Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

Ein Experiment zur Ermittlung der Programmierproduktivität von Studenten

Matthias Wetzel

Universität Stuttgart
Universitätsstraße 38, 70569 Stuttgart
wetzelms@informatik.uni-stuttgart.de

Zusammenfassung

Unterschiede in der Programmierproduktivität zwischen einzelnen Programmierern sind ein wichtiger Gegenstand der Forschung. In diesem Beitrag wird ein zweiteiliges Experiment mit Softwaretechnikstudenten der Universität Stuttgart zur Ermittlung der individuellen Programmierproduktivität beschrieben. Im Ergebnis liegt zwischen der Programmierproduktivität der langsamsten und der schnellsten Studenten der Faktor 2,7. Nach einer Beschreibung des Experimentaufbaus und der Resultate wird auf Verbesserungsmöglichkeiten im Experimentdesign und Schlussfolgerungen aus den Ergebnissen im Hinblick auf mögliche Ansätze zur Verbesserung der Programmierausbildung im Studium eingegangen. Ein mittlerweile bereits umgesetzter Ansatz ist die Einrichtung zusätzlicher betreuter Programmierübungen parallel zu bestehenden Lehrveranstaltungen.

1 Einleitung

Spätestens seit der Studie von Sackman [Sac68] sind Unterschiede in der Programmierproduktivität zwischen einzelnen Programmierern ein wichtiger Gegenstand der Forschung. Auch wenn neuere Untersuchungen [Dic81] [Pre01] die Ergebnisse von Sackman zum Teil relativiert haben, sind in der Praxis zwischen schnellen und langsamen Programmierern Unterschiede von mindestens 4:1 im Hinblick auf die reine Programmierproduktivität zu erwarten [Pre01].

Im akademischen Umfeld stellt sich die Frage, welche Unterschiede zwischen Studienanfängern bestehen und ob sich diese Unterschiede im Lauf des Studiums

Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

tendenziell vergrößern oder verkleinern. Die Beantwortung dieser Frage lässt u.a. Rückschlüsse auf eine optimale Gestaltung der praktischen Lehrveranstaltungen im Studium zu, insbesondere im Hinblick auf die gezielte Förderung schwächerer Studenten.

Dieser Beitrag beschreibt ein im Jahr 2005 an der Universität Stuttgart im Studiengang Softwaretechnik durchgeführtes Experiment, in dem die individuelle Programmierproduktivität von Studenten vor und nach einem Softwarepraktikum im Grundstudium verglichen wurde. Im Unterschied zu z.B. der Studie in [McC01], bei der ebenfalls die Programmierfähigkeiten von Studenten untersucht wurde, wurde dabei keine Bewertungsskala vorgegeben, sondern lediglich die Leistung der Studenten bei der Implementierung von Funktionalität miteinander verglichen.

Folgende Struktur liegt dem Beitrag zugrunde: In Kapitel 2 wird kurz die Programmierausbildung im Studiengang Softwaretechnik beschrieben. Kapitel 3 beschreibt die Durchführung und Kapitel 4 die Resultate des Experiments. In Kapitel 5 wird auf einige Probleme des Experimentdesigns eingegangen, und Kapitel 6 fasst die Ergebnisse zusammen und schließt den Beitrag mit einem Ausblick ab.

2 Programmierausbildung im Studiengang Softwaretechnik

Die im Studiengang Softwaretechnik durchgeführten praktischen Lehrveranstaltungen werden in [Lud97] ausführlich beschrieben. Für das durchgeführte Experiment ist neben den praktischen Lehrveranstaltungen *Programmierkurs* im ersten und *Softwarepraktikum* im dritten und vierten Semester insbesondere die Vorlesung *Programmentwicklung* im dritten Semester von Bedeutung.

Im Rahmen des Programmierkurses sollen den Studenten im ersten Semester die Grundlagen des »Programming in the small« gemäß softwaretechnischen Prinzipien vermittelt werden. Dabei müssen die Studenten in Dreiergruppen zunehmend komplexer werdende Aufgabenstellungen in Ada-95 implementieren. Besonderer Wert wird dabei auf die Einhaltung eines Styleguides und eine gute Kommentierung gelegt. Ergänzend zum Programmierkurs wird ein freiwilliger Stützkurs für Studienanfänger ohne Vorkenntnisse im Programmieren angeboten, der diesen im ersten Semester die notwendigen Grundkenntnisse des Programmierens vermitteln soll.

Das Softwarepraktikum beginnt in der Mitte des dritten Semesters und dauert bis zum Ende der Vorlesungszeit des vierten Semesters. Im Rahmen dieser Veranstaltung führen die Studenten – wiederum in Dreiergruppen – ein komplettes Softwareprojekt durch, von der Analyse über Spezifikation, Entwurf, Implementierung in der Programmiersprache Java, Test und Abnahme. Sowohl hier als auch im Programmierkurs ist der Schein Voraussetzung für das Vordiplom.

