Extreme Programming in der Informatik-Lehre — Ein Erfahrungsbericht

Holger Mügge, Daniel Speicher, Günter Kniesel
Institut für Informatik III, Universität Bonn
Römerstr. 164, 53117 Bonn
{muegge,speicher,kniesel}@cs.uni-bonn.de

Abstract: eXtreme Programming (XP) ist eine Symbiose verschiedener Techniken, die erst durch ihr Zusammenwirken ihre besondere Qualität agiler Softwareentwicklung aufweisen. Sie lassen sich theoretisch leicht vermitteln – ihr Wert erschliesst sich aber nur durch eigene Praxis. Soll XP im Rahmen universitärer Lehre adäquat vermittelt werden, müssen die Anforderungen industrieller Praxis in geeigneter Weise simuliert werden. In diesem Bericht fassen wir unsere Erfahrungen aus der Veranstaltung von fünf XP-Praktika im Informatik-Hauptstudium zusammen. Wir zeigen Adaptionen der XP-Methodik auf und stellen eine Reihe von Techniken vor, mit denen der Spagat zwischen universitären Lernzielen und Projekterfolg bewältigt werden kann.

1 Einleitung

Seit Herbst 2002 hat die Softwaretechnologie-Arbeitsgruppe am Institut für Informatik III der Universität Bonn im Rahmen der Exzellenzstudienangebote des B-IT[1] unter der Leitung von Prof. Dr. Armin B. Cremers fünf XP-Praktika als Blockveranstaltungen von jeweils vier Wochen Dauer und Arbeitstagen zu 8 Stunden durchgeführt.

In diesem Erfahrungsbericht beschreiben wir die verschiedenen Rollen der Veranstalter und mögliche Konflikte. Wir diskutieren Möglichkeiten, den fehlenden industriellen Rahmen zu simulieren. Danach stellen wir Techniken zum Umgang mit den fehlenden Voraussetzungen der Teilnehmer vor und zeigen einige Anpassungen des Prozesses an die Lehrsituation sowie Möglichkeiten zur Weiterentwicklung der Software nach Praktikumsende. Ein Fazit mit Ausblick auf künftigen Vorhaben schließt den Bericht ab.

2 Der Grundkonflikt: Lehren versus Produzieren

Prozess und Reflektion: In der doppelten Zielsetzung eines XP-Praktikums als *Lehrveranstaltung* und als *Projekt* liegt ein grundlegendes Konfliktpotential. Die beiden Ebenen *Projekt* und *Lehrveranstaltung* sollten daher für die Teilnehmer zwar zeitlich eng verzahnt

[1] Bonn-Aachen International Center for Information Technology (B-IT), http://www.b-it-center.de/

aber gut unterscheidbar sein. So bieten z. B. Probleme im Planning Game die Möglichkeit, den Ablauf zu unterbrechen, um Hintergründe und Zusammenhänge zu erläutern.

Rollen: Beck beschreibt in [Be00] die Rollen im Rahmen des XP-Prozesses. Explizit umgesetzt haben wir die Rollen *Kunde* (customer), *Teamleiter* (coach), *Experte* (consultant) und *Teilnehmer* (programmer). Zusätzlich ist uns die Rolle des *XP-Lehrers* wichtig. Er unterbricht zu geeigneten Zeitpunkten den Projektablauf, reflektiert mit den Teilnehmern die Anwendung der XP-Techniken und vergleicht sie mit anderen Methoden.

Rollenkonflikte: Wenn manche Rollen von der selben Person übernommen werden, können Konflikte auftreten. Die folgende Tabelle zeigt welche Rollen die Veranstalter in unseren fünf Praktika übernommen haben.

