

Geleitwort

zur zweiten Auflage

Erinnern Sie sich an Ihre erste Programmiererfahrung? Ich meine nicht die Einzelheiten des Computers oder der Programmiersprache. Ich meine, wie haben Sie sich gefühlt?

Ich erinnere mich noch, wie ich ein paar Befehle aus dem Programmierhandbuch eintippte und darauf brannte, das Programm laufen zu sehen. Es war atemberaubend zu beobachten, wie der Code Leben annahm. Innerhalb weniger Stunden wuchs das Buchbeispiel zu einem, wie ich fand, beeindruckenden Programm. Ich hatte hier und dort Erweiterungen vorgenommen. Nach jeder Änderung startete ich das Programm, um zu sehen, was es tat. Am Abend führte ich alles meinen Eltern vor. Ein Blick in meine Augen ließ keinen Zweifel darüber offen, wie stolz ich wohl war.

Wie sich die Dinge verändert haben. Programmieren ist immer noch mein Ein und Alles. Dabei fällt mir hin und wieder jedoch auf, dass der ursprüngliche Spaß, weshalb wir alle einmal Programmierer geworden sind, auf der Strecke geblieben ist. Geht Ihnen das auch so? In diesen Momenten muss ich stets an meine erste Erfahrung denken. Warum kann Programmieren nicht immer so sein?

Zu einem dieser Momente stolperte ich über die in diesem Buch beschriebenen Techniken. Automatisierte Tests und fortlaufendes Refactoring, beides zusammen genommen, brachten mich dem Beginn meiner Programmierkarriere wieder ein Stückchen näher. Seitdem konnte ich mich meistens so verhalten, als wäre das Einzige, was ich tun müsste, ein paar Zeilen Code zu schreiben. Lassen Sie sich jedoch nicht täuschen. Diese Techniken funktionieren nicht nur für kleine Programme. Je größer der Umfang, desto wertvoller werden diese Techniken.

Ich war begeistert, als mich Johannes nach einem Geleitwort fragte. Dafür gibt es zwei Gründe:

Dieses Buch enthält das gebündelte Wissen, das sich eine Menge von Extreme-Programming-Pionieren gewünscht hätten, als sie vor nunmehr sieben Jahren mit testgetriebener Entwicklung losstiefelten. Wenn Sie diesem Weg folgen, werden Sie unweigerlich in Testprobleme laufen. Obwohl Sie zuerst Ihre Tests schreiben, werden Sie manchmal zum Stehen kommen, weil Sie nicht erkennen, wie Sie Ihren Code testen können. Das ist ganz natürlich. Tatsächlich ist dieser Punkt die perfekte Gelegenheit, einen Schritt zurückzutreten und einen Moment zu reflektieren ... oder dieses Buch in die Hand zu nehmen und nachzulesen, was uns denn Johannes dazu rät.

Ich wollte einmal ein Buch schreiben, ganz ähnlich diesem. Als ich aber Johannes' erste Buchkapitel zum Review las, konnte ich sehen, dass dieses Buch nicht nur erstklassig geschrieben war, sondern auch eine Reihe von Werkzeugen behandelte, die es erlauben, auch die dunkelsten Ecken mit Tests auszuleuchten, Code, der häufig einfach schwer testbar ist. Ich wünschte, ich hätte dieses Buch geschrieben. Deshalb ist der Beitrag eines Geleitwortes eine große Ehre.

Eine Gefahr besteht jedoch beim Lesen dieses Buches. Sie könnten den Eindruck bekommen, dass es nur um Techniken und Werkzeuge geht, während es in Wirklichkeit um Sie selbst geht.

Testinfirmierte Programmierer erzählen gern davon, wie sich ihre Beziehung zum Code durch die Tests verändert hat. Es gibt uns ein unglaubliches Gefühl von Vertrauen, wenn Hunderte von Tests auf Knopfdruck durchlaufen und unsere Software auf Herz und Nieren prüfen. Irgendwann werden auch Sie sich ganz sicher dabei ertappen, wie Sie Ihre Tests mehrere Male hintereinander ausführen, nur für den zusätzlichen Kick, dass alles prima läuft.

Frank Westphal

Extreme Programmer und Coach

Geleitwort

zur ersten Auflage

Als professioneller Softwareentwickler möchte ich Software so schnell wie möglich, so gut wie möglich und mit so wenig Stress wie möglich entwickeln. Automatisierte Unit Tests können helfen, nahe an dieses Ziel heranzukommen. Sie sind eine relativ kleine Investition, mit der das Vertrauen in den produzierten Code erst einmal aufgebaut und später beibehalten werden kann. Habe ich keine automatisierten Tests, bleiben mir nur manuelle Tests. Diese sind aber nicht automatisch wiederholbar und als Konsequenz nimmt der Stress zu, insbesondere, wenn die manuellen Tests unter Zeitdruck durchgeführt werden müssen, was natürlich meistens der Fall ist. Mit automatisierten Tests kann auf Knopfdruck jederzeit bestimmt werden, ob die letzte Änderung die Fitness der Software beeinträchtigt. Dies kann man heute, morgen oder irgendwann in der Zukunft tun, unabhängig davon, ob eine Deadline vor der Tür steht.

Das vorliegende Buch von Johannes Link und Peter Fröhlich ist eine sehr gute praktische Einführung in die Entwicklung mit automatisierten Unit Tests und dem Test-First-Vorgehen. Als Automatisierungsframework wird im Buch JUnit verwendet. Dieses kleine Framework vereinfacht zwar die Erstellung und Verwaltung von Tests, für die erfolgreiche Entwicklung mit Unit Tests braucht es aber mehr. Ein Entwickler muss mit verschiedenen Techniken vertraut sein, insbesondere, wenn Unit Tests im Kontext von Datenbanken oder verteilten Applikationen mit Applikationsservern erstellt werden müssen. Das Buch beleuchtet auch diese Problembereiche und ist deshalb ein sehr wertvoller Beitrag zum Thema automatisierte Unit Tests.

JUnit selber wurde auch mit automatisierten Unit Tests und Test-First, wie in diesem Buch beschrieben, entwickelt. Die Techniken wurden dabei oft unter erschwerten Bedingungen angewendet, wie im Kampf gegen den Jetlag oder bei Stromknappheit in Alphütten. Die Techniken haben sich trotzdem bewährt... Ich hoffe, dass Sie dank diesem Buch in der Zukunft die klassische Unit-Test-Kontrollfrage »Wo sind die Unit Tests?« immer öfter positiv beantworten werden können.

Erich Gamma

Mitautor von JUnit
Technischer Direktor, Object Technology International