Andreas Spillner • Tilo Linz

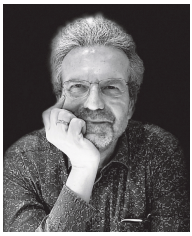
Basiswissen Softwaretest

Aus- und Weiterbildung zum Certified Tester
- Foundation Level
- nach ISTQB®-Standard

Das Standard-werk in 7. Auflage

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. Von 2019 bis 2023 war er Mitglied im Präsidium des Arbeitskreises Softwarequalität & -Fortbildung (ASQF e.V.). 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 30 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. Im Jahr 2023 wurde er zum Ehrenmitglied des GTB ernannt. Tilo Linz ist auch Autor des Buches »Testen in agilen Projekten« (dpunkt.verlag), das aufbauend auf dem vorliegenden »Basiswissen Softwaretest« das Testen in agilen Projekten behandelt.

ASQF • GI • TAV • GTB

2022 erhielten Tilo Linz und Andreas Spillner gemeinsam den Deutschen Preis für Software-Qualität.

Auszug aus der Begründung:

»Beide haben über viele Jahre wesentlich dazu beigetragen, dass Software-Qualitätssicherung als Fachdisziplin wahrgenommen wird und sich etabliert hat. Heute existiert das anerkannte und wertgeschätzte Berufsbild des qualifizierten Software-Quality Engineers. Dafür brauchte es nicht nur die Grundlagen – Inhalte, Ausbildung, Zertifizierung –, sondern auch ein ständiges ›Dranbleiben‹ und die Weiterentwicklung der Themen. Tilo Linz und Andreas Spillner stehen bis heute für diese Kontinuität.

Warum beide zusammen? ›Spillner-Linz‹ ist in der Test-Community fast zu einem geflügelten Wort geworden. Beide haben mit ihren unterschiedlichen Perspektiven das Berufsbild Software-Quality Engineer geprägt und nicht zuletzt mit ihrem Buch ›Basiswissen Softwaretest‹ das ISTQB-Curriculum in der Aus- und Weiterbildung methodisch untermauert.«

(*https://dpsq.de/*)

Andreas Spillner · Tilo Linz

Basiswissen Softwaretest

Aus- und Weiterbildung zum Certified Tester

Foundation Level nach ISTQB®-Standard

7., überarbeitete und aktualisierte Auflage

dpunkt.verlag

Andreas Spillner
andreas.spillner@hs-bremen.de

Tilo Linz
tilo.linz@imbus.de

Lektorat: Christa Preisendanz
Lektoratsassistenz: Julia Griebel
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Birgit Bäuerlein
Herstellung: Stefanie Weidner
Umschlaggestaltung: Eva Hepper, Silke Braun

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 978-3-98889-005-4
PDF 978-3-98890-139-2
ePub 978-3-98890-140-8

7., überarbeitete und aktualisierte Auflage 2024

Wieblinger Weg 17
69123 Heidelberg

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

Vorwort zur 7. Auflage

Testen & Agilität

Im Jahr 2002 erschien die erste Auflage dieses Buches. Ein Jahr zuvor wurde das »Agile Manifest« publiziert. Sowohl das Buch als auch das agile Vorgehen in der Softwareentwicklung haben in den beiden zurückliegenden Jahrzehnten eine bemerkenswerte Entwicklung zu verzeichnen gehabt: »Basiswissen Softwaretest« hat sich als meistverkauftes Buch zum Thema Softwaretest in deutscher Sprache als Standardwerk etabliert und die agile Vorgehensweise ist zu »der« Praktik in der IT-Industrie avanciert. Da war es naheliegend, dass in der Neuauflage des Buches – und des Lehrplans – viele der agilen Praktiken, die das Testen betreffen, aufgenommen wurden.

Testkompetenz ist weiterhin erforderlich.

Softwareentwicklung ist zur Teamarbeit geworden. Während früher einzelne Aufgaben von Fachleuten (Designer, Entwickler, Tester, …) umgesetzt wurden und jeder nur für seinen Bereich verantwortlich war, ist heute die Zusammenarbeit im Team gefragt. »Den« Tester und »den« Entwickler im engeren Sinne gibt es nicht mehr. Das Fachwissen und die Kompetenzen werden aber weiterhin benötigt, sind jedoch nicht mehr direkt an einzelne Personen gebunden. Im Buch wird an einigen Stellen noch von »Tester« und »Entwickler« gesprochen, gemeint ist aber eine Person mit Test- bzw. Entwicklungskompetenz, was sich etwas sperrig liest.

In der vorliegenden 7. Auflage des Buches haben wir den Inhalt umfassend überarbeitet, aktualisiert und an die aktuelle Version 4.0 des Lehrplans zum »ISTQB® Certified Tester – Foundation Level« aus dem Jahr 2023 angepasst.

