# Der frühe Tester fängt den Bug

Axel Böttcher, Veronika Thurner, Daniela Zehetmeier*

Fakultät für Informatik und Mathematik – Hochschule München
Lothstraße 64 – 80335 München
a.boettcher@cs.hm.edu, thurner@hm.edu, daniela.zehetmeier@hm.edu

**Abstract:** Das Thema *Testen* spielt derzeit in der Grundausbildung der Softwareentwicklung oft nur eine untergeordnete Rolle. Entsprechend entwickeln die Studierenden wenig Erfahrung im Erstellen von Unit-Tests und kaum Bewusstsein für deren Bedeutung. Durch den Ansatz "Objects first, Tests second" rückt die Thematik des automatisierten Testens in den Vordergrund der Ausbildung in Softwareentwicklung.

## 1 Motivation und Grundproblem

In vielen Informatik-nahen Studiengängen nimmt die Softwareentwicklung einen hohen Anteil der Grundausbildung ein. Dabei werden meist allgemeine Kernkonzepte wie Variablen und Kontrollstrukturen behandelt, sowie Objektorientierung und weiterführende Themen wie Vererbung, Collections und Ausnahmen. Das Thema *Testen* kommt dabei – wenn überhaupt – erst gegen Ende der Grundausbildung in Softwareentwicklung zur Sprache. Auch die Lehrbuchlandschaft spiegelt dieses Phänomen wider. Nach unseren Recherchen gibt es auf dem Markt der Lehrbücher zur (objektorientierten) Softwareentwicklung kaum Literatur, die das Thema *Testen* von Beginn an in den Vordergrund stellt.[1]

In der Folge haben viele Studierenden das Testen in der Grundausbildung entweder gar nicht kennen gelernt, oder nur als eines von vielen weiterführenden Themen meist am Ende des zweiten Semesters. Entsprechend gering ist die gewonnene praktische Erfahrung. Ferner messen die Studierenden dem Testen oft nur eine untergeordnete Bedeutung bei – denn schließlich ging die Grundausbildung ja über weite Teile auch *ohne* Testen.

## 2 Zielsetzung und Vorgehensweise

Um diesem Problem entgegenzusteuern entwickeln wir einen Lehransatz, der Unit-Tests als zentralen Inhalt bereits in der Grundausbildung zur Softwareentwicklung verankert. Unser Ziel ist dabei, den Studierenden frühzeitig praktische Erfahrung im Testen zu vermitteln, den Nutzen des Testens für die Lösungsqualität erlebbar zu machen und so das Unit-Testen zu einer Selbstverständlichkeit werden zu lassen.

Dazu haben wir im Wintersemester 2013/14 die Veranstaltung "Softwareentwicklung 1" nach dem Ansatz "Objects first, Tests second" durchgeführt. Das heißt, wir haben zunächst die Grundidee und die Kernkonzepte der Objektorientierung eingeführt, sowohl anschaulich als auch implementatorisch (in Java). Unmittelbar danach führten wir die Grundlagen

*Gefördert durch das BMBF Förderkennzeichen 01PL11025 (Projekt "Für die Zukunft gerüstet"), im Programm "Qualitätspakt Lehre"

[1] Ausnahme: Jeff Langr. *Agile Java – Crafting Code with Test-Driven Development*. Pearson Education, 2005.

von Unit-Tests ein. Diese wurden dann über die Veranstaltung hinweg sukzessive weiter ausgebaut und erweitert, jeweils passend zu den gerade behandelten Konstrukten. So wurde z.B. in der Lerneinheit zu *Arrays* die Methode `assertArrayEquals()` eingeführt.

Im Rahmen des Praktikums mussten die Studierenden von Beginn an ihre Lösungen so gestalten, dass sie testbar sind, sowie entsprechende Unit-Tests mit entwickeln. Ergänzend gaben die Lehrenden bei einigen Aufgaben die zu erfüllenden Tests als Teil der Anforderungsdefinition mit vor, hinführend auf die Praxis der testgetriebenen Entwicklung.

## 3 Erfahrungsbericht und Lessons Learned

Eine wesentliche Herausforderung des Ansatzes bestand darin, alle in der Lehrveranstaltung verwendeten Beispiele und Aufgaben so zu gestalten, dass sie konsequent testbar sind. Insbesondere müssen dabei die erstellten Methoden als Ergebnis einen Datenwert zurückliefern und nicht lediglich irgendwelche Informationen auf der Konsole ausgeben. Entsprechend wurde das Konstrukt `System.out.println()` nahezu bedeutungslos.

Diese Forderung schränkt die mögliche Begleitliteratur drastisch ein, da kaum ein anfängertaugliches Java-Lehrbuch ohne exzessive Nutzung der Konsolenausgabe auskommt. Diese wird überwiegend dazu verwendet, um die im Programm aktiven Objekte und deren Veränderung über die Zeit für den Lernenden sichtbar darzustellen.

Verzichtet man auf die testunfreundliche (und praxisferne) Verwendung der Konsolenausgabe, fällt dieser Einblick in den Programmzustand weg, und damit ein Teil der Anschaulichkeit. Als Ersatz wurden GUIs zu den Beispielen entwickelt, die die Objekte darstellen.

Insgesamt haben die Studierenden am Ende des ersten Semesters das Erstellen von Unit-Tests als zentralen Bestandteil jeglicher Softwareentwicklung akzeptiert und verinnerlicht. Dabei beherrschen Sie die behandelten Grundtechniken im Wesentlichen sicher.

Ein weiterer, indirekter Nutzen der frühzeitigen Einführung von Unit-Tests zeigte sich bei der automatisierten Bewertung der studentischen Lösungen im Praktikum. Diese basiert auf Unit-Tests, die die Lehrenden zu den einzelnen Übungsaufgaben entwickelt haben. Da die Studierenden mit der dahinter liegenden Technik vertraut sind war der automatisierte Bewertungsmechanismus für sie sofort nachvollziehbar und wurde gut akzeptiert.

## 4 Weiterführende kritische Fragen für den nächsten Durchlauf

Obwohl der "Objects first, Tests second"-Ansatz bereits beim ersten Versuch recht erfolgreich war, bieten einige Punkte beim nächsten Durchlauf Raum für Verbesserungen im didaktischen Ansatz. Die folgenden Fragestellungen spiegeln dies wider:

- Zu welchem Zeitpunkt und wie intensiv behandelt man die Frage, wieviele (und welche) Tests geschrieben werden müssen?
- Wie schafft die Lehrperson im Praktikum eine ausgewogene Balance zwischen dem Aufwand für die gezielte Verwendung neuer Konstrukte, die Konzeption der Logik und dem zugehörigen Testaufwand?
- Auf welche Weise werden allgemein Unit-Tests an sich qualitätsgesichert, und wie im Speziellen die von den Studierenden im Rahmen ihrer Aufgaben erstellten Tests?

Diese werden wir im nächsten Durchlauf der Veranstaltung gezielt angehen.