Kunde	Experte	Teamleiter	XP-Lehrer	Beobachtete Konflikte
A	A, B	A	A	Kunde = Teamleiter: Nicht vereinbar
A	A, B, C	C	C	Experte = Teamleiter: Problematisch
A	A, B	C	C	Teamleiter = XP-Lehrer: Sehr anstrengend
A	B	B, C	C, B	Experte = XP-Lehrer/Teamleiter: Ungünstig
A	B	C	D	Alle Rollen getrennt: Personeller Idealfall

Die Rollen *Kunde und Teamleiter* sind aus unserer Erfahrung unvereinbar, sonst würde die essentielle Trennung der Entscheidungsbereiche von Kunden und Entwicklern aufgehoben. *Experte und Teamleiter* sind schwer zu kombinieren. Der Experte berät, überlässt aber die Entscheidung dem Team. Der Teamleiter dagegen besitzt Entscheidungsautorität. Die Kombination von *Teamleiter und XP-Lehrer* stellt sehr hohe Ansprüche an den Akteur. Während er den Prozess moderiert, muss er ständig darauf achten, wann er ihn für didaktische Einheiten unterbrechen sollte.

3 Einen Rahmen für XP schaffen

Veranstaltungsform und Dauer: Mit Blockveranstaltungen haben wir gute Erfahrungen gemacht, allerdings sind vier Wochen knapp bemessen. Künftig veranstalten wir ein sechswöchiges Praktikum mit integriertem Seminar. Außerdem läuft derzeit eine Semesterbegleitende Veranstaltung. Hier zeigen sich erhebliche Kommunikationsprobleme.

Teilnehmerzahl: Unsere Praktika hatten vier bis acht Teilnehmer. Diese Gruppengröße hat sich bewährt. Auch mit nur vier Teilnehmern konnte gut gearbeitet werden. Mehr als zehn Teilnehmer halten wir wegen des Kommunikationsaufwands für problematisch.

Geeigneter Raum: Ein eigener Arbeitsraum ist wichtig. Zur Ausstattung gehören: Arbeitsplätze (nicht mehr als Paare), Tafel, Pinnwände für Task-Cards und Notizen, Regal für Bücher, Material und Essen, Besprechungstisch sowie Platz für Stand-Up-Meetings.

Anwendungsdomäne: In unseren Praktika stammen die Aufgabenstellungen bisher immer aus Forschungsprojekten des eigenen Instituts. Die Kunden sind also selbst Software-Entwickler, so dass Kunde und Entwickler die gleiche Sprache sprechen. Manche realistischen und lehrreichen Missverständnisse traten daher leider nicht auf.

Vorbereitung: Es hat sich bewährt, vor Praktikumsbeginn Übungen zu Themen wie Umgang mit der Entwicklungsumgebung, Test-First und Refactoring zu veranstalten und gegebenenfalls die spärlichen Programmierkenntnisse einiger Teilnehmer zu verbessern.

4 XP-Techniken agil lehren

Vermitteln der XP-Regeln: Zu Beginn lassen wir die Teilnehmer "10 Gebote" zum eXtreme Programming aufstellen. Jeder steuert mindestens eine Regel bei, die Vorschläge werden diskutiert, auf einem Plakat festgehalten und gut sichtbar aufgehängt. So lassen sich in ca. einer Stunde die wichtigsten XP-Regeln vermitteln.

Fragen sind willkommen!	Ohne Partner kein Coding!
Wechsle Deinen Partner oft!	Code ohne Test wird gelöscht!
Halte unsere Code-Konventionen ein!	Sei mutig und refaktorisiere gnadenlos!
Tue das einfachste, das funktionieren wird!	Helfe anderen, wie auch Dir geholfen wird!
Sei nicht egoistisch, der Code gehört dem Team!	Befolge Regeln nicht blind, auch diese nicht!

Anleitung zu systematischer Problemlösung: Als Anleitung zu selbstständiger Arbeit haben wir mit den Teilnehmern eine Systematik zur Problemlösung erarbeitet. Diese enthält Namen relevanter Newsgroups, Zugriff auf unser Wiki, Beispielsammlung im CVS etc.

Verantwortung: Während des Praktikums fallen wiederkehrende Aufgaben an, wie z. B. tägliche CVS-Tags einrichten, Tafel wischen etc. Es hat sich bewährt, für jede Aufgabe *einen* Verantwortlichen zu benennen.

Expertenliste: Damit Kommunikation und gegenseitige Hilfe in der Kürze des Praktikums schnell in Gang kommen, machen wir einen Aushang über das Wissensprofil aller.

