J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

Räumlich verteilte Software-Entwicklung unter experimenteller Betrachtung verteilter Inspektionen – Ein Erfahrungsbericht

Claudia Schlumpberger
Abt. Software-Prozessgestaltung
DaimlerChrysler Forschungszentrum Ulm
Postfach 23 60
89081 Ulm
claudia.schlumpberger@daimlerchrysler.com

Dietmar Ernst
Abteilung Programmiermethodik und Compilerbau
Fakultät für Informatik, Universität Ulm
89069 Ulm
dernst@uni-ulm.de

Zusammenfassung

Im Rahmen einer bereits langjährig bestehenden Kooperation zwischen dem DaimlerChrysler Forschungszentrum Ulm und der Universität Ulm wird in jedem Semester interessierten Studierenden des Informatik-Hauptstudiums ein Praktikum zum Thema "Experimentelles Software Engineering für eingebettete Systeme" angeboten. Gestaltet wird dieses von einem Mitarbeiter des DaimlerChrysler Forschungszentrums und einem Mitarbeiter der Universität Ulm, Fakultät für Informatik. Jedes Praktikum fokussiert ein anderes Thema aus dem Bereich des Software Engineering mit entsprechend unterschiedlichen Inhalten und Fragestellungen.

Im Sommersemester 2002 lag der Schwerpunkt des Praktikums auf der Betrachtung räumlich verteilter Entwicklungen und im Rahmen von entwicklungsbegleitenden Qualitätssicherungsmaßnahmen in der Durchführung von räumlich verteilten Inspektionen unter Zuhilfenahme eines Videokonferenzsystems.

Der Leser erhält einen kurzen Überblick über Ziel und Aufbau dieses Praktikums, dessen Vorbereitung und Durchführung sowie einen Einblick in die hierbei gewonnenen Erfahrungen.

1 Einleitung

Industrielle Forschungseinrichtungen wie das DaimlerChrysler Forschungszentrum Ulm sind hauptsächlich auf die industrienahe Forschung ausgerichtet und bringen daher sehr viele Fragestellungen aus den einzelnen Entwicklungsbereichen mit sich.

J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

Im Rahmen von Kooperationen mit universitären Einrichtungen sollen diese Fragestellungen aufgenommen und anhand empirischer Untersuchungen in Form von angebotenen Praktika untersucht werden.

1.1 Motivation und Zielsetzung

Der Einsatz von Software in Form von eingebetteten Systemen – vor allem im verkehrstechnischen Bereich – spielt eine immer größere Rolle. Auf diese Herausforderung muss auch der Hochschul-Nachwuchs entsprechend vorbereitet werden. So organisieren je ein Mitarbeiter des DaimlerChrysler Forschungszentrums Ulm, Abteilung „Software Prozessgestaltung“ und der Universität Ulm, Abteilung „Programmiermethodik und Compilerbau“, seit 1996 jedes Semester ein Praktikum zum übergeordneten Thema „Experimentelles Software Engineering für eingebettete Systeme“. Dabei werden in Anlehnung an aktuelle industrielle Forschungsfragen wechselnde Aktivitäten mit interessierten Informatikstudierenden aus dem Hauptstudium durchgeführt. So wurden in den vergangenen Praktika u.a. folgende Themen behandelt:

- Vergleich statischer und dynamischer Softwareprüfungen in eingebetteten Systemen
- Textuelle Spezifikation versus modellbasierte Spezifikation
- Partielle Prüfung umfangreicher Systeme
- Design und Implementierung mittels strukturierter und objektorientierter Methoden im Hinblick auf eingebettete Systeme

Ziel ist es, die hierbei gewonnenen Erkenntnisse in die industrielle Praxis rückzuspiegeln, um einen Beitrag zu deren Verbesserung zu leisten. Aus Sicht der Lehre sollen die am Praktikum beteiligten Studierenden für Themen aus der Praxis sensibilisiert werden und neue Lösungsansätze kennen lernen.

1.2 Einbettung in das Curriculum

Während des Informatik-Grundstudiums an der Universität Ulm besuchen alle Studierenden im 3. Fachsemester das „Softwaregrundpraktikum“ [1]. Als Lehrziele stehen dabei die Vermittlung erster Kenntnisse und Erfahrungen bei der methodischen Entwicklung großer Software-Systeme im Team im Vordergrund.

