sollen und welche Form die Ergebnisse haben sollen (z.Bsp. Stichpunktliste, kurze Präsentation).
- Wird ein Dokument aufgeteilt, bietet es sich an, dass vorher eine Teilgruppe eine Vorlage erstellt (aus der die Formatierungen, Formulierungsstil, Umfang ersichtlich sind); diese Vorlage sollte gemeinsam von allen Teammitgliedern diskutiert und verabschiedet werden.
- Wird Sourcecode aufgeteilt, gibt es Richtlinien für das gemeinsame Nutzen eines CVS Repositories (zum Beispiel: Regelmäßiges Einchecken; Update bevor man eine Datei bearbeitet; nur Einchecken von compilierbaren Code).

3.5 Projektmanagement

In der Regel haben nicht alle Teilnehmer an dem Praktikum bereits eine Software-Engineering Vorlesung gehört und daher keine rechte Vorstellung, wie man systematisch an ein größeres Programmierprojekt herangeht. Aus diesem Grund haben wir den Lernblock „Projektmanagement für Software-Entwicklung“ dem Teamtraining hinzugefügt.

Als Einstieg diskutierten wir mit den Studierenden anhand der beiden exemplarischen Vorgehensmodelle des Rational Unified Process (RUP) und des Extreme Programmings (XP) die grundlegend gemeinsamen Elemente von solchen Vorgehensmodellen.

Im Anschluss an die Diskussion bekamen die Studierenden von uns ein begründetes verbindliches Vorgehensmodell vorgegeben, das angesichts des kurzen Zeitrahmens des Projekts relativ simpel ist: Es handelt sich im Wesentlichen um ein aufgebohrtes Wasserfall-Modell, das aus drei Phasen besteht: Analyse, Design und Implementierung.

- In der Analyse wird hauptsächlich das Benutzerhandbuch erstellt. Gleichzeitig sollen die Studierenden aber auch schon einen Oberflächenprotoyp und ein logisches Datenmodell entwerfen.
- Im Design wird schwerpunktmäßig die Systemarchitektur entworfen. Gleichzeitig sollen die Studierenden aber anhand eines funktionalen Prototypens wichtige fachliche und technische Aspekte ausprobieren.
- In der Implementierung werden das ausführbare Programm und die Dokumentation erstellt.

Zusätzlich zu diesem Vorgehensmodell, das ein Software-Projekt in einzelne definierte Phasen segmentiert, schlugen wir ein allgemeines Vorgehen zum Problemlösen auf „Mikro-Ebene“ vor:

- Zunächst Problemanalyse und Zieldefinition (Ergebnisorientierung: Es wird festgelegt, was „hinten rauskommen“ soll).
- Als zweites wird die Meilensteinplanung (Zeitorientierung: Es wird eine Deadline für die Aufgabe bestimmt und die Zeit bis zur Deadline in Einzelschritte aufgeteilt; die Einzelschritte sind definiert durch die Teilergebnisse bzw. den Reifegrad eines Ergebnisses, das am Ende des Schrittes vorliegen soll).
- Danach je nach Situation (zum Beispiel wenn es um Strategien, gemeinsame Ideenfindung, Diskussionen oder Beschlüsse geht) gemeinsames Arbeiten im Team:

  „Weite Phasen“: Sammeln von Ideen, zum Beispiel in einem Brainstorming.

  „Enge Phasen“: Bewerten von Ideen, Aussortieren und Entscheiden.
- Oder paralleles Arbeiten in Kleingruppen (Arbeitsteilung).

Die vermittelten Arbeitstechniken und Problemlöseschritte haben die Studierenden im anschließenden Miniprojekt selbstständig ausgewählt und angewendet.

