Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

Eine Plattform für die Softwaretechnik-Fernlehre

Philipp Bouillon, Jens Krinke, Stephan Lukosch
FernUniversität in Hagen, 58084 Hagen
{philipp.bouillon,jens.krinke,stephan.lukosch}@fernuni-hagen.de

Zusammenfassung

*Softwaretechnik-Lehre ist ohne entsprechende Praktika und Projekte nicht sinnvoll. An der FernUniversität in Hagen ist die Softwaretechnik-Lehre auch immer **entfernte** Lehre und die Anforderungen an die Softwaretechnik-Praktika sind somit höher. Wir beschreiben eine integrierte Entwicklungsumgebung, die auf Eclipse basiert und alle Werkzeuge, die für die entfernte Softwaretechnik-Lehre nötig sind, unter einer einzigen, in sich geschlossenen Oberfläche vereint.*

1 Einführung

Die Lehre der Softwaretechnik an Universitäten ist kompliziert: Die Studierenden müssen nicht nur mindestens eine Programmiersprache lernen und die Softwaretechnik sowohl theoretisch als auch praktisch verinnerlichen, sondern sie müssen zusätzlich auch noch lernen, sich im Team zurechtzufinden. Sie müssen ihre Kommunikationsfähigkeiten verbessern und ein Verständnis für die Probleme bekommen, die sich während der Entwicklung eines großen, realistischen Projekts ergeben. Um das Verständnis für die Probleme der Softwareentwicklung zu schärfen, muss vom Dozenten eine große Projektaufgabe vorgegeben werden, die sich von den „Spielprogrammen“ aus Anfängervorlesungen allein schon durch ihre schiere Komplexität abhebt. Um die Aufgabe als ganze bewältigen zu können, werden die Studierenden in Teams eingeteilt, und es werden ihnen verschiedene Werkzeuge an die Hand gegeben, die es ihnen ermöglichen, Teile des Gesamtsystems zu meistern.

An der FernUniversität in Hagen werden sämtliche Softwaretechnik-Kurse als Fernkurse angeboten (sei es über das Internet oder auf dem Postweg); bei den Praktika in der Softwaretechnik kommt es aufgrund der Entfernung zu und zwischen den Studierenden aber zu besonderen Problemen, die gelöst werden müssen. Zurzeit müssen die Studierenden, die an einem Praktikum teilnehmen möchten, die FernUniversität besuchen und sich dann vor Ort zu einer Gruppe zusammenfinden. Die ersten Phasen des Praktikums (grobes Pflichtenheft und Grobentwurf) werden vor Ort in Hagen durchgeführt. Danach reisen die Studierenden zu ihren jeweiligen Heimatorten zurück, von

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

wo aus sie dann online über ein System namens CURE (Collaborative Universal Remote Education Environment) [Haa04a,Haa04b] weiter an dem Design und der Implementierung des Projekts arbeiten. CURE ist die kooperative Lernplattform der FernUniversität.

Um die Zusammenarbeit einer Gruppe während eines Praktikums zu ermöglichen, muss diese in den Bereichen Kommunikation, Koordination und Kollaboration unterstützt werden. Diese Bereiche werden häufig zur Klassifizierung von Groupware herangezogen [Teu95]. Bei der Kommunikation steht der Informationsaustausch zwischen den Gruppenmitgliedern im Vordergrund. Die Koordination dient der Abstimmung bei gemeinsamen Aufgaben. Die Kooperation schließlich fordert zusätzlich die Verfolgung gemeinsamer Ziele. Aus der oben beschriebenen Situation heraus lassen sich in den Praktika an der FernUniversität in all diesen Bereichen Probleme ausmachen:

Kommunikation. Studierende der FernUniversität benutzen in der Regel elektronische Hilfsmittel, um miteinander zu kommunizieren, also z.B. E-Mail, Instant Messaging, Chat oder auch das Telefon. Da sie aber oftmals unterschiedliche Arbeitszeiten (und Arbeitsweisen) haben, ist der überwiegende Anteil der Kommunikation *asynchron*. Eine direkte Rückmeldung auf eine evtl. dringende Frage ist somit meistens nicht möglich.

Koordination. Aus der räumlichen Distanz der Studierenden ergibt sich automatisch ein hoher Koordinationsbedarf, um die einzelnen Tätigkeiten untereinander abzustimmen und eine Zusammenarbeit zu ermöglichen. Abstimmungsprozesse werden oft durch Kommunikation oder durch gemeinsame Artefakte (z.B. gemeinsamer Projektplan) unterstützt.

Kollaboration. Da die Studierenden sich nicht in einem Computerraum treffen können, um miteinander zu arbeiten, müssen sie stattdessen wieder auf elektronische Hilfsmittel zurückgreifen. Meist werden die Studierenden dann auch wieder zu unterschiedlichen Zeiten an dem Projekt arbeiten, weshalb es wichtig ist, dass jeder Einzelne weiß, was sich seit dem letzten Zugriff am Projekt verändert hat. Außerdem muss klar ersichtlich sein, was als Nächstes zu erledigen ist.

Wir werden im Folgenden *Kooperation* als Oberbegriff für diese drei Begriffe benutzen.

