Andreas Spillner · Tilo Linz

Das Standard-werk in 6. Auflage

Basiswissen Softwaretest

Aus- und Weiterbildung
zum **Certified Tester**

- **Foundation Level**
- **nach ISTQB®-Standard**

dpunkt.verlag

Über die Autoren

Andreas Spillner war bis 2017 Professor für Informatik an der Hochschule Bremen. Ab 1991 war er für über 10 Jahre Sprecher der Fachgruppe TAV »Test, Analyse und Verifikation von Software« der Gesellschaft für Informatik e.V. (GI), die er mit gegründet hat. Im »German Testing Board« e.V. war er von Beginn an bis zum Jahr 2009 engagiert und wurde danach zum Ehrenmitglied berufen. 2007 ist er zum Fellow der GI ernannt worden. Seine Arbeitsschwerpunkte liegen im Bereich Softwaretechnik, Qualitätssicherung und Testen. Andreas Spillner ist neben Ulrich Breymann Autor des Buches »Lean Testing für C++-Programmierer – Angemessen statt aufwendig testen« (dpunkt.verlag), das die Testverfahren der ISO-Norm 29119 und deren konkrete Umsetzung in die Programmiersprache C++ erörtert.

Tilo Linz ist Vorstand und Mitgründer der imbus AG, eines führenden Lösungsanbieters für Softwaretest, und seit mehr als 25 Jahren im Themengebiet Softwarequalitätssicherung und Softwaretest tätig. Als Gründungsmitglied und Vorsitzender des »German Testing Board« e.V. und Gründungsmitglied im »International Software Testing Qualifications Board« hat er die Aus- und Weiterbildung in diesem Fachbereich auf nationaler und internationaler Ebene maßgeblich mitgestaltet und vorangebracht. Tilo Linz ist auch Autor des Buches »Testen in Scrum-Projekten« (dpunkt.verlag), das aufbauend auf dem vorliegenden »Basiswissen Softwaretest« das Testen in agilen Projekten behandelt.

Andreas Spillner · Tilo Linz

Basiswissen Softwaretest

Aus- und Weiterbildung zum Certified Tester

Foundation Level nach ISTQB®-Standard

6., überarbeitete und aktualisierte Auflage

dpunkt.verlag

Andreas Spillner
andreas.spillner@hs-bremen.de

Tilo Linz
tilo.linz@imbus.de

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Birgit Bäuerlein
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

ISBN:
Print 978-3-86490-583-4
PDF 978-3-96088-501-6
ePub 978-3-96088-502-3
mobi 978-3-96088-503-0

6., überarbeitete und aktualisierte Auflage 2019

Wieblinger Weg 17
69123 Heidelberg

Hinweis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *hallo@dpunkt.de*.

5 4 3 2 1 0

Vorwort zur 6. Auflage

Bestseller

Bereits Ende 2002 erschien die erste Auflage dieses Buches. Seither entwickelte sich »Basiswissen Softwaretest« zum meistverkauften Buch zum Thema Softwaretest in deutscher Sprache. In der vorliegenden 6. Auflage haben wir den Inhalt weitreichend überarbeitet, aktualisiert und der aktuellen Version 2018 des Lehrplans zum »ISTQB® Certified Tester – Foundation Level« angepasst.

»Certified Tester«-Ausbildungsschema

Das anerkannte und sehr erfolgreiche »Certified Tester«-Ausbildungsschema gliedert sich in Säulen mit jeweils drei Ausbildungsstufen. Details dazu finden sich auf der Webseite des »International Software Testing Qualifications Board« [URL: ISTQB] und des »German Testing Board e.V.« [URL: GTB].

Der »Certified Tester« ist zu einer festen Marke in der IT-Industrie geworden und ist heute der De-facto-Standard für die Ausbildung im Bereich Softwarequalitätssicherung und Softwaretest sowohl in Deutschland als auch weltweit. Ende 2018 hat die Zahl der insgesamt ausgestellten Softwaretest-Zertifikate die 600.000 überschritten, davon in Deutschland knappe 67.000.

Wissen in der IT-Welt gefragt

In vielen Stellenanzeigen spiegelt sich dieser Umstand wider. Nicht nur bei Berufseinsteigern erwarten die Firmen Grundkenntnisse im Testbereich – am besten durch das Zertifikat nachgewiesen.

Certified Tester an Hochschulen

Der »Certified Tester« ist auch Teil der Informatikausbildung an vielen Hochschulen: Von A wie Aachen bis Z wie Zweibrücken wird der Lehrstoff im deutschsprachigen Bereich vermittelt. Welche Hochschulen aktuell entsprechende Lehrveranstaltungen anbieten bzw. planen, diese anzubieten, ist auf den Seiten des »German Testing Board« nachzulesen [URL: GTB Hochschulen]. 5 %–7 % aller Prüfungen zum »ISTQB® Certified Tester – Foundation Level« sind studentische Prüfungen.

Basiswissen

Trotz der stürmischen Entwicklung – auch in der Informatik gibt es Grundlagenwissen, das kaum Änderungen unterliegt. Von Anfang an haben wir den ersten Teil unseres Buchtitels »Basiswissen« ernst genommen und ganz bewusst keine Themen behandelt, die sich erst noch in der Praxis »beweisen müssen«. Auch »Spezialdisziplinen« im Testen, wie beispielsweise der Test von Webapplikationen oder der Test von eingebetteten Systemen, gehören für uns nicht zu den Grundlagen. Hier verweisen wir auf entsprechende aktuelle Literatur zu diesen Themen.

Ergänzende Literatur

Tilo und Andreas haben seit der letzten Ausgabe des vorliegenden Buches aktuelle Bücher veröffentlicht, auf die hier hingewiesen werden soll, da sie eine gute Ergänzung bzw. Vertiefung darstellen.

Buch: Testen in Scrum-Projekten – Leitfaden für Softwarequalität in der agilen Welt

Tilo hat mit seinem Buch »Testen in Scrum-Projekten – Leitfaden für Softwarequalität in der agilen Welt« (Erstauflage 2013) aufgezeigt, wie das Testen in agilen Projekten zu integrieren ist. Inzwischen gibt es einen Lehrplan (»ISTQB® Certified Agile Tester – Foundation Extension«) zu diesem Themenbereich und das Buch wurde entsprechend aktualisiert und liegt seit 2016 in der 2. Auflage vor.

Buch: Lean Testing für C++-Programmierer – Angemessen statt aufwendig testen

Andreas hat zusammen mit Ulrich Breymann[1] das Buch »Lean Testing für C++-Programmierer – Angemessen statt aufwendig testen« geschrieben. Im Buch sind alle Testverfahren des aktuellen ISO-Standards 29119 ausführlich beschrieben, die für den Komponententest relevant sind. Die Vorgehensweisen zum Testfallentwurf werden konkret mit den entsprechenden C++-Programmtexten und den jeweiligen Testfällen dargelegt. Dabei sind die Programmbeispiele so einfach gehalten, dass sie auch ohne C++-Kenntnisse verständlich sind.

Auf beide Bücher wird in diesem Buch des Öfteren verwiesen ([Linz 16], [Spillner 16]), um dem Leser weiterführende Informationen anzubieten. Vielleicht sind wir mit den vielen Hinweisen etwas über das Ziel hinausgeschossen, dafür bitten wir um Nachsicht – aber etwas Werbung für die eigene Arbeit wird hoffentlich noch erlaubt sein, oder?

Was hat sich geändert?

In der vorliegenden 6. Auflage von »Basiswissen Softwaretest« wurde eine umfassende Überarbeitung, Ergänzung und Aktualisierung des Inhalts vorgenommen.

Exkurs-Teile sind nicht Teil des Lehrplans.

Bei der Überarbeitung des ISTQB®-Lehrplans wurden einige Testverfahren auf höhere Ausbildungsstufen verschoben und sind somit nicht mehr Teil des »Foundation Level«-Lehrplans. Wir haben diesen Schritt nicht rigoros umgesetzt, sondern diese Verfahren weiterhin im Buch belassen, aber als *Exkurs* hervorgehoben. Wer das Buch nur zur Prüfungsvorbereitung nutzt, der überschlägt einfach die Exkurs-Teile.