In der Vorlesung Programmentwicklung werden den Studenten im dritten Semester die Grundlagen der objektorientierten Programmierung vermittelt.

Neben einer Einführung in Java wird dabei u.a. auf Design Patterns, Architektur und Test von OO-Programmen sowie die UML eingegangen. Die Bearbeitung der Java-Implementierungsaufgabe für den – nicht zwingend für das Vordiplom erforderlichen – Schein kann individuell oder in Kleingruppen erfolgen.

Begleitend zu diesen praktischen Lehrveranstaltungen vermitteln die Vorlesungen *Einführung in die Informatik I & II* die theoretischen Grundlagen des Programmierens.

3 Beschreibung des Experiments

3.1 Ziele des Experiments

Mit Hilfe des Experiments sollten die folgenden Fragen untersucht werden:

- Welchen Stand haben die Programmierfähigkeiten der Studenten zu Beginn des vierten Semesters (also nach dem Programmierkurs und der Vorlesung Programmentwicklung, aber vor dem Implementierungsteil des Softwarepraktikums) allgemein und im Vergleich miteinander?
- Wie entwickeln sich diese Fähigkeiten bis zum Ende des vierten Semesters nach Abschluss des Softwarepraktikums?

Diese Fragen sollten insbesondere unter dem Gesichtspunkt der unterschiedlichen Vorkenntnisse bei Studienbeginn untersucht werden. Da einige Studenten während des Softwarepraktikums ihr Studium abbrechen, sollte auch untersucht werden, ob ein Zusammenhang zwischen Abbruchwahrscheinlichkeit und vorhandenen Programmierkenntnissen bei Studienbeginn besteht.

Während die Beantwortung der ersten beiden Fragen Aufschluss über den Beitrag verschiedener Lehrveranstaltungsformen zur Vermittlung von Programmierfähigkeiten im Rahmen des Studiums geben soll, könnte zusätzliches Wissen über die Ursache von Studienabbrüchen dabei helfen, die bei Informatikstudiengängen relativ hohe Zahl von Studienabbrechern zu reduzieren.

3.2 Aufbau des Experiments

Um die Programmierfähigkeiten der Studenten vor und nach dem Softwarepraktikum messen zu können, fand das Experiment in zwei Teilen statt, die ab hier mit Teil 1 und Teil 2 bezeichnet werden:

- Teil 1 wurde zu Beginn des vierten Semesters im April und damit nach Abschluss des Spezifikationsteils, aber vor Beginn der Implementierungsarbeiten im Softwarepraktikum durchgeführt.
- Teil 2 wurde nach Ende der Vorlesungszeit des vierten Semesters im Juli und damit nach dem vollständigen Abschluss des Softwarepraktikums durchgeführt.

Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

Bei beiden Teilen ging es um die Implementierung einer vorgegebenen Aufgabenstellung in der Programmiersprache Java mit Hilfe der IDE Eclipse. Um in dem gegebenen Zeitraum von sieben Stunden eine etwas umfangreichere Aufgabenstellung bearbeiten zu können, standen den Teilnehmern bei beiden Teilen einige Java-Klassen mit bereits fertig implementierten Basisfunktionalitäten wie z.B. Schreib- und Leseoperationen für Dateien zur Verfügung. Nach einer kurzen Präsentation, bei der den Studenten die allgemeinen Rahmenbedingungen erläutert wurden, fand die individuelle Bearbeitung der Aufgabenstellung in einem Computerpool statt, wobei jedem Teilnehmer ein eigener Rechner zur Verfügung stand. Die Rechner liefen unter Linux, die Entwicklungsumgebung Eclipse mit den vorgegebenen Klassen und die Beschreibung der Aufgabenstellung waren vorinstalliert. Ein Internetzugang war vorhanden, und die Nutzung des Internets zur Recherche war ebenso gestattet wie das Mitbringen von Programmierbüchern; lediglich der Austausch zwischen den Studenten war verboten.

Die erfolgreiche Teilnahme, nachgewiesen durch eine Unterschrift nach Beendigung jedes Teils, war eine Voraussetzung für den Erhalt des Softwarepraktikum-Pflichtscheins; falls nach Ablauf von sieben Stunden absehbar war, dass dieser Teil des Experiments nicht erfolgreich abgeschlossen werden würde, war ein Abbruch ohne Verlust des Scheins aber zulässig. Erfolg bedeutete dabei jeweils das Bestehen eines vorgegebenen Abnahmetests. Dieser stand den Teilnehmern während des Experiments zur Verfügung; sie konnten also jederzeit selbst feststellen, ob ihre Lösung den Abnahmekriterien genügte. Nach Abschluss jedes Teils musste von jedem Studenten unabhängig vom Erfolg ein Fragebogen ausgefüllt werden. Zur Bewertung der Qualität der erfolgreichen Abgaben wurden nach Ende des Experiments weitere Tests durchgeführt, die den Teilnehmern nicht im Voraus bekannt waren.