Um die genannten Probleme in den Griff zu bekommen, haben wir eine einheitliche Entwicklungsplattform auf der Grundlage von Eclipse entwickelt. Diese beinhaltet sämtliche Werkzeuge, die für die entfernte Entwicklung von Softwareprojekten nötig sind. Dazu wurden bestehende Werkzeuge in die Eclipse-Plattform integriert und einige weitere Programmteile hinzugefügt, die dafür sorgen, dass das Konglomerat verschiedenster Werkzeuge reibungslos zusammenarbeiten kann. Von unserer Plattform können auch Präsenzuniversitäten profitieren, da die meisten der hier genannten Probleme der Softwaretechnik-Lehre auch dort auftreten.

Der Rest dieser Diskussion ist wie folgt aufgebaut: Abschnitt 2 stellt einen Überblick über die aktuelle Situation in der Softwaretechnik-Lehre dar. Es wird dabei beson-

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

ders auf die Probleme eingegangen, die wir mit unserer Plattform lösen. In Abschnitt 3 werden die Komponenten unserer Plattform genauestens beschrieben. Abschnitt 4 gibt einen Ausblick über die Dinge, die noch zu erledigen sind, während der Artikel schließlich mit Abschnitt 5 zusammengefasst und abgeschlossen wird.

2 Softwaretechnik-Lehre heute

Studierende der Informatik erlernen heutzutage bereits im Grundstudium das Schreiben eines Programms in mindestens einer Programmiersprache. Normalerweise wird in diesen Vorlesungen Wert darauf gelegt, dass die Studierenden das Konzept der Programmiersprache verstehen und somit selbstständig in der Lage sind, große Programme zu entwickeln. Um das Verständnis der *Softwareentwicklung im Großen* noch zu erhöhen, gibt es theoretische Softwaretechnik-Vorlesungen (die häufig erst im Hauptstudium stattfinden), in denen die Studierenden erstmals direkt mit der Komplexität eines großen Softwaresystems in Berührung kommen.

Die rein theoretischen Betrachtungen der Softwaretechnik reichen allerdings nicht aus, um den Studierenden ein Gefühl von den Kommunikations-, Koordinations- und Kollaborationsproblemen zu geben, die während eines Softwareprojekts auftreten können. Studierende neigen dazu, sich auf die technischen Probleme zu konzentrieren und zu glauben, dass die sozialen Aspekte erst gar nicht auftreten. Daher werden an vielen Universitäten und insbesondere auch an der FernUniversität in Hagen begleitend zu den Softwaretechnik-Vorlesungen Praktika angeboten, in denen eine Gruppe von Studierenden ein größeres Softwaresystem entwickeln muss. Um solche Praktika zu erleichtern, wird eine Reihe von Softwarewerkzeugen verwendet. Bezogen auf Eclipse gibt es bislang noch nicht viele Plug-ins, die Kollaborationsprozesse unterstützen. Es gibt zwar einige Plug-ins, die Instant-Messaging-Protokolle in Eclipse integrieren, wie z.B. das IM-Plug-in[1] oder PepeMax[2], aber bei Kollaboration geht es um weit mehr als nur um Instant Messaging. Andere Plug-ins wie Jazz [Che03] versuchen, Eclipse um Kollaborationsaspekte zu erweitern, aber Jazz konzentriert sich dabei auf synchrone Kommunikationsprozesse und auf Teams, die eng beieinander arbeiten. Solche Plug-ins sind für die Situation an der FernUniversität also nicht geeignet.

Andererseits unterstützt CURE [Haa04a,Haa04b] als kooperative Lernplattform der FernUniversität bereits die synchrone und asynchrone Kooperation. CURE unterstützt gemeinsames Lernen in verteilten Teams mittels eines Standard-Web-Browsers und des Internets. CURE basiert auf der Kombination der Raum-Metapher, die häufig zur Strukturierung der Kollaboration eingesetzt wird [Gre02, Pfi98], mit Ideen aus Web-basierten Arbeitsbereichen (Wiki) [Leu01] und Kommunikations- und Koordinationswerkzeugen. Zu den Werkzeugen zählen ein E-Mail-System, Dokumentenverwaltung und ein Kalender mit Terminfindungsfunktionen. Studierende

[1] *http://eimp.sourceforge.net/d/*
[2] *http://pepemax.jabberstudio.org/*

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

der FernUniversität sind bereits an dieses System gewöhnt, so dass sich eine Einbindung in unsere neue Plattform für Softwaretechnik-Projekte anbietet.

Jedes Softwaretechnik-Projekt, das im Rahmen der Fernlehre durchgeführt wird, muss als *verteilte Softwareentwicklung* (engl. DSE: Distributed Software Engineering) betrachtet werden. DSE wird bereits von Projektmanagement oder Groupware-Lösungen unterstützt, dabei reicht die angebotene Bandbreite von Kommunikations-, Koordinations- und Kollaborationsunterstützung bis hin zu vollständigen Softwaresystemen wie SourceForge[3] oder GForge[4], die bei Open-Source-Projekten zum Einsatz kommen. In der Tat stellen die beiden letztgenannten Systeme eine gute Infrastruktur zur Erstellung von großen Softwaresystemen zur Verfügung, weshalb einige Universitäten GForge benutzen. Wir haben uns jedoch dazu entschlossen, eine kommerzielle Variante zu verwenden: Intlands CodeBeamer[5] ist leichter zu bedienen und bietet gleichzeitig mehr Funktionalität.