Im Hauptstudium bildet die Vorlesung „Softwaretechnik“ die Basis für das Vertiefungsgebiet „Programmiermethodik“. Weitere, darauf aufbauende Vorlesungen in diesem Gebiet sind „Requirements Engineering“, „Software Qualität“ und „Management von Softwareprojekten“. Parallel zu diesen Vorlesungen wird empfohlen, eines der angebotenen Praktika, u.a. das „Experimentelle Software Engineering für eingebettete Systeme“ zu besuchen, um die in den Vorlesungen erlangten Kenntnisse praktisch zu erproben und zu vertiefen.

J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

2 Vorbereitungsphase

Dieses Kapitel beschreibt die Themenfindung und Planung des Praktikums im Sommersemester 2002. In diesem Zusammenhang wird auch die Verknüpfung zu einem aktuellen Thema aus der industriellen Praxis gezogen.

2.1 Themenfindung

In einem multinationalen Konzern wie DaimlerChrysler findet die Produktentwicklung nicht nur lokal begrenzt, sondern weiträumig verteilt statt. So müssen die Entwickler über weite Strecken und auch Kontinente miteinander kommunizieren.

Ebenso wie der Hardware-Entwicklungsprozess ist auch der Software-Entwicklungsprozess als ein komplexes Gefüge aus verschiedenen internen Entwicklungsmannschaften und externen Zulieferern zu betrachten. Diese wiederum nehmen oft gleichzeitig Auftraggeber- und Auftragnehmerrolle ein, wobei auch diese Rollen häufig wechseln.

So werden einzelne Subsysteme verteilt an den diversen Standorten entwickelt und sind anschließend zu einem funktionsfähigen Gesamtsystem zu integrieren. In der Regel treten dabei folgende Probleme auf:

- Unklarheiten bei der Spezifikation, Realisierung und Integration
- Hohe Aufwände für Kommunikation und viele Missverständnisse

Hohe Organisations- und Zeitaufwände für Reviews/Inspektionen

Alle Beteiligten haben jedoch das gemeinsame Ziel, ein qualitativ hochwertiges Software-Produkt im geplanten Zeit- und Kostenrahmen zu entwickeln. Es ist jedoch offensichtlich, dass oben genannte Faktoren die Erreichung dieses Ziels erschweren.

Aus diesen Überlegungen heraus entstand die Idee, im Rahmen des Software-Praktikums “Experimentelles Software Engineering für eingebettete Systeme” zu beobachten, welche Hindernisse, Probleme oder weitere Fragestellungen sich bei einer räumlich verteilten Entwicklung und der Durchführung von räumlich verteilten Inspektionen ergeben.

2.2 Planung des Praktikums

Folgende Punkte wurden in der Planungsphase zum Praktikum festgelegt:

Zeitlicher Rahmen. Das Praktikum wird mit einer Gesamtdauer von 12 Wochen und einem Umfang von 4 Semesterwochenstunden (SWS) angelegt. Da der Gesamtarbeitsumfang ungefähr doppelt so hoch wie die SWS sein sollte, wird für das Praktikum eine wöchentliche Arbeitsbelastung für die Studierenden von 8 Stunden veranschlagt.

Maßnahmen zur Qualitätssicherung. Die Teilnehmer sollen im Rahmen des Praktikums für das Thema „Software-Qualitätssicherung“ sensibilisiert werden. Im Zeitplan sind hierfür Slots zur Vermittlung theoretischer Grundlagen

J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

zu reservieren. Weiterhin finden nach jeder wichtigen Entwicklungsphase gegenseitige Reviews [2] der einzelnen Subsysteme statt. Auch hier ist ausreichend Zeit für die Vor- und Nachbereitung der Reviews einzuplanen. Für die technische Umsetzung räumlich verteilter Reviews wird entweder ein Videokonferenzsystem oder Netmeeting in Erwägung gezogen.

Festlegen einer Spezifikation für das zu entwickelnde System. Im Rahmen früherer Praktika betrachtete man eingebettete Systeme aus sehr unterschiedlichen Anwendungsgebieten, wie z.B. eine Mikrowelle, einen Anrufbeantworter, einen Kopierer, eine Parkhausschranke oder die Innenlichtsteuerung eines PKWs. In diesem Praktikum wurde jedoch ein etwas umfangreicheres System als die bisher verwendeten benötigt. Deshalb erweiterte man die früher verwendete „Parkhausschranke“ zu einer kompletten Parkhaussteuerung und erstellte eine entsprechende textuelle Systembeschreibung [3]. Dieses System lässt sich auch sehr gut in verschiedene Subsysteme unterschiedlicher Größe zerlegen.