Pair Programming: Mit Pair Programming haben wir durchgängig gute Erfahrungen gemacht. Bei sehr unterschiedlich starken Partnern kann der "Langsamere" tippen, damit er nicht abgehängt wird. Innerhalb eines Paares sollte der eine programmieren, während der andere auf Flüchtigkeitsfehler achtet aber auch die große Linie verfolgt.

Gebändigter Spieltrieb: Wie Schneider und Johnston in [SJ03] gehen auch wir davon aus, dass Studenten gewohnt sind, sich als Kunde zu fühlen und glauben, selbst am besten zu wissen was wichtig ist. Müller und Tichy berichten in [MT01], dass die Teilnehmer übermäßig vorausschauend entwerfen und zu "Featuritis" neigen.

Wir konnten solch überbordende Aktivität durch hohen Arbeitsdruck und anspruchsvolle Aufgaben meist in geeignete Bahnen lenken. Dazu ist es wichtig, dass der Kunde präsent ist und Autorität ausstrahlt (vgl. auch [NA03] und [GKSY04]). Weiteren Ansporn kann auch ein Besuch des Institutsleiters geben.

Projekt-Metapher: Diverse Erfahrungsberichte nennen die Metapher als eine schwierig umsetzbare Technik, darunter [GKSY04] und [NMMB04]. Wir haben statt dessen eine Projekt-spezifische Sprache mit dem Kunden im Team definiert und als Glossar im Wiki gepflegt. So ließen sich Missverständnisse reduzieren.

Abschlusspräsentation: Wir legen großen Wert auf die Abnahme jeder Iteration. In besonderem Maße gilt das für die abschließende Iteration. Nach Möglichkeit wird dazu z. B. der Institutsleiter eingeladen. Eine gut geplante und geprobte Präsentation, zu der alle beitragen, lohnt sich sowohl für die Veranstalter als auch für die Teilnehmer.

5 Den XP-Prozess steuern

Schätzungen und Geschwindigkeit: In der XP-Gemeinde wird das Schätzen von Stories im Rahmen des Planning Games kontrovers diskutiert. Eine Variante benutzt *Konkrete Zeiteinheiten* und macht *absolute Zeitschätzungen* für die Stories, also z. B.: "Story 14 wird ca. 1,5 Tage dauern". Bei einer anderen Variante werden abstrakte Schwierigkeitseinheiten ("Gummibärchen") verwendet und *Schätzungen relativ zu ausgewählten zuvor bearbeiteten Stories* abgegeben, also etwa: "Story 14 wird ca. doppelt so lange wie Story 3 dauern".

Konkrete Zeiteinheiten zu benutzen ist verständlich und liefert einfach interpretierbare Ergebnisse. Vielen Teilnehmern fehlt aber die Erfahrung zum Schätzen. Daher schliessen sie sich häufig selbstbewussteren Teilnehmern an. So werden die Schätzungen verfälscht und die Gruppe kann auseinanderbrechen in "gute" aktive und vermeintlich schlechte passive Teilnehmer. Wir haben mit *relativem Schätzen* gute Erfahrungen gemacht. Auch unterschiedlich erfahrene Programmierer können sich meist einigen, eine Aufgabe als "schwierig" oder "leicht" zu bewerten. Dieses Verfahren ist allerdings erläuterungsbedürftig.

Zeitliche Skalierung: Wir empfehlen, vier bis sechs Iterationen durchzuführen. Die erste und letzte sind von der besonderen Situation (Orientierung bzw. Endspurt) geprägt, so dass nur die "mittleren" Iterationen eine gewisse Routine aufkommen lassen. Iterationen und Stories sind im Praktikum deutlich kürzer, als sonst. Sie können aber problemlos skaliert werden. Für Stories und Tasks ergeben sich von allein praktikable Größen. Die mittlere Storydauer variierte je nach Aufgabe und Erfahrung des Teams zwischen 0,5 und 3 Tagen.