Weitere Testverfahren aufgenommen

Aus vielen Gesprächen mit Lesern wissen wir, dass unser Buch als Nachschlagewerk bei der täglichen (Test-)Arbeit genutzt wird. Deshalb haben wir versucht, neben den Inhalten des Lehrplans weitere grundlegende Testverfahren aufzunehmen. Im Vergleich zu den vorherigen Ausgaben sind neue Verfahren hinzugekommen (z.B. Kombinatorisches Testen, »Pairwise Testing«).

---

1. Ulrich Breymann ist ehemaliger Professor an der Hochschule Bremen und Autor des C++-Standardwerks »Der C++-Programmierer«.

Das durchgehende Fallbeispiel und das Literaturverzeichnis wurden aktualisiert. Das Verzeichnis der Normen und Standards wurde ebenfalls überarbeitet und die Standards gestrichen, die durch den aktuellen Standard ISO 29119 abgelöst wurden. Die Angaben zu den Internetseiten (URLs) wurden kontrolliert und ggf. geändert bzw. ergänzt.

Webseite

Um die Leser über zukünftige Aktualisierungen am Lehrplan und am Glossar zu informieren, betreiben wir die Internetseite [URL: Softwaretest Knowledge]. Auf der Seite werden ggf. auch notwendige Korrekturen zum Buchtext aufgeführt. Dort finden sich ebenfalls Übungsaufgaben zu den einzelnen Buchkapiteln.

Danksagung

Erfolg hat meist viele Väter und Mütter – so auch hier. Wir möchten uns recht herzlich bei allen Kolleginnen und Kollegen des »German Testing Board« und des »International Software Testing Qualifications Board« bedanken. Ohne deren Engagement hätte das »Certified Tester«-Ausbildungsschema nicht den geschilderten Erfolg und weltweite Akzeptanz erhalten. Ebenso möchten wir uns für die vielen Anmerkungen und Rezensionen unserer Leser bedanken, die uns für unsere Arbeit am Buch sehr motiviert haben und zur Qualitätssteigerung beigetragen haben. Unserer Lektorin Frau Christa Preisendanz und dem gesamten dpunkt-Team möchten wir recht herzlich für die sehr gute langjährige Zusammenarbeit danken.

Wir wünschen allen Lesern gutes Gelingen bei der Umsetzung der im Buch beschriebenen Testansätze in der Praxis und – wenn das Buch die Grundlage für die Vorbereitung zur Prüfung zum »Certified Tester – Foundation Level« ist – viel Erfolg bei der Beantwortung der Prüfungsfragen.

Andreas Spillner und *Tilo Linz*
Bremen, Möhrendorf
Mai 2019

Inhaltsübersicht

Inhaltsverzeichnis

Vorwort zur 1. Auflage

Motivation für ein weiteres Testbuch

Warum haben wir ein Buch zum Thema Softwaretest geschrieben, wo es doch eine ganze Reihe von Büchern in diesem Bereich bereits gibt?

Wir beide beschäftigen uns seit vielen Jahren mit Themen und Fragestellungen im Bereich Softwaretest. Der eine mit einer eher wissenschaftlichen Ausrichtung, der andere mit der täglichen Anwendung der Prüfverfahren in der Praxis. Beide sind wir der Auffassung, dass der systematische Einsatz von Prüfverfahren in der Praxis in vielen Firmen verbesserungsfähig ist und es einen großen Mangel an qualifiziertem Personal gibt.

Die Initiative des ISEB (Information Systems Examination Board), ein dreistufiges Qualifizierungsprogramm im Bereich Softwaretest zu definieren, Kursanbieter in diesem Bereich zu akkreditieren und den Teilnehmern die Möglichkeit zu bieten, sich einer unabhängigen Prüfinstanz zu stellen und ein Zertifikat zu erhalten, ist von uns von Beginn an mit starkem Interesse verfolgt worden. Die Initiative ist vom ASQF (Arbeitskreis Software-Qualität in Franken e.V.) und der Fachgruppe TAV (Test, Analyse und Verifikation von Software) der Gesellschaft für Informatik aufgegriffen worden. Eine ausführliche Beschreibung der Zusammenhänge findet sich in der Einleitung des Buches.

Die Lehrpläne für das Qualifizierungsprogramm sind in Gremien von mehreren europäischen Fachexperten erarbeitet worden und bündeln das derzeitige Wissen im Bereich Softwaretest. Die Inhalte sind in vielen Büchern zu finden, aber eben nicht in einem einzigen Buch. Die Lehrpläne listen die einzelnen Punkte auf, die zu behandeln sind, enthalten aber weder detaillierte Beschreibungen der einzelnen Verfahren und Vorgehensweisen beim Testen von Software noch erläuternde Beispiele. Die Idee, ein »passendes« Buch zu schreiben, fanden wir sehr nahe liegend.

Grundlage für unser Buch ist der Lehrplan »Grundlagen des Softwaretestens« ASQF Certified Tester, Foundation Level [URL: GTB Lehrpläne]. Die ersten Erfahrungen mit dem Lehrplan zum Foundation Level führten zu dem Wunsch einer leichten Überarbeitung, die derzeit von einem Gremium des ISTQB (International Software Testing Qualifications Board) in Angriff genommen wird. Wir beide haben durch die Arbeit am Buch kleinere Schwachstellen im Lehrplan erkannt und beteiligen uns aktiv an dessen Überarbeitung. Die nach Beendigung der Arbeit beschlossenen Änderungen, die sich auf den Inhalt des Buches auswirken, werden wir auf der WWW-Seite zum Buch [URL: dpunkt] aktuell dokumentieren.

Der Titel unseres Buches »Basiswissen Softwaretest« sagt eigentlich schon alles. Wir sind davon überzeugt, dass jeder im Bereich der Softwareentwicklung Tätige – sei es beispielsweise in der Programmierung, in der Qualitätssicherung, in der Projektleitung oder im Management – ein Grundwissen im Softwaretest besitzen sollte, um seine tägliche Arbeit besser verrichten und den oft vernachlässigten Bereich Testen besser einschätzen oder überhaupt verstehen zu können. Die Aussage trifft selbstverständlich auch für die zukünftig Tätigen, die Studierenden und Auszubildenden im IT-Bereich, zu. Auch so mancher Lehrende wird sein Wissen mithilfe des Buches vervollständigen können.

Wir haben uns bemüht, ein leicht verständliches und kompaktes Buch zu schreiben, um dem großen Kreis an möglichen Lesern gerecht zu werden. Vorwissen im Bereich Testen oder Qualitätssicherung ist nicht erforderlich, Grundkenntnisse der Softwareentwicklung sind sicherlich hilfreich. Unser Buch hat zwar den Lehrplan zum »Certified Tester, Foundation Level« als Grundlage, es besteht aber keine zwanghafte Kopplung, auch das entsprechende Zertifikat zu erwerben.

Drei Geleitworte

Warum gibt es drei Geleitworte in Englisch und Deutsch in unserem Buch?

Durch unsere langjährigen Aktivitäten im Bereich Softwaretest haben wir zahlreiche internationale Kontakte aufbauen können. Als unser Buchprojekt konkrete Formen annahm, haben wir drei Personen um Geleitworte gebeten. Da alle drei viel gefragte und beschäftigte Persönlichkeiten sind, haben wir eher mit zurückhaltenden Reaktionen gerechnet. Um so erfreuter waren wir, dass wir tatsächlich drei Geleitworte innerhalb der Termine erhalten haben. Unsere »Risiko«-Planung war somit völlig unnötig. Da jedes Geleitwort einen anderen Aspekt vertieft, haben wir uns entschlossen, alle drei aufzunehmen.

Prof. David Parnas Ph.d, Dr. h.c., Dr. h.c., FRSC, P.Eng.

David Parnas, der Begründer der Modularisierung und des Geheimnisprinzips (*information hiding*), engagiert sich seit langem in Fragen der Ausbildung im Bereich Informatik. Für ihn muss die Informatik-Ausbildung eine Professionalität erreichen wie bei den Medizinern oder in den Rechts- und Ingenieurwissenschaften. Die allgemein anerkannte Festlegung von Lehrinhalten und der Nachweis des erworbenen Fachwissens, beispielsweise im Bereich Softwaretest, sieht er als einen Schritt in die richtige Richtung an. Teile des Buches sind während des Forschungsaufenthalts von Andreas Spillner auf Einladung von David Parnas an der McMaster University in Hamilton, Ontario in Kanada entstanden. Ausführliche Informationen zu David Parnas sind zu finden unter [URL: Parnas].