Auswahl einer geeigneten Entwicklungsumgebung. Die Erfahrungen der früheren Praktika haben gezeigt, dass die Studentengruppen meist sehr heterogen in Bezug auf Erfahrungen mit Programmiersprachen und Entwicklungsumgebungen sind. Um ein einheitliches Basislevel für alle Teilnehmer zu schaffen, fällt die Auswahl auf Statemate, ein Werkzeug, das es erlaubt, ein simulationsfähiges Modell mittels Statecharts [4] zu erstellen. Dieses Werkzeug kommt im Bereich der Abteilung „Programmiermethodik und Compilerbau“ vielfach zur Anwendung. Damit ist auch sichergestellt, dass die Grundlagen den Teilnehmern nach kurzer Einarbeitungszeit vermittelbar sind.

Festlegen der Entwicklungsphasen. In der zur Verfügung stehenden Zeit nach Punkt 1 ist eine Komplettentwicklung eines umfangreicheren Systems gemäß Punkt 3 zu aufwändig. Daher beschränkte man sich auf die Analyse- und Entwurfsphase mit dem Ziel eines ablauffähigen und testbaren Modells.

Praktikumsziele. Die verteilte Entwicklung soll anhand folgender Punkte bewertet werden:

- Zielerreichung im vorgegebenen Zeitrahmen und mit erwarteter Qualität
- Entwicklungs- und Testaufwände in späteren Entwicklungsphasen
- Detaillierungsgrad und Angemessenheit der gegebenen Informationen
- Motivation und Engagement der Beteiligten bei der Entwicklung der Subsysteme und im Rahmen der durchgeführen Reviewsitzungen
- Als Vergleichspunkte sollen u.a. Erfahrungswerte aus früheren Praktika dienen.

Das Ergebnis der Planung ist aus der Abbildung 1 ersichtlich.

Eine Schwierigkeit in der Planung und Vorbereitung des Praktikums besteht darin, dass die Teilnehmerzahl nur schwer vorherzusagen ist, da sich Studierende oft erst zu Beginn des Praktikums anmelden. Daher wurde das Praktikum zwar mit zwei Gruppen geplant, aber auch eine dritte Gruppe wäre problemlos möglich gewesen.

J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

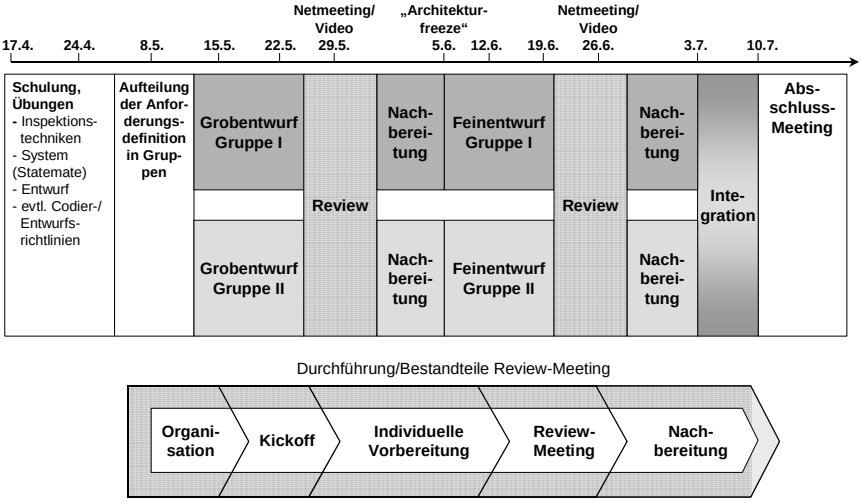

Abb. 1: Konzept des geplanten Ablaufs

3 Durchführung des Praktikums

3.1 Teilnehmende Studierende

Letztendlich haben 5 Studentinnen an dem Praktikum teilgenommen. Vier von ihnen waren im 6. Semester, also eher zu Beginn des Hauptstudiums, und haben parallel zum Praktikum die Vorlesung „Softwaretechnik“ gehört. Eine Studentin war im 8. Semester.

