Horst Lichter, Martin Glinz (Hrsg.) (2001): Software Engineering im Unterricht der Hochschulen
SEUH 7 – Zürich, dpunkt.verlag, Heidelberg.

# Die Rolle der Reflexion in Softwarepraktika

*Claus Lewerentz, Heinrich Rust*[*]
Lehrstuhl Software-Systemtechnik, BTU Cottbus

## Zusammenfassung

*Softwarepraktika dienen in vielen Informatikstudiengängen zur Einübung von Softwaretechnikfertigkeiten. Weil die spezifischen Techniken in der Softwaretechnik rasch veralten, ist es wichtig, dass die Studierenden lernen, ihre eigene Arbeitsweise immer wieder zu verbessern und neu vorgeschlagene spezifische Techniken dahingehend zu bewerten, ob sie in ihrer Arbeitssituation hilfreich sind. Es wird eine zyklische Organisationsform für ein Softwarepraktikum vorgestellt, in dem die Studierenden immer wieder zur Selbstbeobachtung und zur Reflexion der von ihnen gemachten Erfahrungen angehalten werden. Diese wiederholten expliziten Reflexionen münden in Konsequenzen, die im jeweils folgenden Zyklus in der Praxis erprobt werden und am Ende des Zyklusses sowie am Ende des Gesamtprojektes bewertet werden können.*

## 1 Aufgaben und Formen eines Softwarepraktikums

Softwarepraktika sind in unterschiedlicher Form und Dauer Bestandteil verschiedener Studiengänge mit Softwaretechnik-Anteilen. Solchen Softwarepraktika liegen unter anderem folgende Ausbildungsanliegen zugrunde: (1) Die typischerweise zuvor in Vorlesungen gelernten Einzeltechniken werden im Zusammenhang angewendet. (2) Sogenannte „soft skills" werden eingeübt. (3) Studierende erfahren, dass es in der softwaretechnischen Praxis nicht ohne Dilemmata und, in der Folge, ohne Kompromisse abgeht. (4) Der Transfer von softwaretechnischem Wissen in die Praxis wird erleichtert, wenn in man einen Zwischenschritt einschiebt zwischen die (oft bloß rezipierte und in ihrer Relevanz nicht praktisch erfahrene) Theorie und die (oft nicht systematisch reflektierte) Praxis.

An der Brandenburgischen Technischen Universität in Cottbus muß im Grundstudium der Studiengänge Informatik bzw. Informations- und Medientechnik ein Softwarepraktikum absolviert werden. Dieser Artikel beschreibt, wie wir in Cottbus diese Praktika gestalten.

[*] BTU Cottbus, Postfach 10 13 44, D-03013 Cottbus, Deutschland;
Tel.: +49 (355) 69-38 81, Fax: -38 10; (lewerentz, rust)@informatik.tu-cottbus.de

Horst Lichter, Martin Glinz (Hrsg.) (2001): Software Engineering im Unterricht der Hochschulen
SEUH 7 – Zürich, dpunkt.verlag, Heidelberg.



Unser Ansatz beruht auf einer einfachen Kernidee, die wichtige Anteile an der Verantwortung für den Lernerfolg den Studierenden zuweist: Es ist die Aufgabe der Dozenten, für die Studierenden Gelegenheiten zu generieren, Erfahrungen zu machen; und es ist die Aufgabe der Studierenden, ihre Erfahrungen explizit zu machen, sie zu interpretieren und Konsequenzen daraus zu ziehen, oder kurz: sie zu reflektieren. Reflexion ist ein wesentliches Element in unserem Ansatz der Gestaltung eines Softwarepraktikums im Grundstudium, und dieses Element sehen wir auch als ein wichtiges Hilfsmittel für das Lernen in der späteren beruflichen Praxis.

Wir sind im Laufe der Zeit zu einem zunehmend detaillierten Vorgehensmodell gekommen, das unter anderem folgende Elemente enthält: (1) Es wird verlangt, dass das Projekt eine zyklische Struktur hat, wobei ein wasserfallartiger Prozess drei bis vier Male durchlaufen wird. (2) Wir verlangen von den Studierenden detaillierte Protokolle der Aufwände für ihre praktikumsspezifischen Tätigkeiten, um Planungen und wirkliche Aufwände vergleichen zu können und Konsequenzen für kommende Planungsarbeiten zu ziehen. (3) Wir verlangen zum Abschluss jedes Teilprojekts und zum Abschluss des Gesamtprojekts Reflexionspapiere. Diese enthalten zum einen Vergleiche von geplanten und tatsächlichen Aufwänden für das betrachtete Projekt und zum anderen persönliche Beurteilungen des Ablaufes des reflektierten Teilprojektes, wobei auch qualitative Einschätzungen eingehen dürfen. Wir erwarten für die verschiedenen reflektierten Punkte die Ableitung von Verhaltenskonsequenzen für die Folgeprojekte. (4) Jede Gruppe präsentiert ihrem Betreuer wöchentlich ihren Projektstand. Jedes dieser vier Elemente unterstützt die Reflexion.

## 2 Aufbau des Cottbuser Softwarepraktikums

Der „Personal Software Process" und der „Team Software Process" (PSP/TSP, [3,4]), und geringerem Maße auch der Ansatz des „Extreme Programming" (XP [1,2]), der Ansatz von Schön zum reflektierten Handeln [6,7,8] und Paschs Ideen zum Softwareentwurf im Team [5] haben uns bei der Ausgestaltung des Cottbuser Softwarepraktikums inspiriert. Schließlich sind neben den schon angedeuteten Erfahrungen in früheren Durchläufen des Praktikums Erfahrungen aus der industriellen Praxis eingeflossen: Hier werden neue Mitarbeiter nicht selten in ihr erstes Projekt geworfen wie in kaltes Wasser, unterstützt allein durch den Ratschlag: Schwimme! Wir wollen unseren Studierenden Fähigkeiten mitgeben, mit einer solchen Situation so produktiv umzugehen wie es eben möglich ist. Sie sollen ihre Arbeit so organisieren können, dass die Auswirkungen von Fehlern begrenzt werden und aus früheren Fehlern Konsequenzen für die späteren Projektphasen gezogen werden können. Ein zyklischer Projektablauf mit expliziten Reflexionselementen ist hierfür der angemessenste Rahmen.