3.6 Miniprojekt

Etwa die Hälfte des Teamtrainings bestand aus einem „Miniprojekt“: In dieser Phase haben die Studierenden selbstständig gearbeitet, wurden dabei von den Trainern beobachtet und etwa alle 30 Minuten unterbrochen, um Feedback und Tipps zu geben. Ziele des Miniprojektes waren:

- Die Studierenden sollten die gelernten Techniken (siehe Abschnitt 3.4, Teamwork) selbstständig einsetzen und dadurch ihr Verständnis vertiefen; die Trainer sehen, wie die Studierenden die gelernten Techniken umsetzen, und haben die Gelegenheit, durch situationsbezogene Tipps zu korrigieren und zu ergänzen.
- Das Miniprojekt sollte den Übergang markieren vom “geschützten Raum” des Teamtrainings zum Praktikum. Es gab erste „echte“ Aufgaben, die von Rele-

vanz für das Praktikum waren, z.Bsp. Rollenverteilung (Entscheidung des Teams, wen sie zum Projektleiter bestimmen).

- Die Studierenden sollten sich durch die gemeinsame selbstständige Arbeit an den ersten „echten“ Aufgaben des Praktikums besser kennen und einschätzen lernen (siehe Abschnitt 3.3, Teambuilding).

Die Aufgabenstellung für das Miniprojekt lautete (in geraffter Form): *„Werdet euch klar über Stärken/Fähigkeiten/Wissen der einzelnen Teammitglieder! Bestimmt, wer von euch Projektleiter und wer Webmaster wird; überlegt euch, ob ihr noch weitere feste Rollen im Projekt festlegen wollt, und wer sie einnehmen soll!*

Ihr werdet mit Tools wie Java, Eclipse, CVS, ArgoUML/UML-Sequenz-Diagrammen, Latex und AutoFocus arbeiten müssen. Überlegt euch, wer sich bereits damit auskennt und wie ihr am besten erreichen könnt, dass sich alle damit auskennen!

Verschafft euch einen Überblick, wer wann in der Woche Zeit hat und was geeignete Termine für regelmäßige Treffen sind!

Ihr werdet beim Kick-Off ein etwa 50seitiges Pflichtenheft mit der Aufgabenbeschreibung überreicht bekommen; ihr werdet einige Zeit benötigen, um es zu lesen, und dann etliche Fragen haben, die ihr mit den Kunden klären müsst. Erstellt einen ersten vorläufigen Zeitplan für die Woche nach dem Kick-Off!”

Um diese Aufgaben zu bearbeiten, mussten die Studierenden selbst einen Zeitplan definieren; klären, wie sie die Aufgaben lösen wollen; Ergebnissicherung betreiben usw. Während die Studierenden selbstständig arbeiteten, wurden sie von den Trainern beobachtet und etwa alle 30 Minuten unterbrochen, um gemeinsam über den Arbeitsstil und die erreichten Ergebnisse zu reflektieren und Tipps zu geben.

Abb. 6.: Schnappschuss von Studierenden während des Mini-Projektes

3.7 Ablauf des Teamtrainings

Das Teamtraining fand als dreitägiges Blockseminar am ersten Semesterwochenende statt; Beginn war jeweils um 9 Uhr, Ende jeweils um 17 Uhr.

Der erste Tag begann mit einem ersten Kennenlernen der Teilnehmer; es wurde ein wenig Theorie (die „Teamkurve“) vermittelt. Thematischer Schwerpunkt war die Strukturierung von Meetings, Arbeitstechniken für Teams und Umgang mit Kritik. Am Vormittag des zweiten Tags standen effektives und kooperatives Diskutieren, sowie Projektmanagement im Vordergrund. Am Nachmittag begann das Miniprojekt. Am dritten Tag wurde das Miniprojekt fortgeführt und beendet.

Die Teilnehmer bekamen Handouts mit Checklisten zu den wichtigsten Themen, sowie ein Fotoprotokoll des Teamtrainings, in dem alle Tafelanschriften etc. abfotografiert waren.

4 Beobachtungen und Perspektiven

Das Teamtraining wurde von den Studierenden sehr positiv aufgenommen – sowohl in der Evaluation direkt nach dem Training im April als auch rückblickend in der Evaluation am Ende des gesamten Projektes im August.

Zu Beginn des Projektes konnten wir beobachten, dass das studentische Team selbstbewusst und mit optimistischer Stimmung ins Kick-Off ging. Während in den Jahren zuvor ohne Teamtraining die Studierenden anfangs schüchtern und desorganisiert waren, war das diesjährige Team sofort in der Lage (und fühlte sich auch entsprechend sicher vorbereitet) mit der fachlichen Arbeit mit Beginn des STP zu beginnen. Die Folge war, dass dadurch mindestens ein bis zwei Wochen „einspart“ wurden, die dem Team somit zusätzlich für die Bewältigung der fachlichen Aufgabe zur Verfügung standen - insofern war das Teambuilding erfolgreich.