Die Studentinnen hatten zwar Grundkenntnisse über Zustandsautomaten aus der Vorlesung “Theoretische Informatik”, aber keine hatte bisher mit Statemate gearbeitet. So wurden die Teilnehmerinnen an den ersten Terminen anhand von praktischen Übungen mit der Entwicklungsumgebung und den Besonderheiten von Statemate vertraut gemacht. Die Zeit zwischen den Treffen mussten die Teilnehmerinnen dazu nutzen, eigenständig kleinere Aufgaben zu lösen. In den Treffen selbst wurden dann aufgetretene Probleme und mögliche Lösungen aufgezeigt.

Generell war gewährleistet, dass Rücksprachen mit den Betreuern auch außerhalb der festgelegten Termine möglich waren. Weiterhin wurde eine gemeinsame E-Mail-Plattform eingerichtet, unter der während des gesamten Praktikums organisatorische Änderungen, Fragen oder Probleme eingestellt werden konnten.

3.2 Einblick in die Systemspezifikation

Die Systemspezifikation zum System “Parkhausschranke” beschreibt in Anlehnung an Parkhäuser, wie sie in Städten oder größeren Einkaufszentren zu finden sind, ein

J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

fiktives Parkleitsystem. Kunde bzw. Auftraggeber ist das Kaufhaus “Easybuy”, welches für seine Kunden ein neues Parkhaus erbaut hat. Die Zufahrt erfolgt über ein Schrankensystem mit Chipausgabe, die Ausfahrt erfolgt erst nach Bezahlung der Parkgebühr an einem Kassenautomaten und Entgegennahme des Chips an der Ausfahrtsschranke. Erschwerend kommt hier allerdings hinzu, dass es aus baurechtlichen Gründen nur eine gemeinsame Ein- und Ausfahrt gibt, die über ein System aus Induktionsschleifen und Ampeln geregelt werden muss.

Im Parkhaus selbst muss es auch eine Zentrale (Operator) geben, die den ganzen Ablauf der Zufahrt, Bezahlung und Ausfahrt kontrolliert und in Notfällen auch einen manuellen Eingriff in das System zulässt.

3.3 Gruppeneinteilung (Aufteilung in Subsysteme)

Nach der erfolgreichen Einarbeitung in die Entwicklungsumgebung erhielten die Teilnehmerinnen die Anforderungsspezifikation zum System „Parkhausschranke“ mit der Aufgabe, sich ein Konzept zum Zerlegen des Systems in Subsysteme und einer möglichen Aufteilung in Entwicklergruppen zu machen.

Die Teilnehmerinnen formierten sich eigenständig in eine Zweiergruppe, die für das System “Bediencontrol” (Kassenautomat und Operator) verantwortlich war, und in eine Dreiergruppe, die die Entwicklung des Systems “Schrankencontrol” (Ein- und Ausfahrt, Ampelregelung und Induktionsschleifen) übernahm.

Zusätzlich gab es noch das Subsystem „Chipverwaltung“, das von den Betreuern entwickelt wurde, da dieser Teil einerseits detailliertere Statemate-Kenntnisse erforderte und andererseits die zu entwickelten Subsysteme sonst zu umfangreich gewesen wären. Auch die komplette Hardwareumgebung, die für eine spätere Simulation nötig war, wurde von Seiten der Betreuer modelliert.

3.4 Verteilte Entwicklung der Subsysteme

Beide Gruppen entwickelten ihre jeweiligen Subsysteme völlig autark und ohne gegenseitige Nebenabsprachen. Vor allem in der Anfangsphase der Entwicklung waren aber die wöchentlichen Treffen wichtig, um die Schnittstellen zwischen den Subsystemen genauer zu definieren. Die Treffen dienten hauptsächlich dem Erfahrungsaustausch bezüglich Problemen mit der Entwicklungsumgebung, der Vorbereitung auf Reviews und Terminabsprachen, aber auch dem gegenseitigen Kennenlernen.

Ein Termin wurde dazu genutzt, die Gruppen mit der Erstellung eines Qualitätsmodells vertraut zu machen. Hier musste jede Entwicklergruppe für ihr eigenes System jeweils aus Auftraggeber- und Auftragnehmersicht Qualitätsmerkmale priorisieren. Anhand dieser Priorisierung wurden von den Betreuern Checklisten erstellt, die den Gruppen als Hilfsmittel zur Inspektion des jeweils anderen Subsystems im Rahmen der Reviewvorbereitung zur Verfügung gestellt wurde.