Horst Lichter, Martin Glinz (Hrsg.) (2001): Software Engineering im Unterricht der Hochschulen
SEUH 7 – Zürich, dpunkt.verlag, Heidelberg.



Die für unseren Ansatz spezifischen Lernziele sind:

- Eigene Erfahrungen explizieren, interpretieren und für Folgetätigkeiten nutzen: Die eigenen Erfahrungen sind deshalb in der Softwaretechnik besonders wichtig, weil sie nicht so schnell wie das technische Lehrbuchwissen veralten – zumindest solange die Studierenden bereit sind, zuvor gemachte Erfahrungen auch in Frage zu stellen und neue Erfahrungen zu akzeptieren. Im Softwarepraktikum versuchen wir, den Studierenden den Wert ihrer eigenen (wenn auch im Lehrumfeld nur beschränkten) Erfahrungen deutlich zu machen. Wir versuchen dies, indem wir verlangen, dass die Praktikumsteilnehmer ihre Erfahrungen zum Teil explizit und damit für andere nachvollziehbar machen, dass sie sie interpretieren, und dass sie Konsequenzen daraus ziehen.
- Ein Projekt lernfreundlich organisieren: Wir bemühen uns, im Softwarepraktikum das Schlagwort „das Lernen lernen" mit einem präzisen Sinn zu versehen. Wir vermitteln den Praktikumsteilnehmern, welchen Wert es hat, wenn sie ihre Arbeitsprozesse so organisieren, dass sie in früheren Phasen Erfahrungen machen und diese später anwenden können. So versuchen wir, sie mit der Fähigkeit auszustatten, den eigenen Arbeitsprozess lernfreundlich zu organisieren.

Um anschaulich zu machen, mit Hilfe welcher Art von Aufgabenstellungen wir die angedeuteten Ziele zu erreichen versuchen, skizzieren wir zwei Beispiele, die wir im Sommersemester 2000 verwendet haben. Die erste Aufgabe betrifft die Entwicklung eines Klassenhierarchie-Browsers, wobei Symboltabellendaten aus einer Datenbank entnommen und mittels einer grafischen Benutzeroberfläche dargestellt werden sollten. Sie wurde von drei Gruppen mit je vier Personen bearbeitet. Die Aufwände bewegten sich insgesamt im Bereich von ca. 430 bis 480 Stunden, pro Person von etwa 107 bis 120 Stunden. Die zweite Aufgabenstellung betrifft die Programmierung eines kleinen mobilen Roboters, der mit einer Anzahl von optischen Abstandssensoren ausgestattet ist. Es ist ein Labyrinth zu vermessen, es sind Hindernisse in einem abgegrenzten Gebiet zu lokalisieren, und es ist ein gefahrener Weg zurückzuverfolgen. Diese Aufgabe hat sich auch in anderen Jahren bewährt; sie wurde zum wiederholten Male vergeben. Die Aufwände bewegten sich hier von 352 Stunden bei drei Personen bis 889 Stunden bei vier Personen, also von ca. 117 bis ca. 222 Stunden pro Person. Die letzte Zahl ist allerdings ungewöhnlich groß.

Insgesamt scheint ein Aufwand von etwa acht bis zehn Semesterwochenstunden für das Softwarepraktikum typisch zu sein. Dieser Umfang ist auch für diese Veranstaltung vorgegeben.

Wir geben den Studierenden vor, dass sie im Softwarepraktikum einen zyklischen Entwicklungsprozess benutzen müssen. Wir verlangen, dass diese einzelnen Teilprojekte möglichst vollständig sind, dass sie also alle Tätigkeiten enthalten, die in einem Wasserfallmodell vorkommen.

Die in den Teilprojekten gesammelten Arbeitsergebnisse sind jeweils adäquat zu dokumentieren, wie es die Studierenden in früheren Vorlesungen gelernt haben. Allein eine explizite Reflexionsphase, die jedes Teilprojekt abschließt, ist den Studie-

Horst Lichter, Martin Glinz (Hrsg.) (2001): Software Engineering im Unterricht der Hochschulen
SEUH 7 – Zürich, dpunkt.verlag, Heidelberg.



renden aus den vorbereitenden Vorlesungen nicht vertraut. Mit diesen Dokumenten weisen die Studierenden den Projektverlauf und ihren Lernfortschritt nach.

Die Planung ist zuweilen ein Stiefkind der vorbereitenden Vorlesungen, wohl mit Recht, weil Planungsfähigkeiten so schwer rein theoretisch zu vermitteln sind und nicht im Rahmen einer isolierten Hausaufgabe eingeübt werden können. Allerdings sind die Gegenüberstellungen von Planungen und wirklichem Projektverlauf eine der besten Reflexionsgelegenheiten. Daher wollen wir genauer erörtern, worauf es uns im Softwarepraktikum bei der Planung ankommt.

## 2.1 Vorgehen bei der Planung und Projektverfolgung

Die Planung wird aufgeteilt in Aufwands- und Ressourcenplanung. Die Aufwandsplanung enthält eine Aufteilung der in einem Teilprojekt zu leistenden Arbeit in Arbeitspakete und eine Aufteilung der in jedem Arbeispaket zu leistenden Arbeit in Teilaufgaben, die einzelnen Mitgliedern zugewiesen werden und für die die aufzuwendenden Ressourcen (hier: Arbeitszeiten) geschätzt werden sollen. Zusammen mit den Angaben der Ressourcenplanung werden daraus geplante Abschlussdaten für die Arbeitspakete festgelegt.

Die Ressourcenplanung enthält für jedes Gruppenmitglied eine Planung der aufwendbaren Arbeitsstunden über die nächsten Wochen der Projektlaufzeit. Typischerweise vergessen die Studierenden hier zunächst, die Verminderung der von ihnen aufwendbaren Zeit einzuplanen, die durch Klausurphasen und Feiertage entstehen. Dies wird ihnen aber bewusst, wenn aus solchen Gründen Verzögerungen auftreten.