Martin Pol

Martin Pol gehört wohl zu den bekanntesten Persönlichkeiten im Testbereich in Europa, wenn nicht gar weltweit. Er ist Mitautor von T-Map (Test Management Approach) und TPI (Test Process Impro-

vement). 1998 erhielt er den »European Testing Excellence Award«. In seinem Geleitwort hebt er die Bedeutung des Testens im Softwareentwicklungsprozess hervor und wie wichtig Basiswissen im Softwaretest für alle im IT-Bereich arbeitenden Personen ist. Ausführliche Informationen zu Martin Pol sind abrufbar unter [URL: Pol].

Dorothy Graham A.B., M.Sc.

Dorothy Graham, seit vielen Jahren Beraterin und Trainerin im Softwarequalitäts- und Testbereich, hat das ISEB Software Testing Board mitgegründet und gehört ihm seitdem an. Ohne das große Engagement von Dorothy Graham wäre die ISEB-Initiative wohl nicht so weit vorangeschritten und die erfolgte Internationalisierung noch in weiter Ferne. Sie ist Mitautorin von Büchern zu den Themen »Software-Inspektionen« und »Softwaretest-Automatisierung«. Der »European Testing Excellence Award« wurde ihr 1999 zuerkannt. In ihrem Geleitwort macht Dorothy Graham die Vorteile einer Akkreditierung und Zertifizierung durch eine unabhängige Instanz deutlich. Ausführliche Informationen zu ihrer Person sind unter [URL: Graham] zu finden.

Wir möchten Dorothy Graham, Martin Pol und David Parnas für ihre Unterstützung und die aufgewendete Zeit recht herzlich danken.

Danksagung

Wie es sich für ein Projekt mit einem Qualitätsanspruch gehört, haben wir unsere Arbeit externen Reviewern vorgelegt. Wir haben viele Anregungen und Hinweise zur Verbesserung des Textes und der Beispiele erhalten. Wir möchten uns für die vielen geopferten Stunden Freizeit ganz herzlich bedanken bei Uwe Hehn, Ruth Keys, Karin Vosseberg und Mario Winter. Darüber hinaus gilt unser Dank den Kolleginnen und Kollegen der imbus AG, die ebenfalls durch zahlreiche Anmerkungen und konstruktive Diskussionen zum Gelingen des Buches beigetragen haben. Besonders erwähnen möchten wir Matthias Daigl und Thomas Roßner.

Bedanken möchten wir uns auch bei den Mitarbeiterinnen und Mitarbeitern des dpunkt.verlags. Zu Beginn unseres Projekts hatten wir uns eine Deadline gesetzt, was bei jedem Projekt erfolgen sollte. Auch Dank der großen Unterstützung des Verlags ist es gelungen, den von uns geplanten Fertigstellungstermin zu halten.

Wir wünschen allen Lesern des Buches erfolgreiche Stunden in dem Sinne, dass Testen nicht mehr als notwendiges Übel der Softwareentwicklung am Projektende betrachtet wird, sondern als herausfordernde, kreative Tätigkeit, die projektbegleitend erfolgt und neben der Aufdeckung von Fehlern auch zum Nachweis der Qualität der erstellten oder geänderten Software dient. Testen kann und soll auch Spaß machen.

Andreas Spillner und Tilo Linz
Bremen, Möhrendorf
September 2002

Geleitwort von David Parnas[1]

One of the biggest problems in the computer software field is the ability of inadequately qualified people to enter the profession and practice software development without limits. Software has become critical to our society; it is embedded in many devices that we count on, devices such as telephones and banking machines. Nonetheless, many of the people who write that software have not been educated for the job and have never demonstrated their qualifications to any objective professional body. There is no other field in the world where people without approved education can decide to identify themselves as »Engineers« and produce products that are essential to the safety and well-being of the public.

Software is well known for low reliability and lack of trustworthiness. In part this is attributable to the difficulty of dealing with the complexity of today's software systems, but the inadequate knowledge, skills, and professionalism of many of the practitioners also contributes to this problem. Moreover, we can thank the inadequately qualified people who produced today's software for the unreliability and complexity of the products that serve as support software for new products.

Our educational institutions have failed the public in this field. They have not recognized that those who study Computer Science require a professional education, one similar in style to the education provided to those who study medicine, law, or engineering. In those fields, the curriculum is designed around a set of professional requirements. Students are told what they must learn, rather than allowed to learn what they feel like learning. In the software field, universities have allowed the contents of courses to depend on the whim of the instructor, and the choice of courses to be largely up to the student. As a result, when an employer or client meets a graduate of a Computer Science programme, only experienced software developers are able to judge whether or not a graduate has the knowledge and skills needed for the job. Often, we cannot even find a graduate who has the appropriate body of knowledge and experience.

This problem is quite clear in the area of software testing. Many new employees are assigned testing duties without any knowledge how to design tests, how to evaluate test results, and how to draw valid conclusions can be drawn from the test results. Fortunately, there is now a useful international initiative to establish national Testing Boards,

1. Eine deutschsprachige Übersetzung bzw. die engl. Originale sind auf der Webseite des Buches [URL: dpunkt] zu finden.

which will approve courses of instruction and issue certificates to those who are able to demonstrate their understanding of basic terms and procedures in testing.

Unfortunately, this is still a shortage of appropriate study material for people who would like to become better software testers and pass the national tests. This book, »Basiswissen Softwaretest« by Spillner and Linz, fills this gap by providing a well organized and complete view of what is known about how to test software. It provides an essential component of the international effort to establish standards for software professionals and then help people to become fully qualified software developers.

David Lorge Parnas,
Ph.d, Dr.h.c., Dr.h.c., FRSC, P.Eng.
McMaster University, Hamilton,
Ontario, Canada
June 2002

Geleitwort von Martin Pol

Für viele Organisationen spielt die Qualität von Softwaresystemen eine immer größere Rolle. Es werden zunehmend Maßnahmen ergriffen, um eine höhere Qualität zu erreichen. Trotz ermutigender Resultate mit verschiedenen Ansätzen zur Qualitätssicherung ist die IT-Branche weit davon entfernt, fehlerfreie Software entwickeln zu können. Ein solches Ziel wird leider noch für geraume Zeit eine Utopie bleiben. Die Entwicklung von Softwaresystemen ist nach wie vor ein schwieriges »Handwerk« und auf absehbare Zeit kaum ohne Fehler zu meistern. Die Ursachen für Fehler sind vielfältig und schwer vorhersehbar. Man wird weiterhin nicht umhin kommen, viel Energie darauf zu verwenden, die Fehler ausfindig zu machen. Das Testen wird ein wichtiger Bestandteil der Entwicklung und Wartung bleiben und oft mehr als 30 – 40 % des Gesamtbudgets aufzehren.

Das Testen ist nicht mehr nur eine Phase, die nach der Implementierung kommt, sondern ist ebenso ernst zu nehmen wie andere Aktivitäten der Entwicklung von Systemen. Testen ist zu einer eigenständigen Aufgabe und Tätigkeit geworden. Die Informatik hat sukzessive die Grundideen des Testens akzeptiert: Testen von Software ist ein Prozess, bestehend aus Planung und Vorbereitung einerseits und Messen und Prüfen andererseits. Testen dient dazu, die Charakteristika eines Systems festzustellen und die Unterschiede zwischen dem Ist- und dem Sollverhalten aufzuzeigen. Gute Qualität kann als Erfüllung der Anforderungen gesehen werden. Somit ist das Ergebnis des Testens, die vorhandene Qualität aufzuzeigen und Verbesserungen zu ermöglichen. Es gibt einen Einblick darin, welche Risiken bestehen, wenn eine geringere Qualität akzeptiert wird. Dies ist ebenfalls ein wesentliches Ziel des Testens.

Da Time-to-Market, Wettbewerb, Globalisierung und Quality of Service, einschließlich der Qualität von Softwaresystemen, für viele Firmen zu einem wichtigen Überlebensfaktor geworden sind, steigt der Bedarf für einen angemessenen Testprozess immer weiter an. Sowohl die immer größer werdende Bedeutung von Software in unserer Gesellschaft als auch das Budget, das für das Testen ausgegeben wird, bestätigen den Bedarf an einem gut strukturierten und verlässlichen Testprozess innerhalb der Softwareentwicklung. Hierbei sind ein strukturiertes Vorgehen, eine angemessene Organisationsstruktur und die entsprechende Infrastruktur notwendig.