J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

3.5 Organisation der verteilten Inspektionen/Reviews

Bevor das jeweils andere Subsystem inspiziert wurde, hatten die Entwicklergruppen die Aufgabe, gemeinsam einen Termin zum „Modell-Freeze“ festzulegen, um sicherzustellen, dass vor dem Review keine Änderungen mehr in die Modelle einfließen. Allen Beteiligten wurden vorab Blanko-Protokolle in Form von Einzelprüfberichten ausgehändigt, auf denen sie die bei der Inspektion gefundenen Mängel notieren sollten. Im Review selbst wurden diese dann von einem Protokollanten, dessen Rolle einer der beiden Betreuer übernahm, in einem gemeinsamen Reviewprotokoll auf einem separaten Notebook zusammengeführt. Die Moderation übernahm jeweils der andere Betreuer.

Bei der Durchführung der räumlich verteilten Reviewsitzungen kamen die umfangreichen technischen Mittel des Universitätsrechenzentrums Ulm zur Anwendung. Hier bestand die Möglichkeit, ein Videokonferenzsystem, bestehend aus Videokameras und -monitoren – wahlweise steuerbar über ISDN oder Internet (TCP/IP) – zu nutzen.

Damit beide Gruppen die zu besprechenden Teile des Modells gleichzeitig erfassen können, wurde je ein Notebook verwendet und das Programm VNC (Virtual Network Computing) [5] von den AT&T Research Labs Cambridge eingesetzt. Dieses kostenlose Programm ermöglicht es, einen Server-Desktop auf mehrere Client-Rechner zu projizieren, so dass der Desktop und die darauf ablaufenden Programme von jedem der Clients gesteuert werden können. Während der Reviews war es also möglich, dass beide Gruppen von ihrem Notebook das Werkzeug Statemate bedienten und dass die jeweils andere Gruppe dies mitverfolgen konnte.

Um das sich gerade im Review befindende Modell für alle Beteiligten gut sichtbar zu machen, wurden an jeden Notebook zwei Beamer angeschlossen, die das Modell an einer Leinwand vergrößert darstellten. Dabei platzierte man die Leinwand direkt neben der Kamera bzw. dem Monitor. Somit war gewährleistet, dass alle Beteiligten – einschließlich Protokollant und Moderator – immer frontal zur Kamera gerichtet waren, sich also niemals umdrehen oder umsetzen mussten. Die Abbildung 2 stellt den Aufbau des Systems sowie das benötigte Equipement grafisch dar.

4 Erfahrungen

Rückblickend konnten viele Zweifel, die vorab bezüglich verteilter Entwicklung und des Einsatzes eines Videokonferenzsystems vorhanden waren, entkräftet werden. Nachfolgende Kapitel listen die wichtigsten Erfahrungen aus dem Praktikum auf.

4.1 Qualität der verteilt entwickelten Software

Oft herrschen Zweifel über die Güte verteilt entwickelter Software im Vergleich zu nichtverteilter Software. Bei entsprechenden Rahmenbedingungen wie beispielsweise die Durchführung regelmäßiger Reviews – in diesem Zusammenhang auch strikt festgelegte Termine für Softwarefreezes – sowie genügend Zeit zur Vorbereitung eines Reviews ist die Qualität verteilt entwickelter Software nicht schlechter.

