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

Software Engineering kompakt: interne Schulungen bei sd&m

Christiane Stutz, Axel Burghof
sd&m Research, software design & management AG, München
[Christiane.Stutz|Axel.Burghof]@sdm.de

Zusammenfassung

Für viele Mitarbeiter in Software-Unternehmen stellen die Möglichkeiten zur Weiterbildung ein wichtiges Kriterium für die Zufriedenheit mit ihrem Arbeitsplatz dar. Das interne Schulungsangebot eines Unternehmens muss dabei zwei Anforderungen erfüllen: Zum einen sucht der Mitarbeiter die Möglichkeit zur Weiterbildung über sein bisheriges Fachwissen hinaus. Zum anderen möchte das Unternehmen die Mitarbeiter möglichst gezielt auf zukünftige Aufgaben vorbereiten. Dieser Artikel stellt das interne Schulungswesen eines Münchner Softwarehauses vor. Wir erläutern das Konzept der so genannten Schools, die in fünf Veranstaltungen zu jeweils drei bis fünf Tagen die wichtigsten Phasen der Software-Entwicklung abdecken. Dabei handelt es sich um Schulungen, die von internen Mitarbeitern gestaltet und gehalten werden und aus einem Wechsel von Vorlesungen und Übungen zusammengesetzt sind. Die Schulungsinhalte werden an einem durchgängigen Beispiel in Gruppen geübt.

1 Einleitung

Die software design & management AG (sd&m) ist ein Softwarehaus mit Hauptsitz in München und Niederlassungen in ganz Deutschland sowie einer Niederlassung in der Schweiz. Wir entwickeln Individualsoftware für betriebliche Informationssysteme und technische Anwendungen. Unsere Projekte umfassen dabei den gesamten Entwicklungsprozess von der Anforderungsanalyse bis zur Inbetriebnahme. Dafür beschäftigten wir im Jahr 2002 ca. 950 Mitarbeiterinnen und Mitarbeiter[1], wobei nur ein relativ kleiner Teil mit Verwaltungsaufgaben betraut ist, während der überwiegende Teil der Beschäftigen bei sd&m an der Durchführung von Software-Projekten beteiligt ist. Unsere Mitarbeiter sind zu einem großen Teil studierte Informatiker. Unter den Nichtinformatikern sind naturwissenschaftliche und technische Ausbildungsrichtungen am häufigsten vertreten.

[1] Im Folgenden bleiben wir der Lesbarkeit halber für Personen bei der männlichen Form, ohne dabei unsere Kolleginnen ausschließen zu wollen.

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

Die meisten Mitarbeiter von sd&m bringen aus ihrer Ausbildung bereits ein gutes Rüstzeug für den Software-Ingenieur mit: Bei der Einstellung legen wir Wert auf Programmierkenntnisse (die Sprache spielt dabei nur eine untergeordnete Rolle), die Kenntnis von Grundlagen guten Softwaredesigns sowie Erfahrung in der Teamarbeit im Rahmen von Studienprojekten oder bereits Berufserfahrung. Was kann man einem derart ausgebildeten Software-Ingenieur noch beibringen?

2 Überblick über das Schulungsprogramm

Das Ausbildungsprogramm bei sd&m umfasst drei Schienen: (1) Soziale Kompetenz und Persönlichkeit, (2) Management und (3) Software Engineering. Schulungen zur sozialen Kompetenz und Persönlichkeit werden durch einen externen Veranstalter durchgeführt. Zum Management nutzen wir sowohl externe als auch interne Schulungen. Für das Software Engineering haben wir bei sd&m eine eigene interne Schulungsart eingeführt, die so genannten "Schools", die in diesem Artikel beschrieben werden.

Diese Schools werden individuell auf die Bedürfnisse von sd&m abgestimmt. Grundlagenkenntnisse in Programmierung und Software-Entwurf werden dabei vorausgesetzt. In den Schools betrachten wir besonders gute Lösungen für immer wiederkehrende Projektaufgaben und werden so auf die neuen Herausforderungen vorbereitet. Jenseits von Projektdruck können hier elegante und zukunftsweisende Lösungen entwickelt und das Vorgehen aus bereits durchgeführten Projekten reflektiert werden.