Dieses Buch schafft die Grundlage für einen gut strukturierten Testprozess. Es beschreibt die Grundsätze des Testens, alle Aktivitäten zum Testen innerhalb des Entwicklungsprozesses, Techniken des Testens, Testmanagement und Testwerkzeuge. Daher ist das Buch für jeden von

Interesse, der im IT-Bereich arbeitet: Entwickler, Tester, Anwender, Projekt- und Abteilungsleiter etc. Darüber hinaus ist es ein idealer Begleiter, um sich auf das ISEB-Zertifizierungsprogramm vorzubereiten. Egal für welchen Einsatz – ich bin mir sicher, dass dieses umfassende Buch von sehr großem Nutzen sein wird.

Martin Pol
Polteq IT Services B.V.
Amersfoort, The Netherlands
June 2002

Geleitwort von Dorothy Graham

Was macht ein gutes Zertifizierungsprogramm für Fachleute aus? Ein erfolgreiches Programm muss vor allem drei Hauptkriterien erfüllen:

- Es muss ein Grundkonsens über das Thema vorhanden sein
- Es muss denen, die das Zertifikat erhalten (und ihren Arbeitgebern), einen Mehrwert bieten
- Es muss allen Beteiligten gegenüber fair sein und darf niemanden bevorzugen

Im vorliegenden Buch geht es um ein Programm, das diese Kriterien erfüllt. Beim Thema Softwaretesten ist ein ausreichender Konsens vorhanden, der als Basis für ein Zertifikat dienen kann. Das neue deutsche ASQF-Programm und das erfolgreiche ISEB-Programm aus Großbritannien werden sich bald international durchsetzen. Der Hauptgrund für den Erfolg des britischen Programms ist meiner Meinung nach die Unabhängigkeit des akkreditierenden und prüfenden Gremiums. Ich glaube auch, dass diese Philosophie den Erfolg des internationalen Programms für zertifizierte Tester sicherstellen wird, bei dem Deutschland momentan eine Vorreiterrolle spielt.

Bei dieser Unabhängigkeit spielen zwei Aspekte eine Rolle: zum einen die Akkreditierung von Ausbildungseinrichtungen und zum anderen die Prüfung der einzelnen Kandidaten.

Worin liegt der Unterschied zwischen Zertifizierung und Akkreditierung? Bei der Zertifizierung geht es um den einzelnen Teilnehmer, während die Akkreditierung Ausbildungseinrichtungen betrifft. Der Teilnehmer erhält ein Zertifikat, wenn er sein Wissen unter Beweis stellt. Das Wissen wird durch gewonnene Arbeitserfahrung und durch Fortbildungsveranstaltungen erworben. Wenn ein Grundkonsens über das Thema vorhanden ist (und damit eine Art »Lehrplan«) kann ein unabhängiges Gremium Kurse von Ausbildungseinrichtungen prüfen – dabei handelt es sich um die Akkreditierung. Die Akkreditierung bescheinigt, dass die Ausbildung einen bestimmten Standard im Hinblick auf Inhalt und Organisation erfüllt. Wenn sich Ausbildungseinrichtungen selbst akkreditieren, ist dies nur wenig aussagekräftig; die Akkreditierung durch ein unabhängiges Gremium ist sehr viel glaubwürdiger.

Um ein Zertifikat zu erhalten, muss der Kandidat eine Prüfung bestehen, die auf einem anerkannten Lehrplan beruht. Wenn die Prüfung durch ein unabhängiges Gremium gestellt, überwacht und benotet wird, bestehen gleiche Chancen für alle Kandidaten (und Ausbildungseinrichtungen). Durch die Unabhängigkeit des überwachenden Gremiums steigt der Wert des vergebenen Zertifikats.

Es gibt Programme, bei denen sich die Ausbildungseinrichtungen selbst akkreditieren und ihre eigenen Prüfungen stellen. Obwohl diese Programme durchaus einen gewissen Wert haben, bieten unabhängige Programme mehr. Diejenigen, die Zertifizierungsprogramme generell ablehnen, betonen, dass es bei allen Programmen Probleme geben kann. Dies ist durchaus richtig – das perfekte Programm gibt es nicht. Dennoch bin ich der Überzeugung, dass unabhängige Programme wie ISEB, ASQF oder der geplante internationale Standard für die Teilnehmer und ihre Arbeitgeber von erheblichem Wert sein werden.

Woher wissen wir, dass ein unabhängiges Programm funktionieren wird? Weil ISEB bereits in vielen Ländern funktioniert. ISEB steht für Information Systems Examination Board. Das ISEB wurde ursprünglich als unabhängiges Gremium für Qualifizierungen im Bereich Systemanalyse und -design gegründet. Es gehört nun zu der British Computer Society und bietet Qualifikationen in zehn Fachgebieten an, darunter Projektmanagement, IT-Service-Management, Geschäftssystementwicklung, Datenschutz und -sicherheit – und seit 1998 auch Softwaretesten. An der Spitze jedes Fachgebiets bilden Vertreter aus Industrie und Hochschule einen Ausschuss. Arbeitsgruppen kümmern sich um Einzelfragen. So gibt es im Bereich Softwaretesten eine Arbeitsgruppe für die Akkreditierung und eine für die Prüfung. ISEB-Mitarbeiter sorgen für die Verwaltung der Ausschüsse und der Arbeitsgruppen, bei den Prüfungen stellen sie das Aufsichtspersonal, verteilen die von der Prüfungs-AG erarbeiteten Prüfungsfragen und werten die Antworten der Kandidaten aus.

Der Ausschuss Softwaretesten des ISEB wurde 1997 gegründet. Der Lehrplan für den Foundation Level wurde von Ausschussmitgliedern aufgestellt und der erste Kurs im Oktober 1998 abgehalten. Seither läuft das Programm mit großem Erfolg und übertrifft alle Erwartungen. Es ist inzwischen – nach IT-Service-Management – das erfolgreichste Programm des ISEB. Bis Mitte 2002 waren bereits 23 Ausbildungseinrichtungen für den Kurs Softwaretesten (Foundation Level) akkreditiert, darunter Organisationen aus Deutschland, den Niederlanden, Schweden, Irland und Australien sowie Großbritannien. Über 8000 Kandidaten haben in weniger als vier Jahren an der Prüfung zum Softwaretester teilgenommen. Das Programm hat den Status des Softwaretestens und die Anerkennung, die man Softwaretestern entgegenbringt, deutlich gehoben. Auf dem britischen Arbeitsmarkt wird man inzwischen ohne das Zertifikat in der Regel gar nicht erst zu einem Vorstellungsgespräch als Softwaretester eingeladen.

Wie sieht die zukünftige Entwicklung aus? Das International Software Testing Qualification Board (ISTQB) wurde 2002 gegründet, so dass Akkreditierung und Prüfung auf Foundation Level bald in allen angeschlossenen Ländern durchweg von unabhängigen Gremien vorgenommen werden. Softwaretester auf der ganzen Welt werden sich besser verständigen können und anerkannter sein. Dieses Buch wird für deutschsprachige Länder einen wichtigen Beitrag dazu leisten. Das internationale Zertifikat wird weltweit Grundkenntnisse im Bereich Softwaretesten sichern. Dies ist eine spannende Zeit für das Softwaretesten!

Dorothy Graham
Macclesfield, UK
June 2002

1 Einleitung

Software ist allgegenwärtig! Es gibt kaum noch Geräte, Maschinen oder Anlagen, in denen die Steuerung nicht über Software bzw. Softwareanteile realisiert wird. So sind im Automobil wesentliche Funktionen wie die Motor- oder Getriebesteuerung seit Langem durch Software realisiert. Hinzu kommen immer intelligentere softwarebasierte Fahrerassistenzsysteme, vom Bremsassistenten über die automatische Einparkhilfe oder Spurassistenten bis zum vollständig autonom fahrenden Fahrzeug. Die Software – und besonders deren Qualität – trägt somit ganz entscheidend nicht nur zum Funktionieren unserer Welt bei, sondern definiert zunehmend auch unsere Sicherheit.