Selbstzerlegende Stories: Die Aufteilung komplexer Stories in Tasks ist oft zeitaufwändig und lädt zu Detaildiskussionen ein, die besser von kleinen Teams oder Paaren gelöst werden als vom ganzen Team. Deshalb gehen wir im Laufe des Praktikums dazu über, nur noch Stories im Team zu definieren. Die Aufteilung der Story in Tasks wird dann von einem Paar übernommen. Deren Ergebnis wird sofort dem ganzen Team verfügbar gemacht.

6 Weiterentwicklung nach Ende des Praktikums

Dokumentation: Wir haben die Dokumentation von Quelltext auf die Schnittstellen (JavaDoc) beschränkt. Lesbare Tests, gut strukturierte Implementierungen und klare Bezeichner ersetzen Kommentare in Methodenrümpfen. Häufige Kontrolle der Qualität von Code und Testdokumentation können wir sehr empfehlen.

Rationale Management: Die Technik *Travel Light* lässt sich gut mit einer leichtgewichti-

gen Variante von Rationale Management verbinden. Wie auch Andreas Rüping in [Rü03] vorschlägt, benutzen wir ein Wiki zur Dokumentation unserer Entscheidungen.

Personelle Brücke: Das Wissen der Teilnehmer über die Software lässt sich durch Dokumentation und Rationale Management nicht hinreichend erfassen, um z.B. die Übergabe an ein anderes Team zu ermöglichen. Wir setzen deshalb einen "Experten" (z. B. ein Diplomand) ein, der über das Praktikum hinaus an dem Projekt arbeitet.

7 Fazit und Ausblick

Von den Teilnehmern unserer Praktika haben wir sehr positive Rückmeldungen erhalten. Wir möchten daher auch andere Dozenten ermuntern, XP-Praktika anzubieten. Damit sich sowohl Lehr- als auch Projekterfolg einstellt, sind allerdings einige Hürden zu nehmen. Neben dem Grundkonflikt zwischen Lehren und Produzieren zählen dazu: Kurze Projektlaufzeit, mangelnde Vorkenntnisse der Teilnehmer und grosser organisatorischer Aufwand.

Um die Ebenen *Lehrveranstaltung* und *Projekt* klar zu trennen, schlagen wir die neue Rolle *XP-Lehrer* vor und plädieren dafür, die Rollen *Kunde* und *Teamleiter* nicht mit derselben Person zu besetzen. Als Veranstaltungsform halten wir Blockveranstaltungen von minimal vier Wochen für unerlässlich und empfehlen eine Teilnehmerzahl von maximal zehn. Wir stellen einige Techniken vor, um das Vermitteln und den Ablauf des Prozesses zu unterstützen und die Weiterentwicklung der Software nach Praktikumsende zu ermöglichen.

Künftig veranstalten wir ein sechswöchiges Praktikum mit integriertem Seminar. Außerdem wollen wir Partner aus der Industrie als Kunde, Experte oder Teilnehmer einbeziehen.

Literatur

[Be00] Beck, K.: *Extreme Programming*. Addison-Wesley. 2000.

[GKSY04] Goldman, A., Kon, F., Silva, P. J. S., und Yoder, J.: Being extreme in the classroom: Experiences teaching xp. Technical Report RT-MAC-2004-01. Department of Computer Science, University of São Paulo. 2004.

[MT01] Müller, M. M. und Tichy, W. F.: Case study: Extreme programming in a university environment. In: *International Conference on Software Engineering (ICSE)*. S. 537–544. Toronto, Canada. May 2001.

[NA03] Noll, J. und Atkinson, D. C.: Comparing extreme programming to traditional development for student projects: A case study. In: *4th International Conference on eXtreme Programming and Agile Processes in Software Engineering*. May 2003.

[NMMB04] Noble, J., Marshall, S., Marshall, S., und Biddle, R.: Less extreme programming. In: *Sixth Australasian Computing Education Conference*. S. 217–226. 2004.

[Rü03] Rüping, A.: *Agile Documentation*. Wiley. 2003.

[SJ03] Schneider, J.-G. und Johnston, L.: Extreme programming — helpful or harmful in educating undergraduates? *Journal of Systems of Software*. December 2003.