Die erste Aufwands- und Ressourcenplanung des Teams kann nur „über den Daumen“ erfolgen, weil zunächst keine auswertbaren Erfahrungen vorliegen. Bei späteren Planungen sollten dann zuvor gemachte Erfahrungen einfließen. Abbildung 1 enthält ein Beispiel für die Auswertung von Plan und tatsächlichem Ablauf in einer Praktikumsgruppe, wie sie einem Reflexionspapier entnommen ist. Die erste Spalte enthält die Kurzbeschreibung des Arbeitspaketes und der darin enthaltenen Teilaufgaben. Die zweite sind Nummern; so heißt „K2.3.1”: Es handelt sich um das zweite Teilprojekt, darin das dritte Arbeitspaket, und darin wiederum die erste Teilaufgabe. Die letzten beiden Spalten stellen geplante und tatsächliche Aufwände für die Teilaufgaben und das gesamte Arbeitspaket einander gegenüber. Die Daten in der Kopfzeile sind geplante und tatsächliche Abschlussdaten des Arbeitspaketes. Die Ergänzung „(nf)” (für „nicht fertig”) gibt an, dass das Paket nicht wirklich abgeschlossen wurde.

Ein hinreichender Detaillierungsgrad ist bei der Aufwandsplanung nötig, damit man (1) überhaupt eine Chance hat, im Vorhinein zu beurteilen, ob der Plan realistisch ist; damit man (2) den Projektfortschritt während des Projektablaufs verfolgen kann, denn bei einer allzu geringen Zahl von Arbeitspaketen gibt es schlicht nicht genügend Abschlusstermine, deren Nichteinhaltung eine Verzögerung gegenüber dem Plan nachweist; und damit man (3) die Verantwortung für Teilaufgaben einzel-

nen Gruppenmitgliedern zuordnen kann. Im Laufe der Teilprojekte lernen die Studenten diese Konsequenzen aus der Gestalt eines guten Aufwandsplans zu schätzen.

| | | Planung | Tatsächlich |
|---|---|---|---|
| **Theoretisch Algorithmen entwickeln** | **K 2.3** | **29.05** | **29.05 (nf)** |
| Algorithmus 1. Teilaufgabe | K 2.3.1 | 10 h | 9:00 |
| Algorithmus 2. Teilaufgabe | K 2.3.2 | 6 h | 5:00 |
| Algorithmus 3. Teilaufgabe | K 2.3.3 | 6 h | 6:00 |
| Pseudo-Code für alle Teile mit abstrakten Roboterbefehlen | K 2.3.4 | 8 h | 0:00 |
| | | = 30 h | = 20:00 |

Das Paket wurde fristgerecht abgeschlossen, ohne dass der geforderte Pseudocode angefertigt worden ist. Das wurde ins Paket 2.6.1 verschoben. Die Unvollständigkeit dieses Pakets ist begründet durch einen zu hohen Ansatz der möglichen Ressourcen (Zeit). Es wurde nicht beachtet, dass die dieses Paket ausführenden Gruppenmitglieder in dieser Zeit Klausuren hatten und somit die fristgerechte Erfüllung dieses Pakets nicht möglich war. Da die folgenden Pakete den Abschluss dieses Pakets erforderten, haben wir den unerfüllten Teil in ein späteres Paket hineingelegt.

**Abb. 1: Gegenüberstellung von Planung und wirklichem Aufwand**

Die Studierenden lernen, dass es hilfreich ist, die geplanten Teilaufgaben einzelnen Gruppenmitgliedern oder wenigstens Paaren zuzuweisen, weil in vielen Teams allein schon das Unterlassen der Dokumentation von Aufgabenzuweisungen an Teammitglieder zu Missverständnissen führt. Typischerweise bewährt es sich zudem nicht, wenn für eine Aufgabe für alle Gruppenmitglieder der gleiche Aufwand eingeplant wird, weil auch in diesem Fall auch eine klare Verantwortungszuweisung unterbleibt.

## 2.2 Projektverfolgung

Ebenso wie das Planen ist die Verfolgung des Projektfortschritts ein notorisches Problem, weil in Softwareentwicklungsteams allzu oft der Wunsch die realistische Einschätzung trübt. Dabei gehört eine realistische Einschätzung des Stands im Projekt zu den wichtigsten Elementen erfolgreicher Softwareprojekte. Die Studierenden erkennen, dass es hierzu wichtig ist, von allen Gruppenmitgliedern regelmäßig Rückmeldungen über Arbeitsergebnisse abzufordern, und sie lernen so, den Wert von kleinen, klar von Folgearbeiten abgrenzbaren Arbeitspaketen zu schätzen, weil sich nur solche Arbeitspakete für die zuverlässige Verfolgung des Projektfortschritts eignen. So lernen sie, den Wert zahlreicher und aussagekräftiger Meilensteine für das Wissen über den Fortschritt im Projekt zu schätzen. Abbildung 2 gibt einen entsprechenden Auszug aus einem Reflexionspapier.

Die Projektverfolgung ist auf zwei Abstraktionsebenen möglich: (1) Die zyklische Projektstruktur sorgt dafür, dass es Meilensteine am Ende jedes Teilprojektes

gibt, und (2) innerhalb jedes Zyklusses werden Arbeitspakete definiert, die wieder jeweils einen eindeutig geplanten Endtermin haben. Hierdurch wird der Projektfortschritt innerhalb eines Teilprojektes verfolgbar.

Ein weiteres Problem in unserer Planung waren „fehlende Ergebnisse". D.h. es gab Teilaufgaben, an deren Ende kein konkretes Ergebnis gefordert war. So sollten selbständig Klassen entwickelt werden. Da die Umsetzung erst später erfolgen sollte, dachte auch keiner vorher an eine ausführliche Ausarbeitung der Klassen und der damit verbundenen Schnittstellen. Es musste so viel Arbeit nachgeholt werden, die eigentlich schon fertig sein sollte. Für unseren Gesamtplan war dies natürlich schlecht, da sich vieles verschob. Ergo
→ in Zukunft nur Meilensteine mit Vorzeigbarem planen.