3 Konzept der sd&m Schools

Der Name "School" steht bei sd&m für mehrtägige Schulungsveranstaltungen, bei denen in einem Wechsel aus Theorie und Praxis intensiv ein Ausschnitt aus dem Software Engineering gelehrt wird.

3.1 Die richtige Mischung

Vorlesungen und Übungen finden im Wechsel statt, wobei die Übungen mindestens die Hälfte der Zeit einnehmen. Teamarbeit ist ein wichtiger Bestandteil der Übungen – wir bilden Teams aus zwei bis fünf Teilnehmern (je nach Aufgabenstellung), die gemeinsam eine Aufgabe lösen und sich die Arbeit dabei selbstständig aufteilen. Selbst auf Schulungen darf der Aufwand für die Teamfindung nicht unterschätzt werden: In der Regel arbeiten Kollegen miteinander, die sich noch nicht aus der Projektarbeit kennen und den Arbeitsstil der anderen erst einmal kennen lernen müssen. Diesem sozialen Faktor tragen wir dadurch Rechnung, dass die Teams innerhalb einer Schulung nicht häufiger als einmal wechseln.

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

Wichtig ist die Auswahl des richtigen Beispiels für die Übungen. Viele bekannte Schulungsaufgaben langweilen durch allzu triviale Fachlichkeit. Deshalb wählen wir eine aus realen Projekten bekannte Fachlichkeit. Den Umfang des Beispiels begrenzen wir, indem wir nur einen Ausschnitt der Fachlichkeit behandeln, diesen jedoch in seiner ganzen Tiefe. Soweit möglich werden alle Inhalte an einem durchgängigen Beispiel erarbeitet.

3.2 Zeit und Muße zum Lernen

In unsere Ausbildung investieren wir viel – vor allem Zeit. Die Dauer einer School beträgt drei bis fünf Tage. Für diese Zeit ziehen wir uns ins Kloster Zangberg zurück. Dieses Kloster ist der ideale Ort für eine Schulung: Mitten im Grünen zwischen München und Mühlheim am Inn gelegen gibt es hier keinerlei Ablenkung von der Arbeit. Die technische Infrastruktur besteht aus mitgebrachten Laptops, die bei Bedarf (z.B. in der Programmer School) durch ein lokales Netzwerk verbunden werden. Internet-Anschluss gibt es im Kloster nur über Modem an der Klosterpforte.

Das Programm einer Schulung umfasst täglich mindestens acht Stunden, das Ende für die Übungen ist offen. Und so wird auch noch mancher Abend bis spät in die Nacht zum Programmieren und Reviewen von Software-Architekturen genutzt. Dafür opfern unsere Mitarbeiter nicht nur einen ruhigen Abend im Kloster, sondern auch ihre Freizeit. Denn ein Drittel der Schulungszeit gilt als Freizeit und wird durch Überstunden oder Urlaub abgegolten.

3.3 Teilnehmer und Trainer

Schools werden von sd&m für sd&m gehalten. Die Teilnehmer sind jeweils 20-25 Mitarbeiter von sd&m, gemischt aus allen Niederlassungen. Auch die Durchführung erfolgt durch Mitarbeiter von sd&m, in Einzelfällen mit Unterstützung eng verbundener externer Experten. Wir sehen für jeweils zwei bis drei Arbeitsgruppen einen Trainer von sd&m vor. So kann man sich bei den Vorlesungen abwechseln, was für Teilnehmer wie Vortragende weniger anstrengend ist, und in den Übungen stehen immer genügend Ansprechpartner für Fragen zur Verfügung.

Die Trainer werden aus allen Unternehmensbereichen und Niederlassungen von sd&m rekrutiert. Stets ist mindestens ein Trainer dabei, der die Schulung bereits gehalten hat. Dazu kommen aber auch immer Interessenten, die an der Schulung bereits früher teilgenommen haben und ihre Erfahrungen weitergeben wollen. Es gibt keine Trainer, die nur Schulungen machen, sondern alle Trainer arbeiten auch in Projekten und haben so stets einen ausreichenden Praxisbezug.

Die Mischung von Mitarbeitern über die Grenzen von Abteilungen und Niederlassungen hinweg sowie das Zusammenbringen von erfahrenen Mitarbeitern als Trainer und weniger erfahrenen Mitarbeitern als Teilnehmer trägt wesentlich zum Knüpfen persönlicher Netzwerke in der Firma bei.