Ebenso ist der reibungslose Geschäftsablauf in Firmen und Organisationen heute weitgehend von der Zuverlässigkeit der Softwaresysteme abhängig, die zur Abwicklung der Geschäftsprozesse oder einzelner Aufgaben eingesetzt werden. Software entscheidet damit auch über die künftige Wettbewerbsfähigkeit der Unternehmen. Wie schnell beispielsweise ein Versicherungskonzern ein neues Produkt oder auch nur einen neuen Tarif am Markt einführen kann, ist heutzutage davon abhängig, wie schnell die konzerneigenen IT-Systeme entsprechend angepasst oder ausgebaut werden können.

Hohe Abhängigkeit vom reibungslosen Funktionieren der Software

In beiden Bereichen (technische und kommerzielle Softwaresysteme) ist die Qualität der Software damit zum entscheidenden Faktor für den Erfolg von Produkten und Unternehmen geworden.

Die meisten Unternehmen haben diese hohe Abhängigkeit von Software – sowohl vom Funktionieren der vorhandenen als auch von der schnellen Verfügbarkeit neuer oder besserer Software – erkannt. Sie investieren daher in ihre Softwareentwicklungskompetenz und in eine verbesserte Qualität ihrer Softwaresysteme. Ein wichtiges Mittel, dies zu erreichen, ist das systematische Prüfen und Testen der entwickelten Software. Teilweise haben sehr umfassende und rigide Verfahren Einzug in die Praxis der Softwareentwicklung gefunden. In vielen Projekten ist aber weiterhin ein erheblicher Bedarf an Wissensvermittlung zu Prüf- und Testverfahren und deren Leistungsfähigkeit und Nutzen erforderlich.

Grundlagenwissen zum strukturierten Prüfen und Testen

Mit diesem Buch stellen wir Grundlagenwissen bereit, das bei entsprechender Umsetzung zu einem strukturierten, systematischen Vorgehen beim Prüfen und Testen führt und somit zur Qualitätsverbesse-

rung der Software beiträgt. Der Inhalt des Buches ist so abgefasst, dass kein Vorwissen im Bereich der Softwarequalitätssicherung vorausgesetzt wird. Das Buch ist als Lehrbuch konzipiert und auch zum Selbststudium geeignet. Ein durchgängiges Fallbeispiel hilft, jedes dargestellte Thema und seine praktische Umsetzung schnell zu verstehen.

Ansprechen möchten wir Softwaretester[1] in allen Unternehmen und Organisationen, die ihre Softwaretestkenntnisse auf eine fundierte Grundlage stellen wollen, Programmierer und Entwickler, die Testaufgaben übernommen haben bzw. übernehmen werden, aber auch Softwaremanager, die in den Projekten über Verbesserungsmaßnahmen und Budgets entscheiden. Auch Quereinsteiger in entwicklungsnahen IT-Berufen und Mitarbeiter in Fachabteilungen, die an der Abnahme, Einführung oder Weiterentwicklung von IT-Anwendungen beteiligt sind, werden Hilfestellung für ihre tägliche Arbeit finden.

Das lebenslange Lernen ist besonders im IT-Bereich unverzichtbar. Auch zum Thema Softwaretest werden Weiterbildungsmaßnahmen von vielen Firmen und Trainern angeboten. Ebenso werden an immer mehr Hochschulen Lehrveranstaltungen zu diesem Thema durchgeführt. Das Buch soll Lernende und Lehrende im gleichen Maße ansprechen.

Zertifizierungsprogramm für Softwaretester

Der weltweite Standard für die Aus- und Weiterbildung im Bereich Software-Qualitätssicherung und Softwaretest ist heute das »ISTQB® Certified Tester«-Schema des International Software Testing Qualifications Board (ISTQB®). Das ISTQB® [URL: ISTQB] koordiniert die nationalen Initiativen und sorgt für die Einheitlichkeit und Vergleichbarkeit der Lehr- und Prüfungsinhalte unter den beteiligten Ländern. Die nationalen Testing Boards sind zuständig für die Herausgabe und Pflege landessprachlicher Lehrpläne und für die Definition und Durchführung von Prüfungen in ihren jeweiligen Ländern. Sie überprüfen die im jeweiligen Land angebotenen Kurse nach definierten Qualitätskriterien und sprechen Akkreditierungen der Trainingsanbieter aus. Die Testing Boards gewährleisten damit einen hohen Qualitätsstandard der Kurse, und die Kursteilnehmer erhalten mit bestandener Prüfung einen international anerkannten Qualifikationsnachweis. Die entsprechenden Gremien im deutschsprachigen Raum sind das Austrian Testing Board [URL: ATB], das German Testing Board [URL: GTB] und das Swiss Testing Board [URL: CHTB]. In diesen Gremien sind Trainingsanbieter, Testexperten aus Industrie- und Beratungsunternehmen sowie Hochschullehrende organisiert. Wichtige Kompetenz brin-

1. Wir verwenden im Buch überwiegend die männliche Form und wollen damit Frauen sowie alle anderen Geschlechter selbstverständlich nicht ausschließen bzw. ausgrenzen.

gen weiterhin Vertreter verschiedener Fachverbände ein. So arbeiten im GTB u.a. Mitglieder der Fachgruppe TAV (Test, Analyse und Verifikation von Software) [URL: GI TAV] der Gesellschaft für Informatik e.V. (GI e.V.) mit.

Dreistufiges Qualifizierungsschema

Das »Certified Tester«-Ausbildungsschema gliedert sich in Säulen mit jeweils drei Ausbildungsstufen. Details dazu finden sich auf der Webseite des ISTQB® [URL: ISTQB] und des GTB [URL: GTB]. Die Grundlagen zum Softwaretest sind im Lehrplan zum »Foundation Level« beschrieben. Darauf aufbauend kann das Zertifikat zum »Advanced Level« [URL: GTB Lehrpläne] erworben werden, um vertiefte Kenntnisse im Prüfen und Testen nachzuweisen. Die dritte Stufe, das »Expert Level«-Zertifikat, richtet sich an erfahrene, professionelle Softwaretester und besteht aus einer Reihe von Modulen zu unterschiedlichen Spezialthemen (s.a. Abschnitt 6.1.2).

Der Inhalt des Buches deckt den Stoff des Zertifikats »Foundation Level« ab. Das prüfungsrelevante Fachwissen kann im Selbststudium erworben oder nach bzw. parallel zu einer Teilnahme an einem Kurs vertieft werden.

Kapitelübersicht

Die Themen des Buches und somit auch die grobe Struktur der Inhalte der Kurse zum Erwerb des »Foundation Certificate« sind im Folgenden beschrieben.

Grundlagen des Softwaretestens

In Kapitel 2 werden die Grundlagen des Softwaretestens erörtert. Neben der Motivation, wann, mit welchen Zielen und wie intensiv getestet werden soll, wird das Konzept eines grundlegenden Testprozesses beschrieben. Es wird auf die psychologischen Schwierigkeiten eingegangen, die entstehen, wenn beim Test der eigenen Software die selbst verursachten Fehler nachgewiesen werden sollen.

Testen im Softwarelebenszyklus

Kapitel 3 stellt in der Softwareentwicklung gebräuchliche Lebenszyklusmodelle (sequenziell, iterativ, inkrementell, agil) knapp vor und erläutert, welche Rolle das Testen im jeweiligen Modell spielt. Die verschiedenen Teststufen und Testarten werden erklärt und auf die Unterschiede beim funktionalen und nicht funktionalen Test eingegangen. Das Thema Regressionstest wird ebenfalls angesprochen.

Statischer Test

Statische Verfahren, d.h. Verfahren, bei denen das Testobjekt nicht zur Ausführung kommt, werden in Kapitel 4 vorgestellt. Reviews und statische Tests werden bereits in vielen Unternehmen mit gutem Erfolg angewendet. Die unterschiedlichen Vorgehensweisen werden ausführlich beschrieben.

Dynamischer Test

Das Kapitel 5 behandelt den Test im engeren Sinne. Die Klassifizierung der dynamischen Testverfahren in »Blackbox«- und »Whitebox«-Verfahren wird erörtert. Zu jeder Klasse werden unterschiedliche Testverfahren bzw. -methoden an Beispielen ausführlich erklärt.