**Abb.2: Erkenntnis des Wertes klarer Abschlusskriterien von Arbeitspaketen**

## 2.3 Prozessverbesserung durch Reflexion

Die Studierenden müssen in unserem Softwarepraktikum viele Aspekte ihres Projektes dokumentieren. Wir machen den Studierenden deutlich, dass diese Dokumentation der Nachweis darüber ist, dass sie während des Praktikums gelernt haben, ein mäßig umfangreiches Softwareentwicklungsprojekt im Team zu planen und durchzuführen und dabei wichtige Daten zu erheben, diese auszuwerten und Konsequenzen aus den gemachten Erfahrungen zu ziehen.

Das anzufertigende Produkt ist dabei von geringerer Bedeutung; es ist allerdings insofern wichtig, als eine allzu flagrante Verletzung der ursprünglichen Anforderungen, womöglich noch ohne rechtzeitige Erkenntnis, dass die ursprüngliche Aufgabenstellung nicht einzuhalten ist, ein Hinweis darauf ist, dass die Gruppe noch nicht verstanden hat, worauf es bei einem mäßig umfangreichen Softwareentwicklungsprojekt ankommt.

Wir legen Wert darauf, dass die Studierenden lernen, ihren Softwareentwicklungsprozess lernfreundlich zu organisieren. In unserem Ansatz ist ein wesentliches Element die Reflexion von Erfahrungen aus früheren Zyklen mit dem Ziel zur Verbesserung der zu benutzenden Prozesse in den späteren Zyklen. Grundlagen der Reflexion können Messungen und Beobachtungen bei der Prozessausführung in verschiedenster Form sein. Im Cottbuser Softwarepraktikum nutzen wir die folgenden Elemente:

- **Zeitprotokolle** der Gruppenmitglieder, insbesondere als Grundlage für den Vergleich von Plänen und tatsächlichem Ablauf. Es hat sich gezeigt, dass dies eine der fruchtbarsten Reflexionsgelegenheiten war. Dies hat wohl den Grund, dass die Gegenüberstellung von stark voneinander abweichenden harten Zahlen einen unabweisbaren Erklärungsbedarf anzeigt. Insbesondere schult dies die wichtigen Planungsfähigkeiten.

  Die Zeiterfassungen machen den Studierenden zudem bewusst, für welche Tätigkeiten im Rahmen der Softwareentwicklung welche Zeitanteile anfallen.

Dafür muss man aber natürlich die Aufwände entsprechend klassifizieren und auch quantitativ hinreichend exakt erfassen.

Die Studierenden dokumentieren ihre Reflexion bezüglich geplanter und wirklicher Aufwände pro Arbeitspaket und pro Person durch einen Reflexionsbericht am Ende jedes Teilprojektes und am Ende des Gesamtprojektes. Dieser enthält einen Vergleich von geplanten und tatsächlichen Aufwänden pro Arbeitspaket, und von geplanten und tatsächlich aufgewendeten Ressourcen pro Person, jeweils bestehend aus (a) Darstellung, (b) Interpretation und (c) der Angabe von daraus zu ziehenden Konsequenzen. Diese Reflexion betrifft die Genauigkeit der Aufwands- und Ressourcenplanung. Wir erwarten, dass die Gruppe sich über diese Punkte einen Konsens erarbeitet; dieser Teil des Reflexionspapiers ist also ein Gruppenergebnis. Hier werden harte Daten, nämlich die protokollierten Aufwandsdaten, interpretiert und Konsequenzen aus ihnen gezogen. Die schon erwähnte Abbildung 1 zeigt einen Auszug aus diesem Teil eines Reflexionspapiers.

- **Gruppenbesprechungen**. Diese sind eine zweifache Quelle von Reflexionsgelegenheiten: Einerseits werden gewisse Fragen direkt in den Gruppenbesprechungen reflektiert, andererseits dienen die Protokolle dem späteren Rückblick auf den Projektverlauf.

  Die Explizierung und Reflexion von Beobachtungen und Erkenntnissen, die in den wöchentlichen Gruppenbesprechungen zum Thema werden, liegt stark in der Hand der Betreuer. Hier kann die Aufmerksamkeit auf Probleme insbesondere der Kommunikation und der Kooperation in der Gruppe geleitet werden, die so gruppenspezifisch sind, dass sie im vorgegebenen Prozess nicht berücksichtigt sind.

  Wenn Ergebnisprotokolle der Gruppenbesprechungen sorgfältig geführt werden, so sind sie ein gutes Mittel, um bei der Abschlussreflexion festzustellen, wo man Lerngelegenheiten gut oder auch nicht gut genutzt hat, wenn man etwa feststellt, dass sich gewisse Probleme immer wieder wiederholen.
- Die Erarbeitung von **Coding-Styleguides und Review-Checklisten** und ihre Erprobung in der Praxis. Hier müssen sich die Studierenden gründlich über eher technische Fragen Gedanken machen und einen Konsens erzielen, und es wird deutlich, wie schwierig es ist, die wesentlichen Qualitätsanforderungen in eindeutigen Checklisten-Fragen zu formulieren.
- Zusätzlich zum gemeinsam zu erstellenden Reflexionsbericht, der die harten Daten auswertet, verlangen wir von jedem einzelnen Studierenden am Ende jedes Teilprojektes einen **persönlichen Reflexionsbericht**, in dem insbesondere subjektive und eher qualitative Erfahrungen darzustellen und auszuwerten sind.