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

4 Schulungsinhalte

Durch die Schools werden die wichtigsten Phasen der Software-Entwicklung abgedeckt: Studie / Spezifikation, Konstruktion, Realisierung und Test.

4.1 Specification School

Die Specification School deckt in fünf Schulungstagen die frühen Phasen Studie und Spezifikation ab. Inhalte dieser School sind:

- Techniken zum Sammeln von Anforderungen und Abgrenzung des Systemkontextes
- Modellierung von Geschäftsprozessen und Anwendungsfällen
- Datenmodellierung
- Gestaltung von Benutzeroberflächen und Dialogspezifikation

Damit werden wichtige Kernthemen der frühen Phasen der Software-Entwicklung behandelt. Grundlage für die Schulungsinhalte sind die bei sd&m erarbeiteten Bausteine zur Spezifikation (siehe [1], Kapitel 7, und [3]). Diese beschreiben das Vorgehen in der Spezifikation bei sd&m sowie die Ergebnistypen. Während die Grundlagen der Modellierung in der Schulung nur sehr kurz behandelt werden, konzentrieren wir uns auf die Faktoren für eine erfolgreiche Spezifikation: Entdecken aller wichtigen Anforderungen, Konzentration auf die wesentlichen Fakten, Wahl der richtigen Darstellungsform.

Als Beispiel wird in den Übungen eine Skischule behandelt. Die organisatorischen Prozesse der Skischule (Anmeldung, Planung von Kursen, Abrechnung) stellen eine Herausforderung dar, die mit realen Projektinhalten durchaus mithalten kann.

Der letzte Tag ist der Analyse von Projekten gewidmet. Auch ohne die konkrete Fachlichkeit des Beispielprojekts zu verstehen, erkennen die Teilnehmer nun Stärken und Schwächen im Vorgehen und der Darstellung der Ergebnisse in den Spezifikationen echter Projekte, die einige Teilnehmer mitgebracht haben.

4.2 Designer School

Für betriebliche Informationssysteme, die den größten Anteil der Projekte bei sd&m ausmachen, hat die sd&m AG aus vergangenen Projekten die Standardarchitektur Quasar [2] entwickelt, die sie ihren Mitarbeitern in den fünf Tagen der Designer School vermittelt. Dabei kommen auch allgemeine Grundsätze guten Designs nicht zu kurz. In der bewährten, hälftigen Mischung aus Vorlesungen und Übungen lernen die Teilnehmer:

- Muster für das Design im Kleinen
- Trennung von Technik und Fachlichkeit bei grafischen Benutzeroberflächen
- technische Strukturierung des Anwendungskerns

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

- Anbindung des Anwendungskern an einen modernen Persistenzmanager
- Beherrschung von Problemen verteilter Anwendungen

Den Höhepunkt der School stellt die Projektanalyse dar. Eigens für diesen Tag eingeladene Chefdesigner von sd&m stellen ihr aktuelles Projekt vor und stehen danach bis zum Abend Rede und Antwort. Dabei diskutieren die Teilnehmer, wie die Standardarchitektur im Projekt umgesetzt und an welchen Stellen mit welcher Begründung davon abgewichen wurde.

4.3 Technische Chef-Designer (TCD) School

Während die Designer School sich an den angehenden technischen Berater wendet, besteht die Zielgruppe der TCD School aus erfahrenen Chef-Designern. Die Teilnehmer dieser School haben bereits in mindestens einem großen Projekt die Rolle des Chef-Designers übernommen. Die Anzahl der Teilnehmer ist auf 12-15 Personen beschränkt, um rege Diskussionen zwischen allen Teilnehmern zu fördern. Vorlesungen nehmen in dieser School nur noch einen sehr kleinen Raum ein. Im Vordergrund steht die Arbeit in Arbeitsgruppen. Zuerst wird in Gruppen jeweils die Architektur eines großen Software-Systems von sd&m analysiert. Hierbei geht es insbesondere darum, ob und wie die Konzepte der sd&m Standardarchitektur Quasar umgesetzt wurden.

In einer zweiten Gruppenarbeit werden Erfahrungen und Ideen rund um Themen der Konstruktion ausgetauscht, z.B. die Ergebnistypen der Konstruktion oder der Einfluss des Chef-Designers auf die Beschleunigung des Projekts. Die Ergebnisse dieser Gruppenarbeiten werden später bei sd&m konsolidiert und weiterverbreitet.