Eine weitere große Anforderung an die neue Plattform ist die Möglichkeit für die Betreuer des Projekts, einzelne Studierende bewerten und benoten zu können. Außerdem müssen die Betreuer die Gruppe bei Problemen beraten können und ihnen Ratschläge erteilen, bevor es dafür zu spät ist. Zu diesem Zweck ist es wichtig, dass ein Betreuer die Kommunikation der Gruppe nachvollziehen kann: Sowohl die informellen projektbezogenen Gespräche als auch die eingeforderten Dokumente, die während des Projekts entstehen, müssen analysiert werden können. Außerdem muss natürlich auch das Endprodukt, also die entstandene Software jeder Gruppe, ausprobiert und bewertet werden können. Einige Werkzeuge zur Analyse von Software existieren bereits, die vom Betreuer benutzt werden können, um eine solche Analyse durchzuführen. Das JRefleX-Projekt bietet sogar Eclipse Plug-ins an, die zur Analyse der Kollaboration im Team und zur Evolutionsanalyse des Programms herangezogen werden können.

Um eine Plattform zu entwickeln, die die Kooperation der einzelnen Teammitglieder verbessert, mussten neue Plug-ins entwickelt werden und bestehende so erweitert werden, dass sie besser mit anderen zusammenarbeiten können. Welche Plug-ins das im Einzelnen sind, wird im nächsten Abschnitt beschrieben.

3 Softwaretechnik-Lehre morgen

Zu Beginn der Entwicklung einer neuen Plattform stellt sich die Frage, welche Werkzeuge in die IDE integriert werden müssen, um das Teamwork unter den Studierenden zu verbessern, obwohl sie so weit voneinander entfernt sind. Die Frage ist recht einfach, die Antwort hingegen nicht. Studierende (und auch professionelle Softwareentwickler) haben alle ihre unterschiedlichen Arbeitsweisen und Methoden, um ein Softwareprojekt voranzutreiben: Einige arbeiten lieber nachts, andere sind echte Frühaufsteher und erledigen einen Teil der Arbeit vor dem Frühstück. Manche Programmierer tendieren dazu,

[3] *http://www.sourceforge.org/*
[4] *http://www.gforge.org/*
[5] *http://www.intland.com/*

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

alles zunächst genau zu planen und dann zu implementieren, andere hingegen programmieren zuerst und testen danach. Wie also kann eine IDE dabei helfen, die jeweils bevorzugte Arbeitsweise eines Entwicklers zu unterstützen und ihn nicht dazu zu zwingen, eine vorgeschriebene Methode wählen zu müssen? Die wichtigen Punkte, auf die es dabei ankommt, sind *Kommunikation*, *Koordination* und *Kollaboration*. Wie oben schon angedeutet wurde, gibt es an der FernUniversität zwei wichtige Systeme für Softwareprojekte: CURE und CodeBeamer. Beide unterstützen Kooperation.

3.1 Kooperation mit CURE

CURE [Haa04a, Haa04b] ist eine kooperative Lernplattform zur Unterstützung kooperativer Lernsituationen über das Web. Benutzer können unabhängig von ihrem Standort gemeinsam auf Informationen zugreifen. Verteilte Teams können in CURE ihr gemeinsames Lernen selbst organisieren, indem sie in gemeinsamen Arbeitsbereichen Lernmaterial kooperativ bearbeiten und synchron bzw. asynchron miteinander kommunizieren. Eine Gruppe ist dabei definiert als die Menge der Nutzer eines gemeinsamen Arbeitsbereichs. Gemeinsame Arbeitsbereiche und deren Zugangsberechtigungen werden in CURE mit Hilfe der Raum-Metapher modelliert. Der Zugriff auf einen Arbeitsbereich wird durch die Schlüssel-Metapher gesteuert. Um Mitglied einer Gruppe auf einem Arbeitsbereich (d.h. Raum) zu werden, muss man einen passenden Schlüssel für den Raum besitzen. Durch die Kombination von Räumen und Schlüsseln, die an Schlüssel geknüpften Rechte und die auf Schlüssel definierten Operationen werden verschiedenste Formen der Gruppenbildung unterstützt.

So kann jeder Benutzer Räume für bestimmte Gruppen und Zwecke anlegen, und die Besitzer eines Raums können die Zugriffsrechte einschränken. Ein Raum besteht aus mindestens einer Seite, die von jedem Benutzer mit den entsprechenden Rechten mit Hilfe einer einfachen Wiki-Syntax editiert werden kann. Es ist auch möglich, TeX-Kommandos in diese Seiten einzubinden, um beispielsweise komplexe mathematische Formeln zu erzeugen. Neben normalen Seiten kann ein Raum auch Datenseiten enthalten, auf denen ein Benutzer beliebige Dateien ablegen kann. Über eine Anbindung von CURE an NetMeeting[6] oder DyCE [Tie01], ein komponentenbasiertes Groupware-Framework, ist auch eine synchrone Kooperation unter den Benutzern möglich. Außerdem kann ein Raum einen eigenen Chat und eine eigene Mailbox besitzen, deren Daten jeweils persistent gespeichert werden: Keine der Informationen, die an einen Raum gesendet werden, gehen verloren. Sowohl die Diskussionsbeiträge in den Mail-Foren des Raums als auch die Einträge im Chat bleiben erhalten.