Auf die sinnvolle Verwendung des erfahrungsbasierten bzw. intuitiven Tests, nämlich in Ergänzung zu den anderen Verfahren, wird am Ende des Kapitels eingegangen.

Testmanagement

Welche Organisationsformen und Aufgaben beim Testmanagement zu berücksichtigen sind, wird in Kapitel 6 diskutiert. Behandelt werden auch Anforderungen an Fehlerverfolgung und Konfigurationsmanagement sowie das Thema Wirtschaftlichkeit des Testens.

Testwerkzeuge

Testen von Software ist ohne Werkzeugunterstützung sehr arbeits- und zeitintensiv. Im letzten Kapitel des Buches (Kap. 7) werden unterschiedliche Klassen von Werkzeugen zur Testunterstützung vorgestellt und Hinweise zur Werkzeugauswahl und Werkzeugeinführung gegeben.

Fallbeispiel »VirtualShowRoom – VSR-II«

Die in diesem Buch vorgestellten Vorgehensweisen beim Testen von Software werden größtenteils anhand eines durchgängigen Fallbeispiels veranschaulicht. Das folgende Szenario liegt diesem Beispiel zugrunde:

Ein Automobilkonzern betreibt seit mehr als zehn Jahren ein elektronisches Verkaufssystem, genannt VirtualShowRoom (VSR). Dieses Softwaresystem ist weltweit bei allen Autohändlern der Marke installiert und in Betrieb:

- Ein Kunde, der ein Fahrzeug erwerben möchte, kann unterstützt durch einen Verkäufer oder selbstständig sein Wunschfahrzeug am Bildschirm konfigurieren (Modellauswahl, Farbe, Ausstattung usw.). Das System zeigt mögliche Modelle und Ausstattungsvarianten an und ermittelt zu jeder Auswahl sofort den jeweiligen Listenpreis. Diese Funktionalität wird vom Teilsystem *DreamCar* realisiert.
- Hat sich der Kunde für ein Fahrzeug entschieden, kann er am Bildschirm mit *EasyFinance* die für ihn optimale Finanzierung kalkulieren, mit *JustInTime* das Fahrzeug online bestellen und mittels *NoRisk* auch die passende Versicherung abschließen. Das Teilsystem *FactBook* schließlich verwaltet sämtliche Kundeninformationen und Vertragsdaten.

Der Konzernbereich Marketing und Vertrieb hat entschieden, dass das System modernisiert werden soll und die folgenden Projektziele definiert:

- VSR ist ein klassisches Client-Server-System. Das neue System VSR-II soll ein webbasiertes System sein, das auf beliebigen Geräten (Desktop, Tablet, Smartphone) über Browser genutzt werden kann.
- Jedes der bisherigen Teilsysteme *DreamCar*, *EasyFinance*, *FactBook*, *JustInTime, NoRisk* wird auf die neue Technologie portiert und in diesem Zuge auch (in unterschiedlichem Umfang) funktional erweitert.
- Als neues System ist das Teilsystem *ConnectedCar* anzubinden. Dieses System ermittelt und verwaltet Statusinformationen aller verkauften Fahrzeuge und gibt dem Fahrer aber auch dem Händler oder Servicepartner Informationen über anstehende Wartungs- und Reparaturarbeiten. Außerdem bietet es dem Fahrer verschiedene buchbare Services (wie Helpdesk, Notruf etc.). Auch Softwareupdates für das Fahrzeug können »Over the air« eingespielt und freigeschaltet werden.
- Jedes der fünf alten Teilsysteme wird von einem eigenen Entwicklungsteam separat portiert und weiterentwickelt. Ein weiteres Team kümmert sich um die Neuentwicklung von *ConnectedCar*. Insgesamt sind rund 60 Entwickler und weitere Mitarbeiter aus den jeweils betroffenen konzerninternen Fachabteilungen an dem Projekt beteiligt sowie externe Softwarefirmen.
- Die Teams arbeiten agil nach Scrum. Im Rahmen der agilen Entwicklung soll jedes Teilsystem während der Iterationen getestet werden. Die Auslieferung des Systems erfolgt in Inkrementen.
- Um einen komplizierten mehrfachen Abgleich von Daten zwischen Altsystem und Neusystem zu vermeiden, ist vorgesehen, dass VSR-II erst dann erstmalig produktiv in Betrieb geht, wenn die Funktionalität des alten VSR erreicht ist.

Im Rahmen des Projekts und des agilen Vorgehens werden die meisten Projektmitarbeiter in unterschiedlichem Umfang mit Testarbeiten konfrontiert oder betraut werden. Das Grundlagenwissen über Testtechniken und Vorgehensweisen, das sie dazu benötigen, wird in diesem Buch vermittelt. Abbildung 1–1 zeigt das neue System VSR-II in der Übersicht.

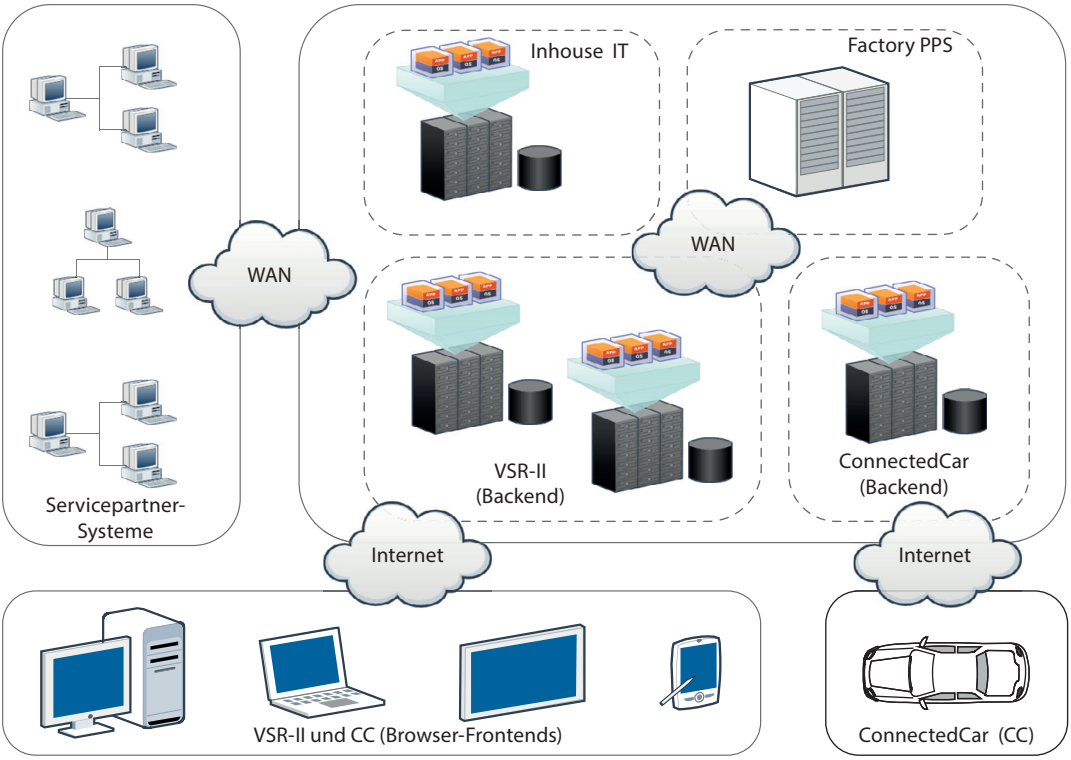

Abb. 1–1 *VSR-II in der Übersicht*

Hinweise zu Lehrstoff und Prüfung

Im Anhang werden wichtige Hinweise zum Lehrstoff und zur Prüfung zum Certified Tester gegeben. Weitere Anhänge des Buches beinhalten ein Glossar und das Quellenverzeichnis. Textpassagen, die über den Stoff des Lehrplans hinausgehen, sind als »**Exkurs**« gekennzeichnet.

WWW-Seite zum Buch

Unter [URL: Softwaretest Knowledge] finden sich neben Übungsaufgaben zu den einzelnen Buchkapiteln weitere und aktuelle Informationen zum Buch wie auch zu anderen Büchern der Autoren, die das Certified-Tester-Ausbildungsschema ergänzen.