In Bezug auf die Umsetzung der Arbeitstechniken zeigte die Beobachtung des Teams während des Projektes, dass viele gelernte Techniken nicht voll eingesetzt wurden. Dafür sehen wir vor allem drei Gründe:

Struktur

Die Studierenden hatten bei vielen Meetings das Empfinden, die gelernte Systematik (Tagesordnung, Sitzungsleitung, Protokoll) nicht einsetzen zu müssen, da es sich ja „nur“ um ein „normales“ bzw. „einfaches“ Meeting handele. Unsere Beobachtung ergab allerdings, dass das „Einsparen“ einiger Minuten Sitzungsorganisation am Beginn des Meetings die Besprechungen insgesamt deutlich ineffizienter gemacht hat.

Hier wollen wir beim nächsten Teamtraining noch stärker die Vorteile einer strukturierten Teamsitzung deutlich machen und versuchen, bereits im Teamtraining eine Routine zu entwickeln, in der Hoffnung, dass diese dann von den Studierenden quasi automatisch in die eigentliche Projektarbeit übernommen wird.

Führung

Der studentische Projektleiter hatte Schwierigkeiten, die mit seiner Rolle verbundene Führungsaufgabe wahrzunehmen, da er dies als unzumutbare Bevormundung seiner Mitstudierenden empfand. Sein „laisez faire“ Stil konnte allerdings das Bedürfnis der Studierenden nach Führung und Fokussierung nicht ausreichend befriedigen.

Hier wollen wir beim nächsten Teamtraining in der gesamten Gruppe transparenter machen, was genau die Führungsaufgaben eines (studentischen) Projektleiters sind und dass sich Basisdemokratie und Führung nicht notwendigerweise widersprechen. Zudem überlegen wir, beim nächsten Praktikum der Gruppe die Wahl zu lassen, ob sie eine Projektleitung möchten.

Reflektion

Der Zeitdruck im Projekt war von Anfang an sehr hoch, so dass sich das Team sehr schnell kaum mehr des Lernumfelds bewusst war; vielmehr stand nur noch der erfolgreiche Abschluss des Projekts im Vordergrund. Zeit für Reflektion über Vorgehen und Techniken des Software-Engineering blieb kaum.

Zurzeit überlegen wir daher, ob wir im nächsten Projekt versuchen sollen, solche Reflektionsphasen zu erzwingen, auch wenn wir nicht sicher sind, wie sich das auf den „Fluss“ des Projektes auswirken wird. Alternativ ist angedacht, jeden Studierenden am Ende des Projekts einen individuellen Projektbericht schreiben zu lassen, in dem er informell den Verlauf des Projektes im Rückblick beschreibt. Eine weitere Idee wäre, Absolventen eines solchen Projektes im darauf folgenden Jahr einen Hiwi-Job als Betreuer eines nächsten Projektes anzubieten. In der Distanz als Betreuer und im nochmaligen Erleben eines Projektes liegt ein großes Lernpotential.

5 Fazit

Zusammenfassend stellen wir fest, dass, auch wenn noch viele Detailverbesserungen denkbar sind, ein Teamtraining ein Software-Engineering Projekt in der Lehre signifikant unterstützen kann: Die Studenten können durch das Teambuilding viel schneller in die eigentliche fachliche Projektarbeit einsteigen, sie arbeiten im Team deutlich effizienter und strukturierter, sie können souveräner mit sozialen und organisatorischen Herausforderungen umgehen.

Ein Teamtraining unterstützt folglich den Lernerfolg eines solchen Projektes zum einen dadurch, dass während des Projektes weniger die sozialen Probleme dominieren, sondern stattdessen mehr Raum für das Lernen und Ausprobieren „klassischer“ fachlicher Software-Engineering Inhalte ist.