Bisherige Erfahrungen mit der CURE-Umgebung haben gezeigt, dass die Studierenden überwiegend die asynchronen Kommunikationsmechanismen von CURE benutzen. Dies hat zwei Gründe: Erstens kommt synchrone Kommunikation an der FernUniversität selten vor (wie oben bereits erläutert wurde), und zweitens bevorzugen die Studierenden ihre eigenen, gewohnten Instant-Messaging-Programme. Die bekannten Instant-Messaging-Werkzeuge lassen sich noch zusammen mit der IDE der Studieren-

[6] *http://www.microsoft.com/windows/netmeeting/*

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

den anzeigen, wohingegen CURE in einem eigenen Browserfenster läuft. Um also zu den synchronen Kommunikationswerkzeugen von CURE zu gelangen, muss die IDE der Studierenden mit CURE überlagert werden: Es findet ein Kontextwechsel statt, der den Arbeitsfluss unterbricht. Dieser notwendige Wechsel führte dazu, dass die Studierenden aufgehört haben, CURE für ihre Chats zu benutzen, was wiederum dazu führte, dass die Betreuer des Praktikums keine Möglichkeit mehr hatten, auf diese Chats einzugehen. Um diesen Nachteil zu umgehen, haben wir ein CURE Plug-in für Eclipse entwickelt, das aus mehreren Sichten besteht. Ein Screenshot dieses Plug-ins ist in Abbildung 1 zu sehen. Die Hauptsicht zeigt die Seite eines Raums, während die Baumansicht auf der rechten Seite alle momentan zur Verfügung stehenden Räume anzeigt. Ein Softwareprojekt wird in einem eigenen Raum abgelegt, während die einzelnen Seiten Informationen über die Teilprojekte, Meilensteine und verschiedene andere Themen enthalten.

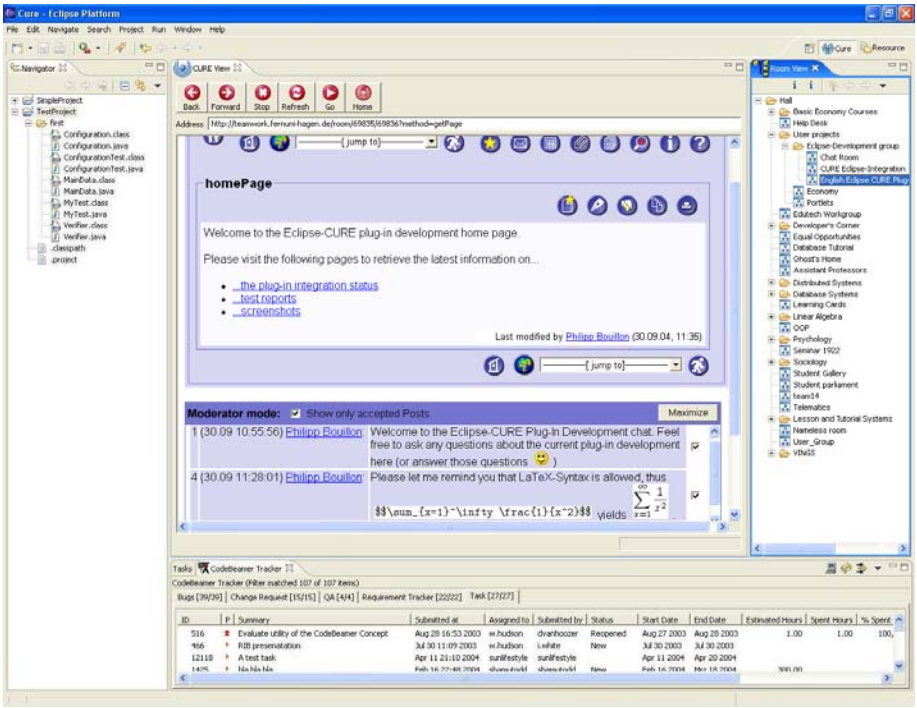

Abb. 1.: ECLIPSE mit CURE und CodeBeamer-Sicht

3.2 Projektmanagement

Softwareprojekte in der Fernlehre benötigen, genau wie andere Softwareprojekte auch, ein strenges Projektmanagement. Die Erfahrung an der FernUniversität zeigt, dass Teams, die CURE zum Projektmanagement eingesetzt haben, bessere Ergebnisse erzielt