Die Aufgabe bestand bei Teil 1 aus dem Einlesen einer Datei mit Datensätzen und der Ausgabe der sortierten und umformatierten Datensätze. Die Klassen mit Lese- und Schreiboperationen zum Einlesen und Ausgeben der Dateien waren vorgegeben. Bei Teil 2 musste eine Musterlösung aus Teil 1 um eine Funktion zur Filterung der Datensätze anhand von Kriterien, die aus einer weiteren Datei eingelesen werden mussten, erweitert werden. Außerdem wurden die Teilnehmer in vier Gruppen eingeteilt, die unterschiedliche Kombinationen von Begleitdokumenten erhielten (Algorithmusbeschreibung mit Klassendiagramm der vorgegebenen Klassen und Ablaufschilderung einerseits sowie aus zusätzlich eingefügten Javadoc-Kommentaren generierte HTML-Dateien andererseits). Ziel der Gruppenaufteilung war es, mögliche Auswirkungen vorhandener Dokumentation auf die zur Lösung der Aufgabe benötigte Zeit und die Qualität der erzeugten Programme zu untersuchen.

4 Resultate des Experiments

4.1 Teil 1

Teilnehmer

Insgesamt nahmen 67 Studenten an Teil 1 des Experiments teil, von denen allerdings 15 Sonderfälle aus verschiedenen Gründen bei der Auswertung nicht berücksichtigt werden konnten. Gründe dafür waren u.a. die Teilnahme an einem Ersatztermin unter nicht kontrollierten Bedingungen wegen Krankheit, Täuschungsversuchen, nicht erfolgter Abgabe des Fragebogens oder unerlaubtem Abbrechen des Experiments. Da zumindest die Teilnehmer aus den letzten drei Gruppen eine unterdurchschnittliche Leistung aufwiesen, sind die Leistungsdurchschnitte und -quantile tendenziell eher schlechter als im Folgenden angegeben. Für die Auswertung verbleiben damit 52 von 67 Studenten, was einer Quote von 78% entspricht.

Coderesultate

Von den 52 betrachteten Studenten haben 34 Studenten den Abnahmetest von Teil 1 bestanden, das entspricht einer Quote von 65%. Demnach haben es 18 bzw. 35% der Studenten – also etwas mehr als ein Drittel – nicht geschafft, die gestellte Aufgabe innerhalb der zur Verfügung stehenden Zeit von sieben Stunden so weit zu lösen, dass zumindest der Abnahmetest bestanden wurde. Drei Teilnehmer haben überhaupt keinen Programmcode geschrieben, sondern lediglich leere Klassen mit einer main-Methode generiert. Von den 18 nicht erfolgreichen Abgaben waren bei Experimentabbruch neun nicht kompilierbar, enthielten also syntaktische Fehler.

Zur besseren Einschätzung des Aufwands wurde die gestellte Aufgabe im Vorfeld von zwei Mitarbeitern des Lehrstuhls gelöst. Ein Mitarbeiter mit sehr guten Java-Kenntnissen benötigte für eine etwas komplexere Version der Aufgabenstellung 3:10 Stunden, woraufhin die Aufgabenstellung durch Verzicht auf einige Sonderfälle vereinfacht wurde. Der andere Mitarbeiter mit mittleren Java-Kenntnissen benötigte für die Bearbeitung der späteren Aufgabenstellung 3:00 Stunden.

Die erfolgreichen Teilnehmer benötigten zwischen 2:10 Stunden und 8:40 Stunden für die erfolgreiche Bearbeitung der Aufgabe (siehe Abb. 1). Der Median lag bei 4:25 und der Mittelwert bei 4:42 Stunden. Berücksichtigt man alle 52 Studenten, liegt der Median bei 5:10 Stunden. Aufgrund der großen Zahl nicht erfolgreicher Teilnehmer können die *LS50*- und *LS25*-Verhältnisse nach [Pre01] nicht sinnvoll gebildet werden.