2 Grundlagen des Softwaretestens

Dieses einleitende Kapitel erklärt die Grundbegriffe des Softwaretestens, die in den weiteren Kapiteln vorausgesetzt werden. Wichtige Begriffe werden zusätzlich an dem praxisnahen Fallbeispiel VSR-II-System veranschaulicht, das im gesamten Buch immer wieder zur Illustration und Motivation des Lehrstoffs verwendet wird. Die sieben Grundsätze des Testens werden vorgestellt. Hauptteil des Kapitels ist der Testprozess, der mit seinen einzelnen Aktivitäten detailliert erläutert wird. Am Ende des Kapitels wird auf psychologische Probleme beim Testen eingegangen und wie diese umgangen bzw. verringert werden können.

2.1 Begriffe und Motivation

Anforderungen an die Qualität

Bei der Herstellung eines Industrieprodukts werden die entstehenden Produkte üblicherweise daraufhin kontrolliert, ob sie den gestellten Anforderungen entsprechen. Es wird meist durch Stichproben geprüft, ob das Produkt die geforderte Aufgabe löst. Je nach Produkt gibt es auch unterschiedliche Anforderungen an die Qualität der Lösung. Erweist sich ein Produkt als fehlerhaft, so müssen ggf. Korrekturen im Produktionsprozess oder in der Konstruktion erfolgen.

Software ist immateriell.

Was allgemein für die Herstellung eines Industrieprodukts gilt, trifft entsprechend für die Produktion bzw. Entwicklung von Software zu. Die Prüfung der Teilprodukte bzw. des Endprodukts gestaltet sich allerdings schwieriger, da die erstellte Software immateriell ist und daher nicht »greifbar« und eine Prüfung deshalb nicht »handfest« durchgeführt werden kann. Eine optische Prüfung ist nur sehr eingeschränkt durch intensives Lesen der Entwicklungsdokumente möglich.

Fehlerhafte Software führt zu Problemen.

Software, die unzuverlässig oder inkorrekt arbeitet, kann zu erheblichen Problemen führen. Hierzu gehören der Verlust von Geld und Zeit, die Schädigung des Geschäftsrufs bis hin zu Verletzungen von Personen oder sogar deren Tod. Beispiele für gravierende Softwarefehler finden sich oft in der aktuellen Tagespresse, wenn etwa die »Autopilot«-Software eines teilautonom fahrenden Autos fehlerhaft ist und zu spät oder falsch reagiert.

Testen liefert eine Einschätzung der Qualität.

Es ist daher wichtig, die Qualität der Software zu prüfen, um das Risiko eines Softwareausfalls oder eines Softwarefehlers zu minimieren. Das Testen von Software liefert eine Einschätzung der Qualität und verringert die Risiken beim Einsatz der Software, da mögliche Fehler während des Testens aufgedeckt werden können. Dem Testen von Software kommt somit eine sehr wichtige, aber auch sehr schwierige Aufgabe zu.

Beispiel: Risiko durch Softwarefehler

Jedes Release des VSR-II-Systems unseres Fallbeispiels muss vor Auslieferung und Einsatz angemessen geprüft werden, um mögliche Fehler vorab zu erkennen und beheben zu können. Führt das System beispielsweise Bestellvorgänge falsch aus, könnte dies für Kunden und Händler, aber auch für den Autokonzern unter Umständen einen großen finanziellen Schaden und/oder Imageverlust zur Folge haben. Jedenfalls birgt die Nichterkennung eines solchen Fehlers ein hohes Risiko beim Einsatz der Software.

Testen ist eine stichprobenhafte Prüfung.

Oft wird unter Testen die (im Allgemeinen stichprobenartige) Ausführung[1] der zu prüfenden Software (Testobjekt) auf einem Rechner verstanden. Dazu werden einzelne Testfälle ausgeführt, d.h., das Testobjekt wird mit Testdaten versehen und ausgeführt. Die anschließende Bewertung prüft, ob das Testobjekt die geforderten Eigenschaften erfüllt und sich konform zu den Anforderungen verhält.[2]

Testen ist mehr als die Ausführung von Testfällen auf dem Rechner.

Zum Testen gehört aber weit mehr als die Ausführung von Testfällen. Software zu testen ist ein Prozess – der Testprozess, der unterschiedliche Aktivitäten umfasst. Die Testdurchführung, inklusive der Prüfung der Ergebnisse, ist nur eine der Aktivitäten. Weitere Aktivitäten im Testprozess sind die Testplanung, die Analyse, das Design und die Realisierung von Tests. Die Anfertigung von Berichten über den Testfortschritt und über die Testergebnisse sowie die Beurteilung der Qualität eines Testobjekts und die Risikobewertung sind zusätzliche Aufgaben. Die Testaktivitäten werden je nach Lebenszyklus der Software unterschiedlich organisiert und durchgeführt. Testaktivitäten und Testdokumentation werden häufig vertraglich zwischen Auftraggeber und Auftragnehmer oder durch gesetzliche oder Firmenstandards festgelegt. Eine detaillierte Beschreibung der einzelnen Aktivitäten im Testprozess folgt in Abschnitt 2.3 und in Abschnitt 6.3.

1. Hier ist das dynamische Testen (s. Kap. 5) gemeint. Beim statischen Test (s. Kap. 4) wird das Testobjekt nicht ausgeführt.
2. Es ist nicht möglich, die korrekte Umsetzung aller Anforderungen durch Testen nachzuweisen (s. u.).

Statisches und dynamisches Testen

Neben den Tests, die auf dem Rechner ausgeführt werden (dynamische Test, s. Kap. 5), können und sollen auch Dokumente wie Anforderungsspezifikation, User Stories und Quellcode so früh wie möglich einer Prüfung unterzogen werden. Derartige Tests werden statische Tests (s. Kap. 4) genannt. Je früher Fehler in den Dokumenten gefunden und behoben werden, je besser ist es für die weitere Entwicklung der Software, da nicht mit fehlerbehafteten Dokumenten weitergearbeitet wird.

Verifizierung und Validierung

Testen umfasst aber nicht nur die Prüfung, ob sich das System entsprechend den Anforderungen, User Stories oder anderen Spezifikationen verhält, sondern auch die Prüfung, ob sich das System entsprechend den Vorstellungen und Wünschen der Nutzer bzw. Anwender in der Betriebsumgebung verhält, ob die beabsichtigte Nutzung möglich ist bzw. das System seinen Zweck erfüllt. Testen beinhaltet somit auch die Validierung (s.a. 7. Grundsatz: Trugschluss: Keine Fehler bedeutet ein brauchbares System in Abschnitt 2.1.6).

Kein umfangreiches System ist fehlerfrei.

Ein fehlerfreies Softwaresystem gibt es derzeit nicht und wird es in naher Zukunft wahrscheinlich auch nicht geben, sobald das System einen gewissen Grad an Komplexität und Umfang an Programmzeilen umfasst. Häufig liegt ein Fehler darin begründet, dass sowohl während der Entwicklung als auch beim Testen der Software gewisse Ausnahmesituationen nicht bedacht bzw. nicht überprüft wurden. Sei es das Schaltjahr, das nicht richtig berechnet wird, oder die nicht berücksichtigten Randbedingungen, wenn es um Zeitverhalten oder Ressourcenbedarf geht. Es ist daher durchaus üblich – oder oft auch unumgänglich – dass Software und Systeme in Betrieb genommen werden, obwohl Fehler bei bestimmten Eingabekonstellationen auftreten. Auf der anderen Seite arbeiten aber sehr viele Softwaresysteme in ganz unterschiedlichen Bereichen zuverlässig tagein, tagaus.

Fehlerfreiheit nicht durch Testen erreichbar

Selbst wenn alle ausgeführten Tests keinen einzigen Fehler mehr aufdecken, kann (außer bei sehr sehr kleinen Programmen) nicht ausgeschlossen werden, dass es zusätzliche Tests gibt, die weitere Fehler aufzeigen würden. Fehlerfreiheit kann mit Testen nicht nachgewiesen werden.

2.1.1 Fehlerbegriff

Testbasis als Grundlage

Eine Situation kann nur dann als fehlerhaft eingestuft werden, wenn vorab festgelegt wurde, wie die erwartete bzw. als korrekt spezifizierte Situation aussehen soll. Zur Bestimmung der korrekten Situation werden die Anforderungen an das zu testende System(teil), aber auch weitere Informationen herangezogen. In diesem Zusammenhang wird von der Testbasis gesprochen, gegen die getestet wird und die als Grundlage der Entscheidung dient, ob ein korrektes oder fehlerhaftes Verhalten vorliegt.