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

haben als Teams, die keine besonderen Werkzeuge zur Verfügung hatten. Allerdings bietet CURE nicht die Funktionalität von existierenden Projektmanagement-Werkzeugen. Daher setzen wir zum Projektmanagement CodeBeamer ein. Dabei handelt es sich um eine serverbasierte Lösung, die eine Reihe von leicht verständlichen Kooperationshilfen anbietet. CodeBeamer unterstützt Projektmanagement durch verschiedene *Tracker*, die zum Verwalten von Anforderungen, Aufgaben, Bugs usw. genutzt werden können. Die einzelnen Trackerdaten können in Diagrammen (z.B. Gantt-Diagramme) visualisiert werden. Durch diese Tracker können sowohl die Studierenden als auch die Betreuer leicht sehen, wie das Projekt voranschreitet. Dieser Aspekt wird auch noch dadurch verstärkt, dass CodeBeamer anzeigt, welche Dinge zuletzt erledigt worden sind, bzw. welche Aufgaben als Nächstes zu bewältigen sind. Es gibt bereits ein Eclipse Plug-in von CodeBeamer, das die Tracker-Ansicht des Produkts in eine eigene Eclipse-Sicht integriert.[7] Der CodeBeamer-Dienst wird von einem Server an der FernUniversität angeboten und kann somit von jedem Teilnehmer des Softwaretechnik-Praktikums in Anspruch genommen werden. Neben dem Plug-in muss auf Studierendenseite keine weitere Software installiert werden. Abbildung 1 zeigt einige Tracker von CodeBeamer in der unteren Sicht.

3.3 Werkzeuge für die Phasen eines Projekts

Neben den zwei oben genannten „großen“ Komponenten unserer Plattform, haben wir einige bereits existierende Plug-ins für Eclipse auf ihre Verwendbarkeit in Softwareprojekten (der Fernlehre) hin untersucht. Für die einzelnen Phasen des Projekts werden unterschiedliche Werkzeuge angeboten, die wir im Folgenden vorstellen.

Phase 1: Erstellen des Projekts

Wenn ein Projekt erstellt wird, muss der Betreuer in der Lage sein, eine Projektbeschreibung zu erstellen und die Meilensteine anzugeben, zu denen bestimmte Teile des Projekts fertig gestellt sein müssen. Eine allgemeine Beschreibung des Projekts kann in einem Raum für das Projekt in CURE hinterlegt werden. Für eine detaillierte Planung können die Task-Tracker von CodeBeamer verwendet werden, wobei jeder Task einen Abschlusstermin haben kann. Es wäre wünschenswert, wenn diese Abschlusstermine automatisch in einen Teamkalender eingetragen werden könnten, der von jedem einzelnen Teammitglied zu jeder Zeit eingesehen werden kann. Außerdem könnte jeder im Team diesen Teamkalender noch um eigene teaminterne Termine erweitern. Leider haben wir kein Plug-in gefunden, das einen solchen Kalender implementieren würde. Die optimale Lösung bliebe an dieser Stelle wohl eine Integration von Eclipse mit MS Outlook oder IBM Lotus Notes, da dies die Anwendungen sind, mit denen die meisten Leute ihre Termine synchronisieren. Aktuelle Open-Source-Projektmanagement-Werkzeuge bieten ihre eigenen Kalender an, die jedoch nicht mit anderen Kalendern

[7] Ursprünglich hatten wir vor, GForge als eine kostenlose Alternative zu benutzen, aber GForge ist nicht leicht in Eclipse integrierbar.

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

synchronisiert werden können. Zurzeit gibt es sogar zwei verschiedene Kalender in der Universitätsplattform: einen in der Verwaltungs- und E-Learning-Plattform und einen in CURE; beide sind voneinander unabhängig und können nicht miteinander synchronisiert werden.

Phase 2: Anforderungsanalyse

Zu Beginn eines jeden Projekts beschäftigen sich die Studierenden fast ausschließlich damit, Unklarheiten zu diskutieren und Anforderungen niederzuschreiben. Um für eine lebhafte Diskussion zwischen den Mitgliedern eines Teams zu sorgen, muss sowohl synchrone als auch asynchrone Kommunikation möglich sein. Weiterhin muss es in der gleichen Umgebung möglich sein, das Anforderungsdokument zu erstellen, ohne ständig zwischen den Anwendungen zur Diskussion und zum Niederschreiben hin- und herwechseln zu müssen. Somit ist die CURE-Architektur mit den Wiki-Seiten, dem Chat und der Mail-Funktion an dieser Stelle ideal. Durch die Versionsverwaltung in CURE können auch immer ältere Versionen der einmal erstellten Dokumente abgerufen werden.

Das Team der Studierenden muss aus der Anforderungsanalyse heraus auch einzelne Tasks erstellen und diese in ihren Trackern eintragen. Die Studierenden müssen ihr Projekt also planen sowie für jede Anforderung und jede Aufgabe einen Tracker-Eintrag erstellen. Somit wird die Nachvollziehbarkeit für alle Teammitglieder und den Betreuer erhöht, da CodeBeamer alle zuletzt bearbeiteten Tasks anzeigen kann. Neben CodeBeamer gibt es auch noch andere Plug-ins, die sich z.B. zum Ziel gesetzt haben, Bugzilla in Eclipse zu integrieren.