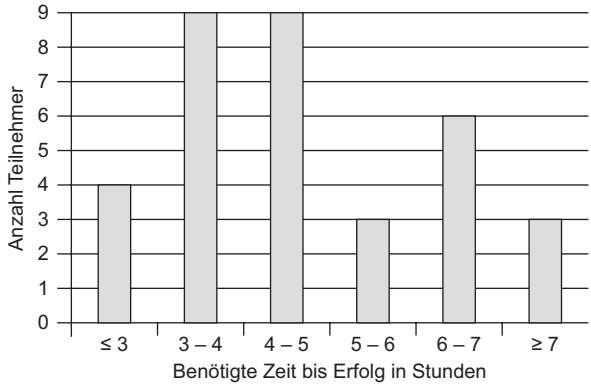

Abb. 1 *Benötigte Zeit der 34 erfolgreichen Teilnehmer in Std. / Anzahl Teilnehmer*

Zur differenzierten Bewertung der Qualität der Abgaben wurden acht zusätzliche Testfälle definiert und jede erfolgreiche Abgabe damit getestet. Die 34 erfolgreichen Abgaben ergaben in 0 – 8 Testfällen korrekte Ergebnisse; der Median lag bei 6 und der Mittelwert bei 5,12. Ursachen für die falschen Ergebnisse waren teils Programmierfehler, teils aber auch Täuschungsversuche wie die statische Festlegung bestimmter Werte im Programm, die eigentlich dynamisch hätten eingelesen werden sollen. Das Streudiagramm in Abbildung 2 zeigt für jede Abgabe die benötigte Zeit bis zum erfolgreichen Abschluss des Experiments in Stunden und die jeweilige Anzahl der Testfälle mit korrektem Ergebnis. Die Korrelation zwischen der benötigten Zeit und der Anzahl der Testfälle mit korrektem Ergebnis ist nach Pearson mit einem Wert von -0,455 auf dem Niveau von 0,01 (2-seitig) signifikant. Die schnelleren Teilnehmer produzierten also zumindest tendenziell auch eine bessere Codequalität.

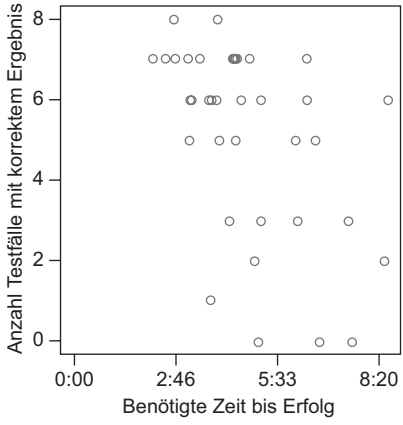

Abb. 2 *Bis zum Erfolg benötigte Zeit in Std. / Anzahl Testfälle mit korrektem Ergebnis*

Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

Fragebogen

In dem nach Beendigung von Teil 1 auszufüllenden Fragebogen wurde u.a. gefragt, wie die Studenten Programmieren gelernt haben; Mehrfachnennungen waren dabei zulässig. Demnach haben sich 31 Studenten das Programmieren weitgehend selbst beigebracht. Jeweils 15 Studenten gaben an, in Schule oder Hochschule Programmieren gelernt zu haben, bei 4 Studenten traf dies für eine Beschäftigung zu, der sie nachgegangen sind.

Zwischen dem Vorhandensein von Programmierkenntnissen vor dem Studium und dem Erfolg bei Teil 1 des Experiments besteht eine schwach positive Korrelation, die nach Pearson mit einem Wert von 0,308 auf dem Niveau von 0,05 (2-seitig) signifikant ist. Tendenziell waren Studenten, die bereits vor dem Studium Programmiererfahrungen gesammelt hatten, also erfolgreicher bei Teil 1. Eine ähnliche schwach positive Korrelation besteht zwischen der Programmiererfahrung in bereits geschriebenen LOC und dem Erfolg bei Teil 1; diese ist nach Pearson mit einem Wert von 0,286 auf dem Niveau von 0,05 (2-seitig) signifikant. Zwischen dem Vorhandensein von Programmiererfahrungen vor dem Studium und den geschriebenen LOC besteht keine statistisch signifikante Korrelation, ebenso wenig wie zwischen den Programmiererfahrungen und der bei Teil 1 benötigten Zeit bis zum Erfolg.

Weiterhin wurde im Fragebogen nach der Anzahl bisher im Studium erworbener Scheine gefragt. Dabei gab es einen Unterschied zwischen den bei Teil 1 erfolgreichen Studenten und den nicht erfolgreichen: Letztere hatten im Mittel 8,72 Scheine erworben (Standardabweichung 1,406), Erstere hingegen im Mittel 9,44 Scheine (Standardabweichung 1,795). Die Differenz von 0,72 ist zwar nicht statistisch signifikant, aber praktisch bedeutsam, weil die absolute Minimalanzahl von Scheinen zum Erreichen des vierten Semesters fünf beträgt, die sinnvolle Minimalzahl sieben. Die bei Teil 1 erfolgreichen Studenten haben damit 30% mehr Scheine über das sinnvolle Mindestmaß hinaus freiwillig erworben als die nicht erfolgreichen und scheinen damit auch über das Programmieren hinaus im Studium tendenziell etwas engagierter zu sein.