Die Zweiteilung der an jedem Teilprojektende abzuliefernden Reflexionsberichte in einem gemeinsamen, auf harten Daten beruhenden Teil und einen persönlichen, auch „weiche“ Erfahrungen auswertenden Teil ermöglicht es den Studierenden, zum einen den Umgang mit harten Daten zu üben und den Wert der von ihnen gesammelten Aufwandsdaten schätzen zu lernen, und zum anderen aber auch eher intuitive Er-

kenntnisse festzuhalten und für das Vorgehen in Folgeprojekten zu nutzen. Die persönlichen Reflexionspapiere sichern zudem, dass jedes Gruppenmitglied sich ein paar Gedanken über den Prozessverlauf macht, und dass die Reflexionsarbeit nicht nur an einem oder zwei Gruppenmitgliedern hängenbleibt, was leicht geschieht, wenn nur ein Gruppenpapier verlangt wird.

Die sorgfältige Nutzung von selbstgemachten Erfahrungen ist nicht nur während der Ausbildung wichtig, sondern ist auch von eminenter Bedeutung für die Praxis, die gewöhnlich durch Turbulenz, also wechselnde Arbeitsbedingungen gekennzeichnet sind, besonders im expandierenden Beratungsgeschäft, wo kein Projekt ist wie das andere: (1) Die Zusammensetzung von Entwicklerteams zwischen Projekten und während Projekten wechselt. (2) Die Auftraggeber wechseln, und sie haben wechselnden Stärken und Schwächen bezüglich der Anforderungsdefinition. (3) Die Ansprechpartner beim gleichen Auftraggeber mit wechselnden Projektaufträgen wechseln. (4) Die zu verwendenden Werkzeuge wechseln durch Innovation.

## 2.4 Spezifische Techniken für Entwurf, Review und Test werden ausprobiert und ihre Wirksamkeit wird reflektiert

Die in Vorlesungen vorgestellten Techniken müssen von den Studierenden nicht einfach als wirksam akzeptiert werden, indem man der Autorität des Dozenten folgt. Vielmehr können die Leistungen der Techniken direkt erfahren und die für diese Wirkungen nötigen Vorbedingungen im Anschluss an eine Verwendung reflektiert werden. Abbildung 3 bietet ein Beispiel für einen Auszug aus einem Reflexionspapier, in dem deutlich wird, dass der Wert einer spezifischen Technik der Softwaretechnik, nämlich von Detaildesigns erkannt wurde.

> Besonders schlecht ist in diesem Teilprojekt meine persönliche Arbeit erfolgt. Die Implementierung der Teilanlage 3 wurde zweimal von mir abgebrochen und nahezu vollständig von vorn begonnen. Ursache ist sicherlich die mangelnde Planung, es reicht eben, wie ich erfahren durfte, nicht aus, sich zu einer Klasse wohlklingende Methoden als Schnittstelle auszudenken, solange man nicht genügend Zeit in die Erkennung von Zusammenhängen investiert.

**Abb. 3: Einsicht in den Wert von Detaildesigns**

Hierbei kann ein Problem darin bestehen, dass in einem Softwarepraktikum die entsprechenden Anwendungsbedingungen, etwa aufgrund noch fehlender Erfahrung der Studierenden mit einer Technik, nicht gegeben sind. Dies kann wohlgeprüfte und unter entsprechenden Bedingungen leistungsfähige Techniken, etwa Qualitätssicherungsmaßnahmen wie Reviews oder systematische Tests, in den Augen der Studierenden ungerechtfertigterweise diskreditieren. Dieses Problem können wir nicht lösen, wenn wir die eigenen Erfahrungen der Studierenden betonen, sondern können ggf. nur darauf hinweisen, daß die gemachten Erfahrungen möglicherweise nicht repräsentativ sind.

Horst Lichter, Martin Glinz (Hrsg.) (2001): Software Engineering im Unterricht der Hochschulen SEUH 7 – Zürich, dpunkt.verlag, Heidelberg.



## 2.5 Koordinations- und Kommunikationsprobleme

In den (Teil-)Projektberichten wird erwartet, daß Koordinations- und Kommunikationsprobleme in der Gruppe dargestellt und reflektiert werden.

Viele Probleme in den Softwarepraktikumsprojekten hängen mit einem Mangel an Koordination zusammen: Die Gruppen treffen sich nicht regelmäßig, oder sie thematisieren nicht die jeweils relevanten Probleme. In verschiedenen Projektphasen haben solche Probleme typischerweise verschiedene Gestalt: (1) am Anfang, wenn sich die Gruppenmitglieder noch nicht kennen; (2) in Motivationstiefs, wenn kein rechter Fortschritt erkennbar ist; und (3) nach „Projektpausen“ (etwa Klausurzeiten, Feiertagen), wenn ein Projekt erst langsam wieder anläuft. Die Betreuer achten auf diese Probleme, machen sie explizit, versuchen, Konsens zu erzielen, und verlangen, dass sie im Ergebnisprotokoll festgehalten werden, damit die Erfahrung später im Rückblick ausgewertet werden kann.

### Kommunikation

In den Abbildungen 4 und 5 finden sich Auszüge aus Reflexionspapieren mehrerer Mitglieder einer Gruppe, in der nach anfänglicher Dominanz des Entwurfsgeschehens durch eine Person ein ausgeglichenerer Entwurfsprozess hergestellt werden konnte. Im Auszug der Abbildung 4, der von einem Studenten mit relativ viel Entwicklungserfahrung stammt, wird auch deutlich, dass Entwurfshilfsmittel auch als wichtige Kommunikationsmittel erkannt wurden. Dass sich die Urteile in der Gruppe decken, belegt ein Reflexionsbericht eines anderen Gruppenmitglieds, von dem ein Auszug in Abbildung 5 wiedergegeben ist.