Die drei Tage TCD School dienen somit weniger dem Vermitteln von Wissen als dem Austausch von Erfahrungen und dem Reflektieren von Projektergebnissen.

4.4 Programmer School

An fünf Tagen motiviert die Programmer School ihre ca. 25 Teilnehmer zu begeistertem, zeitweise auch verzweifeltem Programmieren oft bis spät in die Nacht. Lernziel dieser School ist es, mit moderner Technologie große und langlebige Anwendungen zu bauen. Wir betrachten diese School im Folgenden etwas ausführlicher.

Als Beispiel verwenden wir in dieser School ein Telekom-Abrechnungssystem bestehend aus Kundenverwaltung, Tarifsystem und Abrechnungskomponente. Dieses System ist fachlich überschaubar und zugleich genügend komplex. Technisch betrachten wir nacheinander Architektur und Implementierung des Anwendungskerns, die Persistenz der verwalteten Daten und die grafische Benutzeroberfläche.

Eine geeignete Basis bildet die Programmiersprache Java, erweitert um eine Reihe technischer Komponenten und Entwicklungswerkzeuge, die zusammen eine professionell einsetzbare Entwicklungsumgebung bilden. Grundkenntnisse über Java werden vorausgesetzt oder ggf. vorher in einem eintägigen Crash-Kurs vermittelt. Die notwendige Hardware besteht aus einem Laptop für jeden Teilnehmer, die über

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

ein lokales Netzwerk mit dem Schulungsserver und über ein Modem mit dem Internet verbunden sind.

Die School beginnt mit der praktischen Umsetzung höherer objektorientierter Konzepte, insbesondere zur strikten Trennung von Zuständigkeiten im Programmcode. Ziel dieses ersten Schulungstages ist gutes Programmieren und Designen im Kleinen. Neben einem seichten Einstieg in die Entwicklungsumgebung lernen die Teilnehmer in Übungen, wie man Schnittstellen und Muster einsetzt, um Aufgaben klar zu trennen und notwendige Abhängigkeiten darzulegen.

Der zweite Tag konzentriert sich auf die Implementierung eines Anwendungskerns, also der Fachlichkeit eines betrieblichen Informationssystems, unter Einhaltung der Quasar-Standardarchitektur. Zwischen und nach den Vorlesungen vervollständigen die Teilnehmer einen lückenhaften Anwendungskern des Telekom-Abrechnungssystems.

Anders als in den ersten Übungen arbeiten die Teilnehmer hier zu zweit oder zu dritt zusammen. Um den Einstieg in die komponentenbasierte Softwarearchitektur zu erleichtern, finden diese Übungen noch ohne Anbindung von Datenbank und grafi- scher Benutzeroberfläche statt. Auskodierte Testfälle und eine Dummy-Implementierung der Datenhaltung geben den Teilnehmern schnell Rückmeldung über Erfolg oder Misserfolg ihrer Anstrengungen.

Der dritte Tag widmet sich der Verwaltung persistenter Objekte auf Basis einer relationalen Datenbank. Nach einer Einführung in den von sd&m entwickelten Persistenzmanager besteht die Aufgabe für die Teilnehmer darin, die Beschreibung persistenter Klassen für den Persistenzmanager zu vervollständigen und die Dummy-Implementierung der Datenhaltung durch eine auf den Persistenzmanager gestützte, voll funktionsfähige eigene Implementierung zu ersetzen. Dabei wird die Trennung von Datenbankanbindung und fachlichem Code im Anwendungskern mit Hilfe eines Persistenzmanagers vermittelt.

Gleich zwei Tage sind zum Entwickeln und Anbinden einer grafischen Benutzeroberfläche geplant. Dies entspricht durchaus der Verteilung des Implementierungsaufwands realer Projekte. Als Basis dient hier eine Erweiterung des Web-GUI-Frameworks Struts [5], womit als zusätzliche Sprachen HTML bzw. JSP und als zusätzliche Infrastruktur ein Web-Server einzuführen sind.

Im Anschluss an die letzte Übung hat jede Gruppe Gelegenheit, einen interessanten Aspekt ihrer Arbeit im Plenum zu präsentieren. An den vorherigen Tagen erfolgt das Feedback durch die vorgegebenen oder selbst zu kodierenden Testfälle und durch intensive Betreuung durch die drei Tutoren.