4.2 Teil 2

Teilnehmer

Die Teilnahme und der Erfolg der 52 betrachteten Studenten ergab sich bei Teil 2 wie folgt:

- 62% (32 Studenten) haben bei beiden Teilen erfolgreich teilgenommen.
- 4% (2 Studenten) haben bei Teil 1 erfolgreich teilgenommen und bei Teil 2 nicht teilgenommen.

Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

- 13% (7 Studenten) haben bei Teil 1 ohne Erfolg und bei Teil 2 erfolgreich teilgenommen.
- 6% (3 Studenten) haben bei beiden Teilen ohne Erfolg teilgenommen.
- 15% (8 Studenten) haben bei Teil 1 ohne Erfolg teilgenommen und bei Teil 2 nicht teilgenommen.
- Es gab keine Studenten, die bei Teil 1 erfolgreich teilgenommen haben und bei Teil 2 ohne Erfolg teilgenommen haben.

Da die Teilnahme am gesamten Experiment Voraussetzung für den Erhalt eines Pflichtscheins für das Vordiplom war, kann davon ausgegangen werden, dass die zehn Studenten, die nicht an Teil 2 teilgenommen haben, das Studium abgebrochen haben. Drei davon haben zwischen den beiden Experimentteilen aufgrund des Nichtbestehens von Prüfungen (Mathematik bzw. Praktische Informatik) endgültig den Prüfungsanspruch verloren; die Ursache für den Studienabbruch der anderen sieben ist unbekannt, könnte aber mit Schwierigkeiten im SoPra zusammenhängen.

Um die Leistung der mit verschiedenen Begleitdokumenten ausgestatteten Gruppen (siehe Abschnitt 3.2) vergleichen zu können, wurde versucht, die teilnehmenden Studenten möglichst homogen aufzuteilen, wobei als Zuteilungskriterium die bis zum Erfolg bei Teil 1 benötigte Zeit verwendet wurde. Die bei Teil 1 nicht erfolgreichen Studenten wurden ebenfalls gleichmäßig auf die Gruppen verteilt. Aufgrund von Studienabbrüchen und Krankmeldungen konnte aber keine vollständig homogene Verteilung erzielt werden.

Coderesultate

Von den 42 berücksichtigten Teilnehmern an Teil 2 haben 39 den Abnahmetest bestanden, was einer Quote von 93% entspricht. Von den drei nicht erfolgreichen Abgaben waren zwei gegenüber den vorgegebenen Klassen in höchstens 20 Zeilen verändert, die dritte war nicht kompilierbar.

Auch die Aufgabe des zweiten Teils war im Vorfeld von zwei Mitarbeitern des Lehrstuhls bearbeitet worden, wobei zur Lösung 1:20 bzw. 1:00 Stunden (mittlere bzw. sehr gute Java-Kenntnisse) benötigt wurden. Bei den Studenten lag der Median der bis zum Erfolg benötigten Zeit in allen Gruppen mit verschiedener Begleitdokumentation (siehe Abschnitt 3.2) unter 2:00 Stunden; bis auf zwei Ausreißer haben alle erfolgreichen Teilnehmer weniger als vier Stunden zur Lösung der Aufgabe benötigt. Die Unterschiede zwischen den einzelnen Gruppen hinsichtlich der benötigten Zeit sind statistisch nicht signifikant. Da nur drei von 42 Studenten diesen Teil nicht erfolgreich abgeschlossen haben, können zum Vergleich der Programmierleistung die von [Pre01] empfohlenen Maße *LS50* und *LS25* herangezogen werden. *LS* steht dabei für Langsam-Schnell-Verhältnis und zeigt anschaulich, wie viel Mal länger eine mittlere »langsame« Versuchsperson im Verhältnis zur mittleren »schnelleren« Versuchsperson benötigt:

- *LS50* ist der Quotient von 75- und 25-Percentil (q_{75}/q_{25}) und entspricht damit dem Verhältnis der langsameren zur schnelleren Hälfte.
- *LS25* ist der Quotient von 87,5- und 12,5-Percentil ($q_{87,5}/q_{12,5}$) und entspricht damit dem Verhältnis des langsamsten zum schnellsten Viertel.

Mit q_{75} = 2:00 Stunden und q_{25} = 1:15 Stunden ergibt sich *LS50* = 120 min / 75 min = 1,6. Die langsamere Hälfte brauchte demnach im Mittel etwas mehr als anderthalb mal so lange wie die schnellere Hälfte. Aus $q_{87,5}$ = 2:52 Stunden und $q_{12,5}$ = 1:04 Stunden ergibt sich *LS25* = 172 min / 64 min = 2,7. Das langsamste Viertel brauchte also im Mittel etwas mehr als zweieinhalb mal so lang wie das schnellste Viertel.