Anfangs gab es viele Schwierigkeiten, die durchaus gelegentlich in Verzweiflung resultierten. Kommunikation war zu diesem Zeitpunkt innerhalb der Gruppe eher eine Pflicht als eine Hilfe. Zusammenarbeit bestand im Wesentlichen darin, dass ich einen Vorschlag machte, keiner (wirklich) etwas dazu sagte und der Vorschlag schließlich angenommen wurde. Gelegentlich hatte ich das Gefühl, Diktator zu sein, der seinen Untergebenen nur sagt, was sie zu tun haben. Natürlich führte das dazu, dass die meisten Gruppenmitglieder keinen Überblick über das Projekt erhielten. Schließlich war die entworfene Klassenstruktur aus meiner Sicht für die meisten eher unverständlich als schlüssig.
Und so kam es, wie es kommen musste: Während der Auswertung nach Abschluss des Teilprojektes ließ dann jeder einmal so richtig Luft ab. Nachdem ich dabei relativ viel abbekam, aus meiner jetzigen Sicht sogar großteils gerechtfertigt, ergab sich als Resultat übereinstimmend, dass an der Kommunikation innerhalb der Gruppe wesentlich gearbeitet werden muss. Aus diesem Grund hielten wir es für erforderlich, auch die Moderation der Gruppentreffen abwechselnd verschiedenen Mitgliedern zuzuweisen. Allerdings war es dabei am Anfang relativ schwierig, viele Schweigesekunden zu überbrücken und ein Thema so lange zu vermeiden, bis es der Moderator angesprochen hatte.
Dieses Konzept hat sich jedoch bewährt. Die im ersten Teilprojekt aufgetretenen Probleme waren seit dieser Besprechung wie weggeblasen. Plötzlich hatte jeder Lust, sich intensiv in das Projekt einzubringen, es kamen Vorschläge zu Entwürfen, organisatorische Sachen und Umsetzungen. Man könnte meinen, diese Besprechung wäre eine Art Befreiungsschlag gewesen, nachdem keiner mehr Angst hatte, seine Meinung zu den verschiedensten Themen zu sagen.

**Abb. 4: Einsicht in die Folgen fehlender Kommunikation über Entwürfe.**

Probleme, die sich im Softwarepraktikum ergaben, waren am Anfang durch die großen Unterschiede der Programmiererfahrungen und den verschiedenen Herangehensweisen jedes Einzelnen in der Gruppe an das Praktikum. Ich hatte am Anfang das Gefühl, überrannt zu werden, da ich nicht hinterher kam. Dies kam vor allem, daß F. sofort mit einem Klassendiagramm kam und ich mir erst einmal über ein Vorgehen in einem Softwareprojekt klar werden musste. Nachdem wir F. erst einmal gebremst hatten und er sich offensichtlich auch selbst ein wenig zurückgehalten hatte, wurde mir einiges klar und das Softwarepraktikum konnte beginnen.

**Abb. 5: Bewertung von Entwurfsgeschehen und Reviews von Seiten eines weniger erfahrenen Studierenden**

### Konflikte

Ein typischer Fall für ein Kooperationsproblem ist, dass ein Gruppenmitglied sich nicht genügend engagiert. Studentische Solidarität führt oft dazu, dass dies gegenüber dem Betreuer nicht transparent gemacht wird. Die Gegenüberstellung von geplanten und tatsächlichen Aufwänden macht es den Betreuern erfahrungsgemäß etwas leichter, solche Phänomene nach Abschluss eines Teilprojektes zu erkennen und ihre Behandlung sowie die Dokumentation der Entscheidung über Konsequenzen in einem Projektbericht zu verlangen. Wird ein solcher Missstand in einem frühen Teilprojekt deutlich, dann hat die Gruppe auch Gelegenheit, ihn in späteren Teilprojekten abzustellen oder angemessene Konsequenzen zu ziehen.

Die persönlichen Reflexionspapiere sind ein Prozesselement, das in manchen Gruppen dazu genutzt wird, Frustrationen über andere Gruppenmitglieder Ausdruck zu geben. Hier haben wir auch schon den Fall erlebt, dass der Betreuer der Gruppe dabei helfen konnte, Missverständnisse, die den Konflikten zugrundelagen, aufzuklären und damit gewisse Konflikte zu entschärfen.

Auch die Konfliktfähigkeit wurde geprobt. In allen Gruppen treten Konflikte auf. In den Gruppensitzungen besteht die explizite Gelegenheit, zu reflektieren, ob und wie man diese Konflikte thematisieren kann, um in Folgeteilprojekten die schädlichen Folgen unbearbeiteter Konflikte zu vermeiden und die (womöglich vorhandenen) fruchtbaren Konfliktursachen produktiv zu nutzen. Abbildung 6 liefert einen Auszug aus einem persönlichen Reflexionspapier eines Mitglieds einer Gruppe, in der ein Gruppenmitglied zuweilen „ausfiel“.

Der eigentlich unangenehmste Teil dieses Teilprojektes war jedoch nicht inhaltlicher sondern organisatorischer Art. Ein Gruppenmitglied, dessen Motivation bzw. Zuverlässigkeit bereits aus vorherigen Teilprojekten bekannt war, ließ uns in diesem Teilprojekt besonders hängen. Am ärgerlichsten war hier jedoch nicht seine Abwesenheit, sobald sich die Woche dem Ende näherte (meistens so ab Mittwoch), sondern die Tatsache, dass er uns bei dem ersten durchzuführenden Review hat vergeblich warten lassen. Eine Verschiebung des Termins war somit die Folge.

**Abb. 6: Hinweise auf einen Konflikt in der Gruppe**

Horst Lichter, Martin Glinz (Hrsg.) (2001): Software Engineering im Unterricht der Hochschulen
SEUH 7 – Zürich, dpunkt.verlag, Heidelberg.



### 2.6 Erfolgskriterien

Ein für die Studierenden wichtiger Punkt in vielen Lehrveranstaltungen ist die Frage, wie sie nachweisen sollen, dass sie die Veranstaltung erfolgreich absolvieren. Im Praktikumsleitfaden, der den Studierenden am Anfang der Veranstaltung ausgeteilt wird, finden die Studierenden den in Abbildung 7 wiedergegebenen Passus. Es hat sich bewährt, die Studierenden während der Projektlaufzeit immer wieder auf diesen Passus hinzuweisen, weil dies die Motivation, die entsprechenden Dokumente anzufertigen und die Protokolle sorgsam zu führen, erheblich stützte.