J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

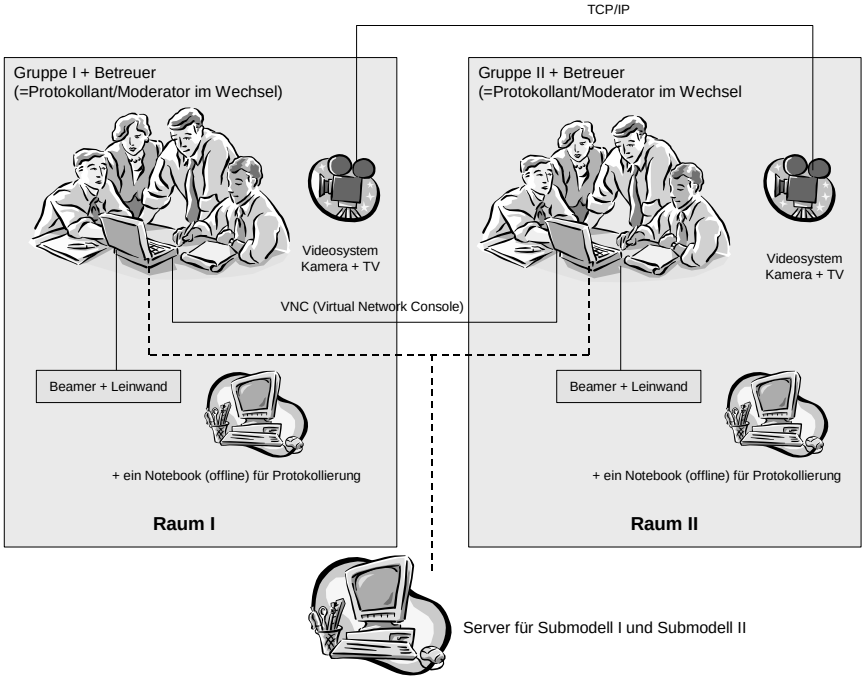

Abb. 2: Aufbau der Videokonferenz

Das Verständnis für das Gesamtsystem bleibt ebenso aufrechterhalten, denn die Einarbeitung in das jeweils andere Subsystem bei der Vorbereitung auf die Reviews und in diesem Rahmen die Bestimmung von Qualitätsmerkmalen (auch für das Fremdsystem) stärken das Gesamtsystemverständnis in beiden Entwicklergruppen. Dieser Umstand wirkt sich positiv auf die Qualität des Endproduktes aus.

4.2 Entwicklungs-, Integrations- und Testaufwand

Vor allem in der ersten Review-Sitzung, bei der die Entwicklung schon angelaufen war, wurden rückwirkend einige Unstimmigkeiten in der Spezifikation gefunden. Daraus resultierten wiederum Mehraufwände bei der Implementierung. Langjährige Untersuchungen aus dem Bereich des Software-Engineering bestätigen diesen Umstand.

Die Vermutung liegt daher auch hier nahe, dass ein zusätzliches Review der Spezifikation den Mehraufwand bei der laufenden Entwicklung gemindert hätte. Gerade bei einer verteilten Entwicklung ist eine vollständige und konsistente Spezifikation ein wesentlicher Bestandteil zur Absicherung der Qualität. Daher sollte gerade hier die Durchführung von Spezifikationsreviews (unter Umständen auch mehrere) verpflichtend sein.

J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

Die Teilnehmerinnen entwickelten ihre "eingebetteten Subsysteme" völlig autark. Außer Event- und Variablenlisten erfolgte keinerlei Dokumentenaustausch. Negativ wurde angemerkt, dass die Bedienoberfläche, die für eine einfache und übersichtliche Simulation notwendig ist, erst spät von den Betreuern zu Verfügung gestellt wurde und deshalb die Simulation und ein Test vorher nur schwer möglich war. Schätzungen ergaben, dass sich die Testphase deshalb um den Faktor 2 verlängerte.

Die beiden Reviews der Modelle hingegen trugen wesentlich dazu bei, dass nach der Integration der Subsysteme nur noch einzelne, überschaubare Änderungen vorgenommen werden mussten.

4.3 Einsatz eines Videokonferenzsystems in Reviewsitzungen

Die vorab gehegten Zweifel, ein Videokonferenzsystem schüre die Technologieangst und mindere die Diskussionsfreudigkeit, konnte nicht bestätigt werden. Bereits beim ersten Review trat nach kürzester Zeit ein gewisser Gewöhnungseffekt ein. Besonders hervorzuheben ist, dass hier eine wesentlich diszipliniertere Diskussionskultur als bei einem On-Place-Meeting gepflegt wurde. Die Teilnehmerinnen fielen sich nicht gegenseitig ins Wort und beschränkten sich auf das Wesentliche.

In diesem Zusammenhang stellt sich auch die Frage, ob die Akzeptanz eines derartigen Systems immer noch so hoch ist, wenn es in Situationen genutzt wird, in denen eine gewisse "Medienflexibilität" gefordert ist, z.B. in Brainstorming-Situationen, in denen vieles auf Papier oder anderen Medien skizziert wird.

Obwohl der Einsatz des in diesem Praktikum eingesetzten Systems mit einem gewissen Mehraufwand bezüglich Vorbereitung, Schulung, Durchführung und vor allem technischer Mittel verbunden ist, erfüllte das System seinen Zweck optimal.

