

Selbstbestimmung oder Anleitung: Erfahrungen mit einem Softwaretechnikpraktikum im Bereich Qualitätssicherung

Sebastian Jekutsch, Christopher Oezbek, Stephan Salinger

Institut für Informatik, Freie Universität Berlin, {jekutsch,oezbek,salinger}@inf.fu-berlin.de

Zusammenfassung

Wir berichten von zwei ähnlichen Softwaretechnikpraktika. In der ersten Version wurde ein freies Softwaresystem anhand wöchentlicher Übungsblätter hinsichtlich Qualitätsmängel untersucht. Im Kontrast zu diesem sehr angeleiteten Vorgehen gab es in der zweiten Version lediglich die eine Zielvorgabe, möglichst viele Schwachstellen zu finden. Der Weg dorthin blieb den Studierenden überlassen.

1 Einführung

Im Folgenden werden zwei Durchführungen des Praktikums Softwaretechnik an der Freien Universität Berlin vergleichend beschrieben. Das Praktikum behandelte in beiden Fällen Themen der analytischen Qualitätssicherung anhand eines freien Softwaresystems, einem Open Source Content-Management-System (CMS).

Die Durchführungen der Jahre 2004 und 2005 unterschieden sich kaum in den Lernzielen und den fachlichen Inhalten, sondern lediglich in der Vorgehensweise: Während das Praktikum 2004 eher Vorlesungscharakter mit praktischen, aufeinanderbauenden Übungsaufgaben hatte, fehlten 2005 Übungsblätter und Anweisungen beinahe gänzlich. Stattdessen wurden lediglich die Erfolgskriterien für die ansonsten eigenständig zu planende Arbeit vorgegeben.

Beide Versionen des Praktikums enthielten die folgenden Inhalte:

- Nach einer Einführung in allgemeine Anforderungen an Content-Management-Systeme galt es zunächst, sich in die Software einzuarbeiten.
- Selbst auszuwählende Teile der Software wurden anhand verschiedener Qualitätskriterien untersucht.
- Wurden funktionale Schwächen entdeckt, so waren sie in die für uns zugängliche Fehlerdatenbank einzutragen. Die Entwickler des CMS haben die neuen Einträge daraufhin bewertet.

- Zum Abschluss des Praktikums wurde von den Studierenden ein Paket von automatisierten Testfällen, Analysen und gemeinsam erstellten Ratschlägen an die Entwickler des CMS übergeben.

Bei der Konzeption lagen folgende Kriterien zugrunde, die ein Softwaretechnikpraktikum aus unserer Sicht erfüllen sollte (siehe auch [Prechelt93], [Shaw00]):

1. Theoretisch vermittelte Techniken und Methoden sollen angewendet und die Konsequenzen einer Nichtanwendung im Vergleich zu einem unreflektierten Ad-Hoc-Vorgehen erkannt werden.
2. Das Anwendungsbeispiel sollte eine realistische Größe haben und nicht lediglich akademisches „Spielzeug“ sein.
3. Die Aufgabe soll nicht von einer Person alleine lösbar sein.
4. Zeitmanagement, Kommunikation, Konfliktlösung und Entscheidungsfindung sollen geübt werden.

Voraussetzung war der Abschluss der Vorlesung Softwaretechnik. Die Studierenden arbeiteten in Teams von drei bis vier Personen. Es gab rotierend einen Gruppenleiter, der Ansprechpartner für uns war.

2 Angeleitete vs. selbstbestimmte Version

Die erste Durchführung des Praktikums besaß Vorlesungscharakter:

- Es gab zwei Präsenztermine in der Woche: einen von uns durchgeführten Vortragstermin und eine Praxisstunde. Die restliche praktische Arbeit war in Eigenregie zu leisten.
- Die Vorträge wiederholten relevante Themen aus der Softwaretechnikvorlesung mit speziellem Bezug auf die Praktikumsituation. Im wöchentlichen und zweiwöchentlichen Rhythmus wurden Übungsblätter mit festem Fertigstellungstermin verteilt.
- In der Praxisstunde wurden die Ergebnisse der vergangenen Übungsblätter vorgestellt und diskutiert.

Es gab vier große Themenblöcke: Modultests für einen kleinen Bereich der Software, Durchsichten bzw. statische Codeanalyse für einen anderen Teil und Abnahme einer speziellen Funktionalität der Software¹.

Letztlich blieben die Ergebnisse des Praktikums hinter den Erwartungen zurück. Es wurden nur wenige Defekte gefunden, die erstellten Testfälle waren trivial, die Durchsichten schienen dem Ziel zu folgen, möglichst einfache Perspektiven einzunehmen und

¹ Wir gehen auf die Inhalte im Folgenden nicht weiter ein. Genauereres kann auf der Veranstaltungswebsite <http://projects.mi.fu-berlin.de/w/bin/view/SWTprak/> nachgelesen werden.

bei der funktionalen Abnahme äußerten sich Fehldeutungen, was die Software leisten müsse und in den Abnahmekriterien auftauchen solle.

Für die schwachen Ergebnisse machten wir vor allem den Charakter des Praktikums verantwortlich:

- Vordergründiges Ziel der Studierenden schien es gewesen zu sein, das Übungsblatt zwar rechtzeitig, aber ohne viel Aufwand zu lösen. Die Ergebnisse bzgl. der impliziten Zielsetzung, die Qualität der Software durch Aufdecken von Defekten und Versagen zu erhöhen, blieben hinter den Möglichkeiten zurück.
- Bei der Besprechung wurde das schwache Ergebnis entweder mit der schlecht strukturierten und dokumentierten Software begründet oder mit den ungenauen Vorgaben in den Übungsblättern.