Phase 3: Entwurf

Es gibt bereits eine Reihe von UML-Entwurfswerkzeugen, aber keines der verfügbaren Open-Source-Werkzeuge erlaubt das gemeinsame Modellieren über das Internet. In dieser Hinsicht wäre zumindest eine grafische Diff-Sicht sinnvoll, die dem Benutzer direkt anzeigt, welche Veränderungen seit dem letzten Zugriff vorgenommen wurden (zum Beispiel indem veränderte, hinzugefügte oder entfernte Elemente grafisch hervorgehoben werden). Des Weiteren sind die meisten UML-Werkzeuge nur schwer in ein Softwaretechnik-Praktikum zu integrieren, da sie entweder zu schwer zu bedienen sind oder nicht genug Funktionalität besitzen. Für unsere Zwecke am besten geeignet sind Together[8] und MagicDraw[9]. Andere Werkzeuge sind entweder (noch) nicht mit Eclipse integrierbar oder haben Stabilitätsprobleme.

Ein vielversprechendes Projekt namens GroupUML [Bou04] wird derzeit an der TU München entwickelt. Dabei handelt es sich um ein Werkzeug, mit dem mehrere Benutzer gleichzeitig halb-formale UML-Diagramme erstellen können. Zum einen ist es möglich, komplette UML-Diagramme zu zeichnen, auf der anderen Seite können aber auch

[8] *http://www.borland.com/together/eclipse/*
[9] *http://www.nomagic.com/*

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

Kommentare, Freihand-Zeichnungen und beliebige andere Figuren in die Dokumente integriert werden. Kurz gesagt handelt es sich also um eine Kombination aus Shared Whiteboard und UML-Diagrammeditor. Das Besondere an dem Werkzeug ist, dass alle Teammitglieder gleichzeitig sehen können, was gerade vor sich geht. Gleichzeitig können sie auch, wenn sie erst später dazukommen, nachvollziehen, was sich bereits getan hat, da Chat-Meldungen ebenfalls von dem Programm aufgezeichnet werden. Die Chat-Meldungen können gegebenenfalls sogar mit bestimmten Elementen des aktuellen Arbeitsbereichs verbunden werden. Wir werden dieses Projekt noch genauer evaluieren und dann entscheiden, ob sich eine Integration in unsere Plattform realisieren lässt.

Phase 4: Implementierung und Modultests

Eclipse ist schon hervorragend ausgestattet, um den Programmierer bei der Entwicklung von Java-Programmen zu unterstützen. Andere Sprachen werden zwar noch nicht so gut unterstützt, aber das wird sich in Zukunft ändern. Zum Testen wird JUnit schon mit Eclipse ausgeliefert. Unsere Studierenden müssen JUnit außerdem benutzen, um ihre Module zu spezifizieren. Unabhängig von der Programmiersprache gibt es bereits eine Team-Schnittstelle in Eclipse, die eine Anbindung an ein CVS-System erlaubt. An der FernUniversität gibt es einen CVS (oder SubVersion)-Server, auf den die Studierenden dann mit Hilfe der Team-Schnittstelle zugreifen können.

Zusätzlich wird an der FernUniversität CodeBeamer dazu eingesetzt, die Studierenden während der Implementierungsphase zu unterstützen. Mit Hilfe der Tracker, die in CodeBeamer zur Verfügung stehen, können sich die Studierenden einen Überblick über die aktuellen Aufgaben verschaffen und auch gegenseitig die Bugs aus den Programmteilen entfernen. Mit Hilfe der automatischen Testfunktion von CodeBeamer ist es möglich, das gesamte Projekt über Nacht testen zu lassen. Am nächsten Morgen werden dann die Tasks automatisch aktualisiert, so dass jedes Teammitglied einen messbaren Fortschritt des Projekts erkennen kann. Des Weiteren gibt es noch eine grafische Visualisierung der Testergebnisse, so dass sich jeder Studierende schnell einen Überblick über Probleme verschaffen kann.

Zur Verbesserung der Implementierungsphase sollen noch externe Werkzeuge wie JML (eine Spezifikationssprache für Java) zur Verfügung gestellt werden. Wir werden auch noch die Effekte von Code-Checking Plug-ins wie CheckStyle oder PMD sowie von Debug-Hilfen wie DeltaDebugging [Bou03] oder anderen Werkzeugen evaluieren. Gefundene Bugs werden natürlich im Bug Tracker von CodeBeamer verwaltet.

Phase 5: Abgabe und Benotung

Wenn die Studierenden am Ende ihres Praktikums ihre Software zusammen mit den geforderten Dokumenten abgegeben haben, ist es die Aufgabe der Betreuer, die Studierenden zu beurteilen und zu benoten. Es gibt verschiedene Kriterien zur Leistungsbewertung: (1) Kooperationsfähigkeiten, (2) Qualität des Designs und der Implementierung, (3) Grad der Übereinstimmung der fertigen Software mit den Designdokumenten und (4) benötigte Zeit. Um diese Kriterien aber vernünftig und gerecht bewerten zu