Für den Scheinerwerb sind wenigstens folgende Dokumente vorzulegen:
Aussagekräftige Zeitprotokolle jedes einzelnen Gruppenmitglieds über die gesamte Projektlaufzeit
Stilfibel für das Kodieren und für das Dokumentieren
Je einen Plan für jedes der Teilprojekte, je mit inhaltlicher Beschreibung, Aufwands abschätzung, Abschlussdatum und Datum der Planung.
Nach jedem Teilprojektabschluss: Gegenüberstellung von Planungen und tatsächlichem Projektablauf mit Erläuterungen in einem Reflexionspapier, das auch die explizite Lehren enthält, die die Teilnehmer aus den Abweichungen gezogen haben.
Bei Änderungen von Plänen ausstehender Teilprojekte: Überarbeitete Pläne, Begründung für die Veränderung
Getestete, dokumentierte und der Stilfibel entsprechende Programme für jedes Teilprojekt
Wenigstens ein Protokoll über die Durchführung eines Reviews
Die Ergebnisprotokolle der gemeinsamen Sitzungen, über die gesamte Projektlaufzeit
Ein zusätzliches Reflexionspapier von jedem einzelnen Gruppenteilnehmer zum Projektschluss.

Weitere notwendige Bedingungen sind:
Kontinuierliche Mitarbeit an den Teilprojekten über die gesamte Projektlaufzeit
Regelmäßige Teilnahme an den Plenums- und den Betreuerterminen
Eingehende Kenntnis nicht nur der selbst angefertigten Teile von Programm und anderen Arbeitsergebnissen, sondern auch der Teile, die von anderen Gruppenmitgliedern angefertigt sind.

**Abb. 7: Erfolgskriterien im Cottbuser Softwarepraktikum**

An anderer Stelle des Leitfadens wird beschrieben, wie etwa ein Ergebnisprotokoll, ein Reviewprotokoll oder ein Testplan zu gestalten ist.

## 3 Erfahrungen und Ausblicke

Die Sichtweise von Softwareentwicklung als Tätigkeit mit expliziten Reflexionsaspekten als Leitline für die Gestaltung des Softwarepraktikums ist bei uns im Laufe der Semester immer relevanter geworden. Wir haben die Erfahrung gemacht, dass eine immer konsequentere Gestaltung des Softwarepraktikums mit der Maßgabe, Softwareentwicklung als reflektiertes Handeln zu organisieren, zu immer befriedigenderen Lernergebnissen geführt hat. Auch die Leistungen erfolgreicher Prozesselemente, die wir ursprünglich aus anderen Gründen in das Softwarepraktikum eingeführt hatten, etwa die Projektplanung, die Aufwandsprotokollierung und die

Durchführung von Reviews, ließen sich oft nachträglich auch damit begründen, dass sie das Wechselspiel von Praxis und Reflexion unterstützen.

Wir selbst haben erst mit der Zeit gelernt, wie Reflexionselemente in die Organisation eines Softwarepraktikums eingefügt werden können. Bei unserem ersten Entwurf für einen Leitfaden für das Softwarepraktikum vor drei Jahren haben wir noch einen einfachen Durchlauf durch ein Wasserfallmodell der Softwareentwicklung empfohlen und keine Anfertigung von Reflexionspapieren verlangt. Dieser Leitfaden war in der Ausbildungspraxis aber auch wenig erfolgreich. Wir haben die in diesem und allen späteren Durchläufen durch das Softwarepraktikum gesammelten Erfahrungen in Zusammenarbeit mit den Praktikumsteilnehmern ausgewertet, Konsequenzen daraus gezogen und diese durch Überarbeitung des Leitfadens fixiert.

Unser Hauptziel ist der Lernerfolg der Studierenden. Es ist nicht eindeutig, wie man diesen messen müsste. Nach objektiven Kriterien, etwa hinsichtlich der Planungsgenauigkeit, sind wir wohl nicht erfolgreich. Dafür ist die Zahl von (typisch etwa) drei Zyklendurchläufen wohl auch zu klein, um schon einen Lernerfolg konkret nachzuweisen. Allerdings gibt es eine Vielzahl weniger harter Kriterien, die uns an einen Erfolg glauben lassen. Inbesondere die Reflexionsberichte weisen darauf hin, dass die Studierenden lernen; allerdings lässt sich aus diesen nicht ablesen, inwiefern der Transfer aus der (theoretischen) Einsicht in die Praxis gelingt.

Der bisher von den Betreuern des Praktikums erbrachte Aufwand wird mit steigenden Studierendenzahlen nicht mehr tragbar sein. In Erwartung eines solchen Anstiegs in den kommenden Jahren haben wir im Sommersemester 2000 versucht, den Betreuungsaufwand zu vermindern und gleichzeitig Studierenden aus dem Hauptstudium eine neuartige Lerngelegenheit zu verschaffen. Wir haben im Rahmen eines Seminars im Hauptstudium Studierende als Berater zur Betreuung einiger Gruppen eingesetzt. Sie haben den Projektverlauf beobachtet und die Gruppen in technischen, vor allem aber in organisatorischen Fragen beraten. Neben dieser allgemeinen Beratung gehörte es zu den Aufgaben der Seminarteilnehmer, sich ein spezielles Problemgebiet auszusuchen, hinsichtlich dessen die Praktikumsgruppe besonders untersucht werden sollte, und hinsichtlich dessen sie ihre eigene Beratungstätigkeit reflektieren sollten. Für dieses Problemgebiet sollten Beobachtungen gemacht und Daten gesammelt, dokumentiert und interpretiert werden, eigene Interventionen sollten beschrieben und hinsichtlich ihres Erfolgs bewertet werden. Beispiele für solche Problemgebiete waren Planung und Projektverfolgung, Entstehung und Behandlung von Konflikten, und die Rollenverteilung in der betreuten Arbeitsgruppe.

Die Aufgabe der Berater bei der Betreuung war es also, der Gruppe Probleme bewusst zu machen, sie bei der Lösung der Probleme zu beraten, und ihre eigene diesbezügliche Tätigkeit zu reflektieren. Allerdings verfügen die Berater nicht über formale Autorität, und entsprechend haben sie auch keine offizielle Verantwortung für den Gruppenerfolg.