Kritik an der Programmer School richtet sich besonders an die Übungen. In vielen Übungen werden Lücken im Code gefüllt. Diese Übungen werden oft durch Kopieren von Code-Segmenten gelöst, ohne dass dabei der Zusammenhang verstanden wird. Für die Übungen planen wir deshalb eine Umstellung von Code-Gerüsten zum kompletten Kodieren einfacher Anwendungsteile. Eine ausführliche Schritt-für-Schritt-Anleitung soll die notwendigen Zusammenhänge vermitteln. Als Ausgangspunkt verwenden wir die Referenzimplementierung einer Anwendungskomponente.

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

Die Programmier-Übungen zu Beginn der School werden durch Kodierarbeiten am Konfigurationscode der Anwendung ersetzt, so dass insgesamt mehr Zeit für die Anwendungsprogrammierung zur Verfügung steht. Durch Hinweise auf mögliche Erweiterung werden wir Teilnehmern mit höherem Einstiegsniveau gerecht.

In der School verwenden wir eine reale, komplexe Entwicklungsumgebung:

- SUN Java JDK SE 1.3.1 oder 1.4.1 [4]
- IntelliJ IDEA 2.6 als grafisches Entwicklungswerkzeug [5]
- CVS Versionsverwaltung [6]
- JUnit Test Framework [7]
- Ant Build-Werkzeug [8]
- MySQL Datenbank [9]
- sd&m Quasar Persistence + Modeler's Workbench
- Jakarta Tomcat Web-Server [10]
- Jakarta Struts Web Application Framework [11]
- Mozilla HTML Editor

Bis auf IntelliJ ist die eingesetzte Software frei erhältlich. Alle Komponenten werden mit Ausnahme der Datenbank in professionellen Projekten eingesetzt, so dass die Teilnehmer auch in dieser Hinsicht auf zukünftige Projekte vorbereitet werden.

Die resultierende Vielzahl eingesetzter Technologien ist natürlich nicht ohne Nachteile. Die Vorbereitung einer solchen School birgt einen hohen, weil fehlerträchtigen Konfigurationsaufwand sowie erheblichen Aufwand für die Softwareverteilung. Durch die zahlreichen neuen Werkzeuge in kurzer Zeit sind viele Teilnehmer überfordert. Eine Besserung erhoffen wir uns durch ein Umschwenken in den Übungen von der komplexen Web- auf eine reine Java-Benutzeroberfläche.

4.5 Test School

Die Test School umfasst in vier Schulungstagen den ganzen Bereich des Testens von der Testplanung und -konzeption anhand der Ergebnisse früher Phasen bis zur Durchführung und Analyse der Ergebnisse (Fehler). Lehrziele sind die Schulung und Verbesserung der Mitarbeiter in Testplanung, Testverfahren, Teststufen, systematischer Testfallermittlung und Handhabung von Testwerkzeugen.

Der Ablauf orientiert sich dabei an einem Projekt. Am Anfang steht die Planung des Tests, am Ende der School der Systemtest und das Fehlermanagement. Der letzte Tag ist dem eigenen Projekt gewidmet, in dem das Gelernte direkt auf die Wirklichkeit angewendet und überprüft werden kann.

Die Test School richtet sich an alle Software-Ingenieure, (Chef-)Designer und Projektleiter, denn alle müssen sich im Rahmen der Projektdurchführung oder Angebotserstellung mit dem Thema Testen auseinander setzen.

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

5 Reihenfolge der Schools

Wann welche School belegt wird, ist hauptsächlich von den Erfahrungen eines Mitarbeiters und dessen Bedarf abhängig. Für die Specification School ist die Erfahrung in der Spezifikation in mindestens einem Projekt Voraussetzung. Sie soll belegt werden, wenn in näherer Zukunft Spezifikationsaufgaben für diesen Mitarbeiter anstehen. Analoges gilt für die Designer School. Für die TCD School müssen die Teilnehmer wie oben beschrieben sehr viel mehr praktische Erfahrung mitbringen und die Designer School bereits besucht haben. Die Programmer School richtet sich dagegen eher an Einsteiger mit wenig Programmiererfahrung. Wie erwähnt ist die Test School für Mitarbeiter mit unterschiedlichstem Vorwissen interessant.