Zur Bewertung der Qualität der Abgaben wurden 13 Testfälle definiert und die erfolgreichen Abgaben damit getestet. Im Unterschied zu den Testergebnissen bei Teil 1 war die Qualität der Abgaben deutlich homogener:

- Der Median von Testfällen mit korrektem Ergebnis lag in Gruppe 1 bei 12 und in allen anderen Gruppen bei 13 Testfällen.
- Bis auf zwei Ausreißer bestanden alle erfolgreichen Abgaben mindestens acht Testfälle.

Die Korrelation zwischen bis zum Erfolg benötigter Zeit und bestandenen Testfällen war im Unterschied zu Teil 1 nicht statistisch signifikant.

Fragebogen

Mit dem Fragebogen nach Teil 2 sollte untersucht werden, wie sich die Programmierkenntnisse der Studenten während des Softwarepraktikums entwickelt haben. Insbesondere stand dabei die Frage im Mittelpunkt, ob sich vor dem SoPra bestehende Unterschiede während des SoPras verkleinern oder vergrößern würden.

Zwischen der Anzahl vor dem SoPra geschriebener LOC und der Anzahl LOC, die während des SoPras geschrieben wurden, gibt es eine Korrelation, die nach Pearson mit einem Wert von 0,607 auf dem Niveau von 0,01 (2-seitig) statistisch signifikant ist. Die Teilnehmer, die bereits vor dem SoPra mehr Programmiererfahrung gesammelt hatten, haben also auch tendenziell während des SoPras innerhalb des Teams einen größeren Teil der Programmieraufgaben übernommen. Zwischen der Anzahl während des SoPras geschriebener LOC und der bei Teil 2 bis zum Erfolg benötigten Zeit gibt es aber keine statistisch signifikante Korrelation, ebenso wenig wie zwischen der Anzahl vor dem SoPra geschriebener LOC und der bei Teil 2 bis zum Erfolg benötigten Zeit.

Zusätzlich wurde im Fragebogen nach Verwendung und Einschätzung der Nützlichkeit der Algorithmusdokumentation und der aus zusätzlichen Javadoc-Kommentaren generierten HTML-Seiten gefragt. Demnach haben nur zwei Drittel der Studenten eine vorhandene Algorithmusdokumentation benutzt. Von diesen

zwei Dritteln empfanden wiederum zwei Drittel und damit 44% der Ausgangsmenge die Algorithmusdokumentation als nützlich. Zur Verfügung gestelltes Javadoc wurden ebenfalls von zwei Dritteln genutzt, von diesen allerdings nur zur Hälfte und damit von 33% der Ausgangsmenge als nützlich bewertet. Von den Teilnehmern, denen keine Algorithmusdokumentation zur Verfügung stand, gaben 45% an, dass eine solche Dokumentation die Aufgabe vermutlich erleichtert hätte. Bei Javadoc waren 35% dieser Ansicht. Setzt man voraus, dass Studenten, die eine zur Verfügung stehende Dokumentation nicht benutzten, diese implizit als nicht nützlich bewerteten, ergibt sich eine nahezu perfekte Übereinstimmung der Beurteilung der Nützlichkeit von Dokumentationstypen unabhängig von deren tatsächlichem Vorliegen in der konkreten Situation. Diese Übereinstimmung könnte möglicherweise auf die Existenz grundlegend verschiedener und individuell bereits im Studium gefestigter Herangehensweisen an Programmierprobleme hindeuten. Auch wenn in diesem Experiment keine statistisch signifikante Korrelation zwischen benutzter und als nützlich oder nicht nützlich eingestufter Dokumentation und Erfolg aufgetreten ist, erscheint eine weitergehende Forschung in diesem Bereich dennoch lohnend.

5 Schwachpunkte im Experimentdesign

Die folgenden vier Punkte haben sich bei Durchführung und Auswertung des Experiments als mögliche Schwachstellen gezeigt, die bei einer Wiederholung des Experiments genauer betrachtet und evtl. anders gehandhabt werden sollten:

5.1 Code&Fix anstelle von Software Engineering

Die Rahmenbedingungen des Experiments führten dazu, dass die reine Programmierleistung im Sinne der Implementierung von Funktionalität im Mittelpunkt stand: Die im Studiengang Softwaretechnik vermittelten Grundsätze guten Software Engineering wie die Erstellung einer Spezifikation, der gründliche Entwurf einer tragfähigen Architektur, sorgfältige Implementierung mit guter Kommentierung sowie planvolles Testen spielten keine wesentliche Rolle. Im Gegenteil begünstigte der den Teilnehmern ständig zur Verfügung stehende Abnahmetest – bei dem als einziges Erfolgskriterium eine bestimmte Funktionalität verlangt wurde und die Programmqualität nicht bewertet wurde – einen Code&Fix-Ansatz. Allerdings wurde bei dem Experiment auch die Anzahl fehlgeschlagener Abnahmetests mitprotokolliert, die von den Teilnehmern selbst durchgeführt worden waren: danach war mit 13 von 52 Teilnehmern bei einem Viertel bereits der erste durchgeführte Abnahmetest erfolgreich, mehr als drei Viertel haben nur acht oder weniger fehlgeschlagene Abnahmetests durchgeführt. Lediglich bei zwei Teilnehmern ist mit 41 und 44 fehlgeschlagenen Abnahmetests von einem extremen Code&Fix-Ansatz auszugehen.

Die Vernachlässigung der Programmqualität und der Nicht-Implementierungsaspekte der Softwareentwicklung bei einem solchen Experiment führen sicherlich dazu, dass die Ergebnisse keine direkten Rückschlüsse auf die Gesamtleistung der Teilnehmer als Softwareentwickler zulassen. Andererseits sollte jeder Entwickler das grundlegende Handwerkszeug des Programmierens beherrschen, auch wenn er oder sie später hauptsächlich spezifiziert oder testet. Während also ein gutes Abschneiden bei einem derartigen Experiment nicht zwangsläufig einen guten Software-Ingenieur ausweist, deutet ein stark unterdurchschnittliches Ergebnis auf Lücken in der bisherigen Ausbildung hin.

5.2 Anonymisierung

Um datenschutzrechtliche Bedenken von vornherein auszuschließen, wurden die Ergebnisse vollständig anonymisiert erhoben. Zu diesem Zweck wurde von jedem Studenten bei der Präsentation der Aufgabenstellung von Teil 1 ein Linux-Benutzername in der Form exp*Zufallsnummer* mitsamt zugehörigem Benutzerpasswort zufällig gezogen. Die Programmabgaben und die ausgefüllten Fragebögen konnten damit nur einem bestimmten Benutzernamen bzw. der Zufallsnummer zugeordnet werden. Die Unterschrift unter eine Teilnehmerliste zum Zweck der Überprüfung der Mitwirkung für den Schein erfolgte bei Verlassen des Poolraums ohne Zuordnung zu einer speziellen Abgabe.

Diese Vorgehensweise sicherte zwar weitgehende Anonymität zu (abgesehen allenfalls von Krankheitsfällen), sie brachte aber auch eine Reihe von Problemen mit sich. So konnten sich viele Studenten bei Teil 2 trotz im Vorfeld erfolgter Hinweise nicht an ihre Zufallsnummer erinnern, was die Auswertung erschwerte. Ohne die Zuordnung der Zufallsnummern zu Namen war es auch nicht möglich, Studenten, die das Studium aufgrund des Nichtbestehens von Prüfungen aufgeben mussten, auf Experimentteilnehmer abzubilden. Rückblickend erscheinen eine im Vorfeld garantierte, strikte Geheimhaltung von Experimentdaten und eine anonymisierte Veröffentlichung der Ergebnisse unter Verzicht auf Maßnahmen zur vollständigen Anonymisierung die bessere Alternative zu sein.

5.3 Effekte zusätzlicher Dokumentation als Experimentvariable

Durch die Aufteilung in vier Gruppen bei Teil 2 des Experiments sollten die Effekte zusätzlich zur Verfügung stehender Dokumentation auf die Programmierproduktivität bei einer Wartungsaufgabe untersucht werden. Dies ist unter anderem wegen der zu kleinen Fallgruppen nicht gelungen. Bei einer Teilnehmermenge von ca. 40 sollte maximal ein zusätzliches Unterscheidungsmerkmal eingeführt werden, damit die interessierenden Fallgruppen groß genug sind. Hinzu kommt, dass die Einführung zusätzlicher Variabilität in nur einem Experimentteil den Vergleich erschwert.