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

können, muss die Nachvollziehbarkeit für die Betreuer unbedingt gewährleistet sein. In klassischen Softwaretechnik-Projekten, bei denen einer Gruppe eine Aufgabe gestellt wird und die am Ende ein fertiges Produkt liefert, ist es nahezu unmöglich, die Studierenden am Ende individuell zu bewerten. Mit der hier vorgestellten Plattform wird das anders: Der Betreuer ist in der Lage, die benötigten Daten über die Kommunikationsbeteiligung aus der persistenten Kommunikation in CURE zu extrahieren (Chat-Protokolle, Mail-Archive, alle Versionen der Wiki-Seiten). Zusätzlich hat der Betreuer Einsicht in alle Tracker von CodeBeamer und in das CVS-Archiv. (Beispielsweise bietet CodeBeamer grafische Visualisierungen der Vorgänge im CVS-Archiv.) Außerdem kann die Architektur des fertigen Systems mit Hilfe von Reverse-Engineering-Werkzeugen mit dem ursprünglichen Entwurf verglichen werden. Ein Plug-in, das speziell für diesen Zweck entworfen wurde, ist JRefleX [Won03]. JRefleX kann sowohl eine Kollaborationsanalyse (wie hat das Team zusammengearbeitet) als auch eine Evolutionsanalyse (wie hat sich das System über die Zeit hin entwickelt) durchführen. Wir werden JRefleX noch evaluieren und es dann in unsere Plattform integrieren.

4 Erfahrungen und Ausblick

Bei der Analyse der verschiedenen Plug-ins haben wir festgestellt, dass die meisten Plug-ins gute Lösungen für eine einzelne, spezifische Problemstellung bieten. In unserer Situation reicht es jedoch nicht aus, dass eine Unterstützung in Eclipse existiert. Stattdessen brauchen wir eine Lösung, in der die einzelnen Plug-ins enger miteinander zusammenarbeiten. So gibt es zwar einige Instant Messaging Plug-ins für Eclipse, aber sie sind untereinander nicht austauschbar, da sie für diesen Zweck auf einem Framework wie Koi[10] basieren müssten.

Ähnliches gilt für Kollaborations- und Projektmanagement-Werkzeuge: Jede Lösung implementiert ihre eigenen Diskussionsforen, Kalender, Dokumentmanager usw. Aber es ist unmöglich, diese mit anderen Foren oder Kalendern zu integrieren. Das führt zu einer Situation, in der unsere Softwaretechnik-Plattform mindestens vier (!) Diskussionsforen anbietet: (1) klassische Newsgroups, (2) ein Diskussionsforum in der allgemeinen Lehrplattform, (3) die Mail- und Chat-Lösung in CURE und (4) die Diskussionsforen innerhalb von CodeBeamer. Das verwirrt natürlich und muss deshalb von *einer* allgemeineren, integrierten Lösung abgelöst werden.

CURE wurde bereits einmal alleine und wird gerade ein weiteres Mal in Verbindung mit CodeBeamer für die Unterstützung von Studierenden in Softwaretechnik-Praktika eingesetzt. Im ersten Softwaretechnik-Praktikum arbeiteten 6 Gruppen mit 5 bis 7 Studierenden an der Realisierung eines synchronen kooperativen Spiels. Die Gruppen nutzten CURE hauptsächlich zur Kommunikation und zur Dokumentation von wichtigen Entwurfsentscheidungen. Alle Gruppen berichteten, dass CURE die Gruppenkommunikation erleichtert hat. Alle Gruppen gaben zu, einzelne Arbeitspunkte und Zuständigkeiten nicht intensiv genug in CURE dokumentiert zu haben. Letzteres führte

[10] *http://www.eclipse.org/koi/*

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

dazu, dass die Aufgabenverteilung in den einzelnen Gruppen nicht für alle Mitglieder klar war und es so zu Missverständnissen bei der Koordination gekommen ist. Bei der Begutachtung der einzelnen Gruppenergebnisse war jedoch auffällig, dass die Gruppen mit dem höchsten Kommunikationsaufkommen in CURE die besten Ergebnisse abgeliefert hatten.

Im zweiten Softwaretechnik-Praktikum arbeiten zurzeit 3 Gruppen mit jeweils 7 Studierenden an Erweiterungen der CURE-Plattform, die synchrones kollaboratives Lernen ermöglichen sollen. Aufgrund der Erfahrung des ersten Praktikums wurde in der ersten Präsenzphase des Praktikums CodeBeamer vorgestellt und die Wichtigkeit einer klaren Aufgabenverteilung herausgestellt. Die Gruppen konnten sich frei entscheiden, ob sie zu ihrer Koordination CodeBeamer einsetzen wollten. Eine der Gruppen hat sich für die Nutzung von CodeBeamer entschieden, während die anderen Gruppen ihre Zuständigkeiten nur in CURE dokumentieren wollten.

Die Gruppe, die CodeBeamer verwendet, stellte bereits in der Präsenzphase erste Aufgaben in den Task Tracker von CodeBeamer ein und machte im Vergleich zu den anderen Gruppen in der Anfangsphase des Praktikums die schnellsten Fortschritte. Ob dies an der transparenteren Aufgabenverteilung lag, muss durch eine endgültige Evaluation geklärt werden. Nach etwa zwei Monaten stellte diese Gruppe jedoch die Nutzung von CodeBeamer ein. Die Koordination erfolgte von diesem Zeitpunkt an nur noch über einen wöchentlich stattfindenden Team-Chat. Die Gruppenmitglieder begründeten diese Entscheidung damit, dass CodeBeamer zu Beginn des Praktikums zur klareren Aufgabenverteilung beigetragen hat. Später jedoch seien die einzelnen Aufgaben und Zuständigkeitsbereiche klar gewesen, so dass zur Koordination der wöchentliche Team-Chat und die Benachrichtigungen durch CVS über den Gruppenfortschritt gereicht hätten.