Was gilt als Fehler?

Ein Fehler ist somit die Nichterfüllung einer festgelegten Anforderung, eine Abweichung zwischen dem Istverhalten (während der Ausführung der Tests oder des Betriebs festgestellt) und dem Sollverhalten (in der Spezifikation, den Anforderungen oder den User Stories festgelegt). Wann liegt aber ein nicht anforderungskonformes Verhalten des Systems vor?

Im Gegensatz zu physischen Systemen entstehen Fehler in einem Softwaresystem nicht durch Alterung oder Verschleiß. Jeder Fehler ist seit dem Zeitpunkt der Entwicklung in der Software vorhanden. Er kommt jedoch erst bei der Ausführung der Software zum Tragen.

Fehlerwirkung

Für diesen Sachverhalt wird der Begriff Fehlerwirkung verwendet. Der englische Fachbegriff hierfür lautet »Failure«. Weitere Bezeichnungen sind Fehlfunktion, äußerer Fehler oder Ausfall. Beim Test der Software oder auch erst bei deren Betrieb wird eine Fehlerwirkung für den Tester oder Anwender nach außen sichtbar. Zum Beispiel ist ein Ausgabewert falsch oder das Programm stürzt ab.

Fehlerzustand

Zwischen dem Auftreten einer Fehlerwirkung und deren Ursache muss unterschieden werden. Eine Fehlerwirkung hat ihren Ursprung in einem Fehlerzustand (»Fault«) der Software. Dieser Fehlerzustand wird auch als Defekt oder innerer Fehler bezeichnet. Auch das englische Wort »Bug« ist gebräuchlich. Dies ist beispielsweise eine falsch programmierte oder vergessene Anweisung im Programm.

Fehlermaskierung

Es ist durchaus möglich, dass ein Fehlerzustand durch einen oder mehrere andere Fehlerzustände in anderen Teilen des Programms kompensiert wird (Fehlermaskierung). Eine Fehlerwirkung tritt in diesem Fall erst dann zutage, nachdem der oder die maskierenden Fehlerzustände korrigiert worden sind. Korrekturen können somit zu Seiteneffekten führen.

Ein Problem ist, dass ein Fehlerzustand nicht zu einer Fehlerwirkung führen muss. Eine Fehlerwirkung kann gar nicht, einmal oder immer und somit für alle Benutzer des Systems auftreten. Eine Fehlerwirkung kann weit entfernt vom Fehlerzustand zum Tragen kommen. Ein Beispiel ist eine kleine Verfälschung von gespeicherten Daten, die bei der Programmausführung erst zu einem späteren Zeitpunkt aufgedeckt wird.

Fehlhandlung

Ursache für das Vorliegen eines Fehlerzustands ist wiederum die vorausgegangene Fehlhandlung einer Person, wie z. B. eine fehlerhafte Programmierung durch den Entwickler. Der englische Begriff hierfür ist »Error«.

Fehlhandlungen können aus vielfältigen Gründen entstehen. Einige typische Fehlhandlungen bzw. Gründe (bzw. Grundursache) für Fehlhandlungen sind:

- Menschen machen Fehler – wir alle!
- Ein hoher Zeitdruck ist vorhanden – was in Softwareprojekten sehr häufig vorkommt.
- Die Komplexität der umzusetzenden Aufgabe, der Architektur, des Designs sowie des Programmtextes ist sehr hoch.
- Es gibt Missverständnisse zwischen den Projektbeteiligten – oft ein unterschiedliches Verständnis bzw. eine Auslegung der Anforderungen und weiterer Dokumente.
- Es gibt Missverständnisse über die Systeminteraktionen (systeminterne und systemübergreifende Schnittstellen), da deren Anzahl bei größeren Systemen oft sehr hoch ist.
- Die Komplexität der genutzten Technologien ist hoch oder es werden neue, bei den Projektbeteiligten (noch) unbekannte Technologien verwendet, die noch nicht richtig verstanden und daher falsch angewendet werden.
- Es liegt Unerfahrenheit oder unzureichende Ausbildung bei den Projektbeteiligten vor.

Eine Fehlhandlung einer Person führt zu einem Fehlerzustand in einem Programmstück, was zu einer Fehlerwirkung führt, die außen sichtbar ist und durch Testen aufgezeigt werden soll (s. Abb. 2–1, Debugging s. u.). Statische Tests (s. Kap. 4) können im Programmtext direkt Fehlerzustände aufdecken.

Fehlerwirkungen können aber auch durch Umweltbedingungen ausgelöst werden, wie Strahlung, Magnetismus etc., oder auch durch Umweltverschmutzung mit den entsprechenden Auswirkungen auf die Firmware und Hardware. Diese Art Fehler wird hier nicht behandelt.

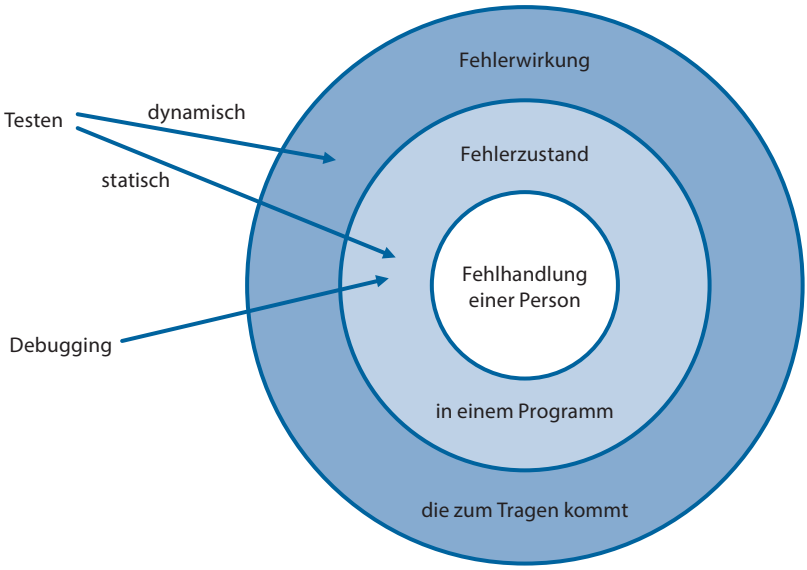

Abb. 2–1
Zusammenhang zwischen Fehlhandlung, Fehlerzustand und Fehlerwirkung

Falsch positives Ergebnis und falsch negatives Ergebnis

Nicht jedes unerwartete Ergebnis der Tests ist auch immer eine Fehlerwirkung. Es kann vorkommen, dass ein Testergebnis eine Fehlerwirkung anzeigt, obwohl der Fehlerzustand bzw. die Ursache für die Fehlerwirkung nicht im Testobjekt liegt. Dies wird als »falsch positives Ergebnis« (»false-positive Result«) bezeichnet. Die umgekehrte Situation kann ebenfalls vorkommen, dass Fehlerwirkungen nicht auftreten, obwohl die Tests diese hätten aufdecken sollen. Dies wird als »falsch negatives Ergebnis« (»false-negative Result«) bezeichnet. Bei jeder Auswertung von Testergebnissen ist somit zu beachten, ob eine der beiden Möglichkeiten vorliegt. Es gibt noch zwei weitere Ergebnisse: »richtig positiv« (Fehlerwirkung durch den Testfall aufgedeckt) und »richtig negativ« (erwartetes Verhalten bzw. Ergebnis des Testobjekts mit dem Testfall nachgewiesen). Nähere Ausführungen hierzu finden sich in Abschnitt 6.4.1.

Aus Fehlern lernen

Konnten Fehlerzustände aufgedeckt und die Fehlhandlungen ermittelt werden, die zu den Fehlerzuständen führten, dann lohnt es sich, mögliche Ursachen zu analysieren, um daraus zu lernen und in Zukunft gleiche oder ähnliche Fehlhandlungen zu vermeiden. Die so gewonnenen Erkenntnisse können zur Prozessverbesserung genutzt werden, um das Auftreten von zukünftigen Fehlhandlungen und damit Fehlerzuständen zu verringern oder zu verhindern.