5.4 Umfang und Art der Aufgabenstellung

Der Umfang der Aufgabenstellung bei Teil 1 des Experiments war zu groß gewählt. Im Nachhinein lässt sich zumindest für Teil 2 zeigen, dass das schnellste Viertel der Studenten im Mittel etwa so lange bis zum Erfolg gebraucht hat wie die Institutsmitarbeiter, die die Aufgabe im Vorfeld bearbeitet hatten. Wegen *LS25* = 2,7 braucht das langsamste Viertel der Studenten im Mittel also mehr als zweieinhalb mal so lange wie die Institutsmitarbeiter. Um mindestens 7/8 der Studenten die Möglichkeit zu geben, das Experiment erfolgreich zu beenden – was gleichzeitig auch die Aussagekraft der gewonnenen Werte erhöht –, sollte bei der Aufgabenstellung also darauf geachtet werden, dass Institutsmitarbeiter, die die Aufgabe im Vorfeld lösen, nicht mehr als 1/4 bis 1/3 der den Studenten maximal zur Verfügung stehenden Zeit dafür benötigen (mittlere Kenntnisse der Institutsmitarbeiter in der verwendeten Programmiersprache vorausgesetzt). Darüber hinaus wäre überlegenswert, den Erfolg bei einem derartigen Programmierexperiment nicht nur an der Lösung einer einzigen Aufgabenstellung festzumachen, sondern stattdessen mehrere, nach Schwierigkeitsgrad abgestufte Aufgaben zu stellen. Eine solche feinere Skala für den Erfolg hätte den Vorteil, auch schwächeren Teilnehmern ein erreichbares Ziel zu bieten, würde allerdings auch die Verwendung der benötigten Zeit als einfachen Vergleichsmaßstab erschweren.

6 Ergebnis und Ausblick

Das durchgeführte Experiment hat gezeigt, dass die Leistungsunterschiede der Studenten beim Programmieren vor der Durchführung des Softwarepraktikums erheblich sind. Etwas mehr als ein Drittel der Studenten war zu Beginn des vierten Semesters trotz der Lehrveranstaltungen in den ersten drei Semestern nicht in der Lage, die gestellte Java-Programmieraufgabe zu lösen; ein Sechstel der Studenten beherrschte nicht einmal die grundlegende Java-Syntax.

Erst die Implementierungsarbeit im SoPra scheint zu einer Angleichung der Programmierproduktivität zu führen: Nach dem SoPra sind nur noch 7% der dann noch verbleibenden Studenten nicht in der Lage, die gestellte Java-Programmieraufgabe zu lösen. Der *LS25*-Wert von 2,7 bei Teil 2 liegt unterhalb des in der Literatur angegebenen Verhältnisses von 4:1, so dass es sich bei den Studenten nach dem SoPra um eine relativ homogene Gruppe zu handeln scheint. Auch wenn sich die vor dem SoPra vorhandene Programmiererfahrung auf die im SoPra geleistete Programmierarbeit auswirkt, scheint sie auf die Programmierproduktivität nach dem SoPra keinen wesentlichen Einfluss mehr auszuüben.

Allerdings scheint die Aneignung der notwendigen Programmierpraxis im SoPra parallel zur Bearbeitung der eigentlichen Aufgabe des SoPras einige Studenten vor größere Probleme zu stellen. Von den bei Teil 1 des Experiments nicht erfolgreichen Teilnehmern haben 44% das Studium zwischen Teil 1 und Teil 2

abgebrochen, während diese Quote bei den erfolgreichen Teilnehmern von Teil 1 nur 6% beträgt. Nicht bestandene Prüfungen können dafür nur teilweise die Ursache sein, so dass die Vermutung naheliegt, dass Schwierigkeiten mit der Programmierung im SoPra mit zum Studienabbruch beitragen. Unter anderem um diesen Schwierigkeiten vorzubeugen, werden jetzt im Studiengang Softwaretechnik begleitend zur Vorlesung Programm-entwicklung im dritten Semester betreute Programmierübungen in Java angeboten – damit alle Studenten schon vor dem SoPra eine ausreichende Programmierpraxis haben und im SoPra nicht plötzlich vor unlösbaren Schwierigkeiten stehen.

Literatur

[Dic81] T. E. Dickey, Programmer variability, Proc. IEEE 69, 7 (Jul. 1981), 844 – 846.

[Lud97] Ludewig, J. Der Modellstudiengang Softwaretechnik. in P. Forbrig, G. Riedewald (Hrsg.): SEUH'97 (Software Engineering im Unterricht der Hochschulen). Berichte des German Chapter of the ACM, Band 48, Teubner, Stuttgart, 9 – 23.

[McC01] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. SIGCSE Bull. 33, 4 (Dec. 2001), 125 – 180.

[Pre01] Lutz Prechelt. Kontrollierte Experimente in der Softwaretechnik. Springer Verlag, 2001.

[Sac68] Sackman, H., Erikson, W. J., and Grant, E. E. Exploratory experimental studies comparing online and offline programming performance. Commun. ACM 11, 1 (Jan. 1968), 3 – 11.

[SoPra] Webseite der Abteilung SE der Universität Stuttgart zum SoPra-Experiment 2005: *http://www.iste.uni-stuttgart.de/se/teaching/courses/sopra/experiment/index.html*