Zum anderen wird der Lernerfolg durch eine explizite Behandlung von sozialen und organisatorischen Themen unterstützt, die im Praxisalltag zwar wichtig sind, aber als „unbequem“ empfunden und gerne verdrängt werden. Allerdings plädieren wir dafür, neben den fachlichen und technischen Inhalten verstärkt auch Softskills wie Teamwork

und Projektmanagement als *gleichberechtigt wichtige* Lernziele innerhalb des Software-Engineering wahrzunehmen - oder, wie Prof. Ludewig etwas plakativer zum Thema Softskills sagte: „Es gibt keinen Gott der Informatik, der das Gebot verkündet hätte: Du sollst nur Beweisbares lehren!“ [Lud 02].

Wir empfehlen allen Lehrstühlen, die solche Praktika oder Projekte durchführen, im Rahmen der Vermittlung für Software-Ingenieure wichtiger Softskills auch solch ein vorbereitendes Teamtraining in Betracht zu ziehen. Wie das Beispiel der TU Darmstadt zeigt, können bei der Konzeption eines solchen Trainings Kooperationspartner von Psychologie-Lehrstühlen oder Hochschuldidaktischen Stellen der Hochschule unterstützend hinzugezogen werden; damit kann auch der Aufwand bei der Schulung der Trainer und bei der Konzeption eines Teamtrainings eingeschränkt werden. Und auch wir selbst stehen gerne bei der Initiierung von Teamtrainings beratend zur Seite.

Literatur

[Aut 02] Homepage des Design- und Simulationswerkzeuges AutoFocus. München, 2004. http://autofocus.in.tum.de

[Aut 04] Homepage des AutoRAID Praktikums. München, 2004. http://wwwbroy.in.tum.de/~autoraid

[Den 98] M. Deneke, U. Schröder, M. Brunner: Constructionist Learning in Software Engineering Projects. SEES 1998, Software Engineering Education Symposium 18 - 20 November 1998. Poznan, Polen, 1998.

[Fle 02] A. Fleischmann: Entwurf eines berufsbegleitenden Aufbaustudiengangs für Software-Engineering. Diplomarbeit am Fachbereich Informatik der Technischen Universität Darmstadt; Betreuer: Prof. Henhapl. Darmstadt, 2002.

[Gna 03] M. Gnatz, L. Kof, F. Prilmeier, T. Seifert: A Practical Approach of Teaching Software Engineering. In P. Knoke, A. Moreno, M. Ryan (Hrsg.): 16th Conference on Software Engineering Education and Training, pages 120-128. IEEE Computer Society, March 2003.

[Hen 02] W. Henhapl (Hrsg.): Informationsseite des Instituts für Praktische Informatik an der TU Darmstadt zum Software-Engineering Projekt. Darmstadt, 2002. http://www.pi.informatik.tu-darmstadt.de/se2002/

[Kom 04] Hochschuldidaktische Arbeitsstelle der Technischen Universität München: Teamtrainings nach dem KOMPASS Konzept. Darmstadt, 2004. http://www.tu-darmstadt.de/hda/tutorengruppe/kompass.html

[Sen 04] T. Senninger (Hrsg.): Kooperationsspiele, eine Sammlung. 2004. http://www.abenteuerprojekt.de/Spiele/kooperation.php

[Lan 00] B. Langemaack, M. Braune-Krickau: Wie die Gruppe laufen lernt. Anregungen zum Planen und Leiten von Gruppen. Beltz Verlag, Weinheim 2000.

[Lud 02] J. Ludewig: Thesen zur Softwaretechnik in den Universität. Vortrag am 30. April 2002 in der TU Darmstadt auf dem Symposium „Softwaretechnik Aus- und Weiterbildung“. Darmstadt, 2002. http://www.informatik.uni-stuttgart.de/ifi/se/publications/download/VortragDA.pdf

[Hen 02] M. Henninger, H. Mandl: Zuhören, verstehen, miteinander reden. Hans Huber Verlag, 2003.

[Tau 01] D. Taubner: Software-Entwicklung im industriellen Maßstab. In: J. Desel (Hrsg.:) "Das ist Informatik", Springer Verlag 2001. Seite 85-98, http://www.sdm.de/dt/tec/pub/art/art/seiim/seiim.pdf