4.4 Die Entwicklungsumgebung

Statemate hat sich als Entwicklungsumgebung in diesem Praktikum hauptsächlich zur Entwicklung des Subsystems "Schrankencontrol" (da dies als ein wirkliches "eingebettetes System" bezeichnet werden kann) bewährt.

Der Lerneffekt bei den Teilnehmerinnen war hoch und sie waren in der Lage, in der doch begrenzten Zeit ein anspruchsvolles System zu modellieren. Jedoch ist der Einsatz nur bei Studierenden aus dem Hauptstudium sinnvoll, da sie erst hier Kenntnisse für die Umsetzung einer Spezifikation in Zustandsautomaten mitbringen. Dennoch ist die 3-wöchige Einlernphase notwendig. Weiterhin ist wichtig, dass während der gesamten Entwicklungszeit immer ein fachlich versierter Ansprechpartner für kurzfristige Fragen zur Verfügung steht.

J. Siedersleben, D. Weber-Wulff (Hrsg.) (2003): Software Engineering im Unterricht der Hochschulen, SEUH 8, Berlin, dpunkt.verlag, Heidelberg

5 Fazit und Ausblick

Die Erkenntnisse aus dem Praktikum zum Thema “Räumlich verteilte Entwicklung unter experimenteller Betrachtung räumlich verteilter Inspektionen“ bestätigen einerseits bereits bekannte Probleme aus dem Bereich des Software Engineering, andererseits konnte man neue Erkenntnisse im Umgang mit Entwicklungsumgebung und neuen Technologien gewinnen.

Eine Befragung der Teilnehmerinnen ergab, dass der Aufbau des Praktikums in der hier beschriebenen Form für gut befunden wurde. Als ein sehr positiver Aspekt wird hervorgehoben, dass der gesamte Entwicklungsprozess zu jedem Zeitpunkt transparent gehalten und somit kontrollierbar war. Negativ hingegen wurde bewertet, dass die Bedienoberfläche erst sehr spät geliefert wurde und sich so der Testaufwand des Gesamtsystems erhöhte. Schon die Ausgabe einer Skizze der Oberfläche vorab wäre für die Testvorbereitung hilfreich gewesen.

Zwar wurde der Einsatz von Statemate ebenso positiv bewertet, allerdings gibt es auf Seiten der Teilnehmerinnen Bedenken bezüglich des Gebrauchswertes im weiteren Verlauf des Studiums und Berufes.

Interessant ist auch die Erkenntnis der Studentinnen, dass die wöchentlichen Treffen den Revieweffekt etwas mindern, da es hier unbemerkt immer wieder zu gemeinsamen Absprachen kam. Offen bleibt dabei die Frage, wie ein solches Experiment mit strikt getrennten Gruppen, z.B. im Rahmen einer Zusammenarbeit mit anderen Universitäten, verlaufen würde.

Allgemein wird man selten die Gelegenheit haben, mit einer ausschließlich weiblichen Entwicklergruppe zu arbeiten. Die Studentinnen arbeiten normalerweise in der Mehrzahl mit männlichen Kommilitonen zusammen und empfanden in diesem Praktikum den Druck des “Sich-Beweisen-Müssens” als wesentlich weniger stark. Weiterhin wurde das Arbeiten mit den anderen Gruppenpartnerinnen als organisierter und teamorientierter empfunden.

Literatur

1. Schwarz, M.; Schulte, W.: Realistische Aufgabenstellungen für das Softwaregrundpraktikum, Workshop des German Chapter of the ACM und der Gesellschaft für Informatik: "Software Engineering im Unterricht der Hochschulen SEUH'97", Rostock, 27-28. Februar, Berichte des German Chapter of the ACM, Band 48, B.G.Teubner Stuttgart, 1997, S.94-104.
2. Frühauf, K., Ludewig J., Sandmayr H.: Software-Prüfung: Eine Anleitung zum Test und zur Inspektion, B.G.Teubner Stuttgart, 1995.
3. Houdek, F.: Softwareanforderungen und Systemspezifikation „Parkhausschranke“, DaimlerChrysler Forschungszentrum Ulm, 2002.
4. Harel, D.: Statecharts: A visual formalism for complex systems, Science of Computer Programming, Band 8, 1987, S. 231-274.
5. Richardson T.; Stafford-Fraser, Q.; Wood, K.R.; Hopper, A.: Virtual Network Computing, IEEE Internet Computing,, Band 2, Nr. 1, Jan/Feb 1998, S. 33-38.