Die Teilnahme an den Schulungen folgt somit keiner festen Reihenfolge, sondern wird individuell gestaltet. Gewöhnlich wird die Programmer School als erste School besucht. Danach erfolgt die Auswahl aus Designer, Specification und Test School nach Weiterbildungswunsch und -bedarf des Mitarbeiter. In der Regeln nehmen Mitarbeiter alle 1-2 Jahre an einer School teil.

6 Allgemeine Erfahrungen mit den Schulungen

Am Ende jeder School stellen sich die Veranstalter der Kritik aller Teilnehmer. Hier erhalten wir in der Regel viel Lob von zufriedenen Mitarbeitern. Natürlich gibt es daneben aber auch immer Kritik und Anregungen zur weiteren Verbesserung der School, die hier zusammengefasst werden.

6.1 Erfahrung mit dem Ablauf

Positive Resonanz erhält die Gestaltung aus Vorlesungen und Übungen und die Gruppenarbeit.

Die Übungen werden von Teilnehmern und Trainern als Mittel zur Vermittlung praktischen Wissens sehr geschätzt. Problematisch ist die Bemessung der Übungszeit. Oft können Übungsaufgaben aus Zeitmangel nicht zur Zufriedenheit der Teilnehmer fertiggestellt werden. Auch wünschen sich die Teilnehmer Zeit, um nach einem Feedback ihr Ergebnis korrigieren zu können. In der Specification School haben wir bereits mit einer Übung in zwei Teilen gute Erfahrungen gesammelt. Dabei erhalten die Gruppen nach einem ersten Teil der Übung ein Feedback und haben danach im zweiten Teil der Übung Gelegenheit, dieses Feedback einzuarbeiten.

Der Umgang mit den teilweise sehr komplexen Übungsaufgaben ist unterschiedlich. Während ein Großteil der Teilnehmer die Herausforderung dieser schweren Aufgaben schätzt, fühlen sich andere Teilnehmer überfordert und demotiviert, wenn sie die Aufgabe nicht in der gegebenen Zeit lösen können.

Neben den Übungen werden die Projektanalysen, in denen die Schulungsinhalte an realen Projekten reflektiert werden, als besonders wertvoll empfunden. Je nach

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

Thema der School wird hier die Art der Spezifikation, das Design oder die Teststrategie unter die Lupe genommen. Bei einer solchen Analyse werden den Teilnehmern nicht nur anhand eines Beispiels die Inhalte der School nochmals vor Augen geführt. Insbesondere trainieren sie dabei auch die Beurteilung von Projekten anhand weniger Unterlagen in kurzer Zeit – eine Fähigkeit, die in der weiteren Arbeit bei Reviews und in der Angebotserstellung von hohem Wert ist.

6.2 Erfahrungen mit der Gruppenarbeit

Die Arbeit in Gruppen ist ein wichtiger Erfolgsfaktor der Schools. Durch die eher kleinen Gruppen trägt jeder merklich zum Ergebnis bei und feiert seine privaten Erfolge. Die Ergebnisse werden abwechselnd präsentiert, so dass jede Gruppe mindestens einmal ihr Ergebnis vor den kritischen Augen der Kollegen verteidigen muss. Teilnehmer wie Trainer sind sich einig, dass sie in diesen Feedback-Runden am meisten lernen, weil die eigenen Lösungen anhand der vorgestellten Lösung hinterfragt werden.

Für die Zufriedenheit mit der Gruppenarbeit ist besonders die Gruppengröße und die Betreuung wichtig. Wir haben auch mit einer School von über 60 Teilnehmern Erfahrung gesammelt. Hier wurden 15 Gruppen gebildet, die von 9 Trainern betreut wurden. So stand stets ein Ansprechpartner zur Verfügung. Entscheidend für den Erfolg war hier, dass für das Vorstellen der Ergebnisse immer drei Gruppen zusammengenommen wurden, so dass jede Gruppe ihr Ergebnis vorstellen und diskutieren konnte.