Wir nutzten das Softwarepraktikum im Sommersemester 2000 mithin in zweierlei Weise: (1) Es war eine Möglichkeit für die Praktikumsteilnehmer im Grundstudium, Erfahrungen zu machen und daraus zu lernen. (2) Es war eine Möglichkeit für Stu-

Horst Lichter, Martin Glinz (Hrsg.) (2001): Software Engineering im Unterricht der Hochschulen
SEUH 7 – Zürich, dpunkt.verlag, Heidelberg.



dierende im Hauptstudium, den Ablauf von Softwareentwicklungsprojekten am Beispiel zu erleben, Schwierigkeiten der Einflussnahme kennenzulernen, und beides zu reflektieren.

Unsere ersten positiven Erfahrungen sind, dass die Seminarteilnehmer lernten, die Mitglieder der von ihnen betreuten Praktikumsgruppe in verschiedener Hinsicht sehr differenziert zu beurteilen.

Eine Überraschung, die wir bei der Durchführung der Praktika in den letzten Jahren erlebten, bezog sich auf die Unterschiedlichkeit der Erfahrungen, die die Studierenden in den verschiedenen Teilprojekten machten. Es hat sich herausgestellt, dass die meisten Gruppen die Gesamtaufgabe nicht in die empfohlenen vier, sondern nur in drei Teilprojekte aufgeteilt haben. Dabei hat sich gezeigt, dass in den verschiedenen Teilprojekten verschiedene Probleme und Lernbereiche dominieren:

1. Teilprojekt: Planungsprobleme werden explizit, Aufwandsprotokollierung und Reflexionselemente werden, zuweilen mit Schwierigkeiten, eingeführt.
2. Teilprojekt: Konsolidierung und erste Anwendung des im ersten Teilprojekt Gelernten; es treten Motivationstiefs auf, die expliziert und behandelt werden müssen, sowie andere Probleme der Arbeitsorganisation, insbesondere Konflikte unter den Gruppenmitgliedern; erste schwerwiegendere technische Probleme treten auf und müssen gelöst werden.
3. Teilprojekt: Es entsteht Projektdruck durch den festen Endtermin, aufgrund dessen man keine Teilaufgaben mehr in spätere Teilprojekte verschieben kann, wie das am Ende der früheren Teilprojekte noch möglich gewesen war. Die technischen Schwierigkeiten spitzen sich gemeinsam mit den organisatorischen zu.

Unsere Perspektive für das Softwarepraktikum ist, dass wir den Studierenden zunehmend effektiv, aber auch effizient vermitteln wollen, wie sie ihre eigene Arbeit sowohl während des Studiums als auch später im Beruf als Lernprozesse organisieren können. Insbesondere wollen wir vermitteln, wie man Softwareentwicklungsprojekte als Lernprojekte organisiert. So bleibt der oft gehörte Slogan „das Lernen lernen" für sie nicht abstrakt, sondern wird mit einer konkreten Vorgehensweise unterlegt.

Wir haben die Erfahrung gemacht, daß die expliziten Reflexionsdokumente zu einer wesentlich qualifizierteren Diskussion sowohl zwischen den Praktikumsteilnehmern als auch mit den Auftraggebern und Beratern führten und eine sachliche Analyse von positiven und negativen Projektverläufen erlaubten. Die Studierenden haben gelernt, gezielt Maßnahmen zur Veränderung ihrer Arbeitsprozesse zu ergreifen und die Effekte zu bewerten. Dadurch entstand vor allem eine erheblich realistischere Einschätzung der grundsätzlichen Komplexität von Software-Entwicklungsprozessen und der Schwierigkeiten von Teamarbeit.

Im Zusammenhang mit dem Softwarepraktikum gibt es Lernende auf verschiedensten Ebenen: Die Teilnehmer des Softwarepraktikums lernen die Anwendung der konkreten Techniken und die reflektive Auswertung der dabei gemachten Erfahrungen; fortgeschrittene Studierende unterstützen als Moderatoren diese Reflexion-

prozesse und reflektieren selbst ihre Moderationstätigkeit; und Dozenten und wissenschaftliche Mitarbeiter sammeln Erfahrungen mit der Durchführung von Praktika und reflektieren diese ebenfalls mit dem Ziel der Prozessverbesserung.

## Danksagung

Wir danken den Teilnehmern des Cottbuser Softwarepraktikums der letzten Jahre und insbesondere unserem Kollegen Hans-Gerd Köhler für Mitwirkung an unserem ständigen Lernprozess.

### Literatur

1. Anderson, A.; Beattie, R.; Beck, K.; Bryand, D.; DeArment, M.; Fowler, M.; Fronczak, M.; Garzaniti, R.; Gore, D.; Hacker, B.; Hendrickson, Ch.; Jeffries, R.; Joppie, D.; Kim, D.; Kowalsky, P.; Mueller, D.; Murasky, T.; Nutter, R.; Pantea, A.; and Thomas, D.: Chrysler goes to „extremes". *Distributed Computing*, S. 24 - 28, October 1998.
2. Beck, K.: *Extreme Programming Explained: Embrace Change*. Addison Wesley Longman, Reading/Massachusetts, 1999.
3. Humphrey, W. S.: *A Discipline for Software Engineering.* Addison-Wesley, Reading/Massachusetts, 1995.
4. Humphrey, W. S.: *Introduction to the Team Software Process.* Addison-Wesley, Reading/Massachusetts, 2000.
5. Pasch, J.: *Softwareentwicklung im Team.* Springer-Verlag, Berlin, 1994.
6. Schön, D. A.: *The Reflective Practitioner.* Basic Books, New York, 1982.
7. Schön, D. A.: *Educating the Reflective Practitioner.* Jossey-Bass Publishers, San Francisco, 1986.
8. Schön, D. A. (Hrsg.).: *The Reflective Turn. Case Studies In and On Educational Practice.* Teachers College Press, 1991.