Dies geschah, obwohl wir regelmäßig die Kriterien an ein gutes Ergebnis wiederholten.

Aufgrund der gewonnenen Erfahrung versuchten wir, die zweite Durchführung im nachfolgenden Jahr am zu erreichenden Ergebnis zu orientieren.

Das Praktikum bestand nun im Wesentlichen aus zwei Arbeitsaufträgen:

1. Es sollten so viele relevante Versagensfälle der Software provoziert werden wie möglich. Die Relevanz errechnete sich hierbei vor allem aus der Einstufung (severity) des Versagensberichts durch die Entwickler des CMS.
2. Jede Gruppe musste wöchentlich auf der Webseite der Veranstaltung einen Bericht hinterlegen, der den beschrittenen Weg, die zugrunde liegende Überlegung, die erzielten Ergebnisse und die zukünftigen Arbeitsschritte beschreibt.

Wie man die Versagensfälle findet, blieb den Studierenden überlassen. Es gab in der großen Runde regelmäßig Rückmeldung und Diskussionen zu angemessenen und gewählten Vorgehensstrategien. Für die Endnote war eine gut begründete Vorgehensweise genauso wichtig wie ihr Erfolg.

3 Vergleich der Versionen und Schlussfolgerungen

Zum Vergleich interessant sind

- der Erfolg im Sinne der Qualitätssicherung. Hier zeigte die zweite Durchführung bessere Ergebnisse bei vergleichbaren Rahmenbedingungen. Der Grund lag unter anderem schlicht im erhöhten Arbeitseinsatz.
- der Lernerfolg im Sinne der Softwaretechnikausbildung. Dieser ist schwer zu benennen, da es keine Abschlusskontrolle gab. Unbestritten ist, dass sich deutlich verschiedene Schwerpunkte ergaben.
- das Interesse der Studierenden im Verlauf der Veranstaltung. Hier polarisierte die zweite Version mehr: Während die erste Version

durchschnittliche Noten von Seiten der Studierenden bekam, fielen bei der selbstgeleiteten Version die Stimmen sowohl stark positiv als auch negativ aus. Wir sahen also sowohl mehr Spitzenleistung als auch mehr demotivierte Studenten in der zweiten Version.

Die Studierenden haben in der zweiten Version sowohl im Umfang (Anzahl gefundener Versagensfälle und Menge der zumindest durchdachten Techniken) als auch in der Güte der Ergebnisse (Relevanz der Versagensfälle gemessen an der vergebenen „severity“ im Defektverfolgungssystem und deren Dokumentation, bevorzugt in Form von automatisierten Testfällen) eine bessere Leistung gezeigt.

Durch die Pflicht, sich selbst um die einzusetzenden Methoden und Techniken zu kümmern, stand dabei natürlich zunächst nicht deren Durchführung im Vordergrund, sondern deren Ökonomie, also die Frage, ob es sich überhaupt lohnen wird, die Technik einzusetzen.

Die Unzufriedenheit der Studierenden mit dem zweiten Praktikum resultierte aus der Unklarheit der Vorgehensweise. Die gestellte Aufgabe war weder intuitiv noch nach „Schema F“ zu lösen, die potentiell defekte Software sehr umfangreich und das Ziel nicht schon nach ein paar Tagen zu erreichen. Dies traf nicht das gewohnte Lern- und Arbeitsschema an einer Universität.

Darüber hinaus offenbarte sich die Gruppenarbeit als schwierig. Ein verbreiteter Irrtum war, dass Gruppenarbeit gemeinsame gleichzeitige Arbeit an einem Problem bedeutet. Zudem entstand ein Wettbewerb zwischen den Gruppen, weil ein einmal berichteter Mangel natürlich für die anderen Gruppen „verloren“ war. Dieser Wettbewerbscharakter hat vielen Teilnehmern missfallen.

Trotzdem sind wir Veranstalter zufriedener mit dem (auch objektiv besseren) Ergebnis der zweiten Version, denn wir sind uns sicher, in dieser freien Form praxisrelevantere Lernziele erreicht zu haben als bei der Orientierung an Übungsblättern.

Allerdings erscheint es uns notwendig, eine Balance zwischen dem bequemen Wochentakt der angeleiteten Version und dem elfwöchigen Alleingang des zweiten Durchlaufs zu erreichen. Wir wollen Zwischenstopps einführen, bei denen man Teilziele zu erreichen hat, die für sich einen Wert haben. Am Ende eines solchen zeitlich klar begrenzten Blocks bleibt dann die Zusammenfassung der einzelnen Ergebnisse, die jede Gruppe für ihre eigene Weiterarbeit nach Belieben verwenden kann.

Zudem werden wir die Gruppen und ihr inneres Funktionieren genauer beobachten und in Zweifelsfällen deeskalierend eingreifen müssen.

Literaturverzeichnis

[Prechelt93] Lutz Prechelt: Ziele und Wege für Softwaretechnik-Praktika. In *Proceedings Workshop SEUH'93*, B.G Teubner, 1993, pp. 78-82

[Shaw00] M. Shaw. Software engineering education: A roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (Limerick, Ireland, June 04 - 11, 2000). ICSE '00. New York, pp. 371-380