5 Zusammenfassung

Wir haben gezeigt, wie wir Eclipse als die Basisumgebung für Softwaretechnik-Projekte in der Fernlehre verwenden möchten. Zur Gesamtplattform gehören neben Eclipse die beiden großen Komponenten CURE, das auch zurzeit schon die kooperative Lernplatform der FernUniversität ist, und das Projektmanagementsystem CodeBeamer. CURE wurde bereits in Eclipse integriert, wodurch der Benutzer nun innerhalb von Eclipse auf die Wiki-, Mail-, Chat- und Kalenderfunktionen von CURE zugreifen kann, ohne dazu den Browser öffnen zu müssen. CodeBeamer wird mit einem Eclipse Plug-in ausgeliefert, das die Tracker in eine Sicht in Eclipse integriert.

Erste Erfahrungen haben gezeigt, dass CURE und CodeBeamer Studierende bei der verteilten Softwareentwicklung unterstützen und deren Kollaboration erleichtert. Zusammen mit weiteren Plug-ins, die entsprechend erweitert oder angepasst werden müssen, wird diese Plattform die Durchführung von Softwaretechnik-Projekten (nicht nur in der Fernlehre) in allen Phasen erleichtern: Studierende werden in der Lage sein, einfacher und effizienter zu kollaborieren, sie werden ihr Projekt leichter verwalten können und ein besseres Endprodukt abgeben können. Da nahezu sämtliche Vorgänge

Klaus-Peter Löhr · Horst Lichter (Hrsg.) (2005): Software Engineering im Unterricht der Hochschulen, SEUH 9, Aachen, dpunkt.verlag, Heidelberg

persistent gehalten werden (Kommunikation über Chat und Mail, Tracker, Versionskontrolle), wird es für die Betreuer leichter sein, einzelne Studierende separat zu bewerten.

Die vorgestellte integrierte Entwicklungsumgebung mit CURE und CodeBeamer wird zurzeit zum ersten Mal eingesetzt. Über erste Erfahrungen mit dem Einsatz der Entwicklungsumgebung haben wir berichtet. Eine abschließende Evaluation der Kollaboration in den Gruppen wird zeigen, ob Studierende weitere Hilfen und Werkzeuge benötigen und ob sie die bereitgestellten Hilfsmittel akzeptieren.

Literatur

[Bou03] P. Bouillon, M. Burger, A. Zeller. Automated debugging in eclipse. In: Proceedings of the 2003 OOPSLA Workshop on Eclipse TechnologyeXchange, S. 1-5, 2003.

[Bou04] N. Boulila, A. H. Dutoit, B. Brügge. Scoop: A framework for supporting synchronous collaborative object-oriented software design process. In: Proceedings of the 2004 ASE Workshop on Cooperative Support for Distributed Software Engineering Processes, S. 39-53, 2004.

[Che03] L.-T. Cheng, S. Hupfer, S. Ross, J. Patterson. Jazzing up eclipse with collaborative tools. In: Proceedings of the 2003 OOPSLA Workshop on Eclipse Technology eXchange}, S. 45-49, 2003.

[Gre02] S. Greenberg, M. Roseman. Using a room metaphor to ease transitions in groupware. In: Beyond Knowledge Management: Sharing Expertise, Cambridge, MA, 2002. MIT Press.

[Haa04a] J. M. Haake, T. Schümmer, M. Bourimi, B. Landgraf, A. Haake. CURE – Eine Umgebung für selbstorganisiertes Gruppenlernen. i-com Zeitschrift für interaktive und kooperative Medien, September 2004.

[Haa04b] J. M. Haake, T. Schümmer, M. Bourimi, B. Landgraf. Supporting flexible collaborative distance learning in the CURE platform. In: Proceedings of the Hawaii International Conference On System Sciences (HICSS-37), 2004.

[Leu01] B. Leuf, W. Cunningham. The WIKI way. Addison-Wesley, Boston, MA, USA, 2001.

[Pfi98] H. Pfister, C. Schuckmann, J. Beck-Wilson, M. Wessner. The metaphor of virtual rooms in the cooperative learning environment CLear. In: *Cooperative Buildings*, LNCS 1370, S. 107-113. Springer-Verlag Berlin Heidelberg, 1998.

[Teu95] S. Teufel, C. Sauter, T. Mühlherr, K. Bauknecht. Computerunterstützung für die Gruppenarbeit. Addison-Wesley, 1995.

[Tie01] D. A. Tietze. A Framework for Developing Component-based Co-operative Applications. Dissertation, Technische Universität Darmstadt, 2001.

[Won03] K. Wong, W. Blanchet, Y. Liu, C. Schofield, E. Stroulia, Z. Xing. JRefleX: Towards supporting small student software teams. In: Proceedings of the 2003 OOPSLA Workshop on Eclipse Technology eXchange, S. 50-54, 2003.