6.3 Niveau und Projektnutzen

Wie am Feedback zu Vorlesungen und Übungen deutlich wird, ist es schwierig, das richtige Niveau für die Schulungen zu treffen. Das Niveau der Einstiegsvorlesungen wird in allen Schools von vielen Teilnehmern als zu niedrig empfunden. Dies liegt zum Teil an einem sehr unterschiedlichen Erfahrungshintergrund der Teilnehmer. Hierin äußert sich auch ein Problem im Anmeldungsprozess für die Schulungen. Einerseits sind die Lehrinhalte einer School den Teilnehmern bei der Anmeldung nicht transparent genug. Andererseits werden auch die geforderten Voraussetzungen nicht klar genug dargestellt. Hinzu kommt, dass Anmeldungen selten abgelehnt werden. Hier fehlt es an einem Auswahlprozess, der Teilnehmer nicht nur nach deren Schulungswünschen und freien Plätzen, sondern auch nach den Erfahrungen der Teilnehmer den Schulungen zuordnet.

Ein weiterer Ansatz zur Verbesserung besteht in einer Vergrößerung des Angebots mit differenzierten Inhalten. So wurde die TCD School erst im Jahr 2002 in das Programm aufgenommen, um erfahrenen Mitarbeitern eine angemessene Fortbildung zu bieten. In 2003 wird dieses Angebot durch die Designer Refresher School erweitert, die zwischen Designer und TCD School einzuordnen ist.

Der zukünftige Nutzen der Schulungsinhalte in Projekten wird von den Teilnehmern allgemein als gut eingestuft. Dies hängt jedoch wieder stark von der Art der

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

Projekte ab, in der die Teilnehmer mitarbeiten. Die Inhalte von Specification School und Test School sind in jedem Projekt anzuwenden, das die jeweilige Phase durchläuft. Die Inhalte der Programmer, Designer und TCD School sind dagegen stark auf objektorientiertes Vorgehen ausgerichtet. Die Übertragung in beispielsweise Hostprojekte fällt einigen Mitarbeitern schwer.

Während wir bisher versucht haben, die Schulungen unabhängig von einer bestimmten Technologie zu halten, erkennen wir zunehmenden Bedarf nach Schulung in konkreten Technologien, um Mitarbeiter auf die Projekte besser vorzubereiten.

7 Fazit

Mit den Schools bietet sd&m seinen Mitarbeitern eine Reihe interessanter Schulungen an, die alle Phasen der Software-Entwicklung abdecken. Die wichtigsten Erfolgsfaktoren dieser Schulungen sind ein Wechsel von Vorlesungen und Übungen, das Arbeiten in kleinen Gruppen, die Behandlung realistischer Beispiele und die Analyse eigener Projekte. Die Schools sind ein erfolgreiches Modell der Schulung, in dem auch der ausgebildete Software-Ingenieur Nutzen für seine Arbeit erlangt.

Um die Schulungen interessant zu halten, werden die Schulungsinhalte jährlich entsprechend der sich weiterentwickelnden Standardarchitektur und -methodik angepasst. Mit einer zunehmend technischen Ausrichtung auf Komponenten und Werkzeuge begegnet sd&m dem wachsenden Marktdruck und den komplexer werdenden Technologien. Um das Verständnis größerer Zusammenhänge zu verbessern, werden die Aufgaben weiter in Richtung Design bzw. Implementierung kompletter Teilsysteme umgestaltet. Durch zusätzliche Schools wird das Angebot inhaltlich stärker auf den Bedarf der Teilnehmer ausgerichtet.

Literatur

1. Siedersleben, J. (Hrsg.): Softwaretechnik. Hanser, München, 2002, 2. Auflage
2. Siedersleben, J. (Hrsg.): Quasar: Die sd&m Standard-Architektur. sd&m Research, 2002
3. Stutz, C.; Siedersleben, J.; Kretschmer, D.; Krug, W.: Analysis beyond UML. Proceedings of IEEE Requirements Engineering Conference 2002
4. http://java.sun.com Java
5. http://www.intellij.com IntelliJ IDEA Entwicklungsumgebung
6. http://www.cvshome.org CVS Versionsverwaltungssystem
7. http://www.junit.org A regression testing framework for Java
8. http://jakarta.apache.org/ant Java-basiertes Build-Werkzeug
9. http://www.mysql.de Relationales DBMS
10. http://jakarta.apache.org/tomcat/index.html Web-Server bzw. Servlet-Container
11. http://jakarta.apache.org/struts/index.html A framework for building web applications