»Certified Tester«-Ausbildungsschema

Das anerkannte und sehr erfolgreiche »Certified Tester«-Ausbildungsschema besteht aus den Ausbildungsstufen »Foundation«, »Advanced« und »Expert«. Ergänzt wird es durch Module für die Arbeit in agilen Teams sowie durch Spezialisierungsmodule. Details dazu finden sich auf den Webseiten des »International Software Testing Qualifications Board« [URL: ISTQB] und des »German Testing Board e.V.« ([URL: GTB], [URL: GTB Schema]). Der »Certified Tester« hat sich zu einem

Gütesiegel in der IT-Industrie entwickelt und ist heute der De-facto-Standard für die Ausbildung im Bereich Softwarequalitätssicherung und Softwaretest – in Deutschland und weltweit.

Beeindruckende Zahlen

»*Die Weiterbildung zum Certified-Tester ist international erfolgreich. Über 118.000 absolvierte Softwaretester-Prüfungen in Deutschland (Stand 09/2023) sind gute Gründe, stolz zu sein. Weltweit sind es mehr als 1.200.000 (Stand 06/2023)*« (aus [URL: GTB Schema]).

Damit hat sich die Anzahl der Prüfungen in den letzten fünf Jahren nahezu verdoppelt: 2018 waren es weltweit über 600.000, davon knapp 67.000 in Deutschland.

Wissen in der IT-Welt und an den Hochschulen gefragt

»*Studierenden und Auszubildenden wird durch das Angebot des GTB aktuelles, von der Industrie gefordertes Wissen vermittelt: Der wachsende Anteil an Stellenausschreibungen, in denen ein ISTQB® Certified Tester Zertifikat explizit gefordert wird, beweist eine bereits jetzt vorhandene hohe Durchdringung des Marktes. Zudem haben in den letzten Jahren knapp 600 Studierende in Deutschland erfolgreich das Certified Tester Zertifikat abgelegt*« (aus [URL: GTB Hochschulen]).

Der »Certified Tester« hat sich im Laufe der Jahre zu einem festen Bestandteil der Informatikausbildung an vielen Hochschulen entwickelt: Von A wie Aachen bis Z wie Zittau wird der Lehrstoff im deutschsprachigen Raum vermittelt. Welche Hochschulen aktuell entsprechende Lehrveranstaltungen anbieten oder planen, kann auf den Seiten des GTB nachgelesen werden [URL: GTB Hochschulen].

Ergänzende Literatur

Tilo und Andreas haben zwei weitere Bücher veröffentlicht, auf die hier hingewiesen werden soll, da sie eine gute Ergänzung bzw. Vertiefung darstellen. Andreas – als Norddeutscher – würde sagen: »Butter bei die Fische«, denn beide Bücher vertiefen die im Basiswissen-Buch vorhandenen Beschreibungen zur Agilität bzw. zu Testverfahren.

Buch: Testen in agilen Projekten

In seinem aktuellen Buch »Testen in agilen Projekten – Methoden und Techniken für Softwarequalität in der agilen Welt« (3. Auflage, 2024) zeigt Tilo, wie das Testen in agile Projekte integriert werden kann. Das Buch deckt damit die ISTQB®-Lehrpläne zum agilen Testen ab.

Buch: Lean Testing für C++-Programmierer

Andreas hat zusammen mit Ulrich Breymann[1] das Buch »Lean Testing für C++-Programmierer – Angemessen statt aufwendig testen« geschrieben. In dem Buch werden alle für den Komponententest relevanten Testverfahren des ISO-Standards 29119 ausführlich beschrieben. Die Vorgehensweisen zum Testfallentwurf werden konkret mit den entspre-

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

chenden C++-Programmtexten und den zugehörigen Testfällen dargestellt. Dabei sind die Programmbeispiele so gehalten, dass sie auch ohne C++-Kenntnisse verständlich sind.

Auf beide Bücher wird in diesem Buch immer wieder verwiesen ([Linz 24], [Spillner 16]), um den Leserinnen und Lesern weiterführende Informationen zu bieten. Vielleicht sind wir mit den vielen Verweisen etwas über das Ziel hinausgeschossen, dafür bitten wir um Nachsicht – aber ein wenig Werbung für die eigene Arbeit wird hoffentlich noch erlaubt sein, oder?

Basiswissen

Von Anfang an haben wir den ersten Teil unseres Buchtitels »Basiswissen« ernst genommen und bewusst keine Themen behandelt, die sich in der Praxis erst noch »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 die entsprechende aktuelle Literatur zu diesen und anderen Themen.

Was hat sich geändert?

Aber auch Grundlagen unterliegen dem Wandel und sind aktuell zu halten. Ebenso haben die etablierten agilen Praktiken mit Bezug zum Softwaretest ihren Weg in das Buch gefunden. In der vorliegenden 7. Auflage wurden die Inhalte umfassend überarbeitet, ergänzt und aktualisiert.

Folgende Themen wurden neu aufgenommen oder ausführlicher behandelt:

- Whole-Team-Ansatz
- Test-First-Ansatz
- Shift-Left
- Retrospektive
- Testen im Kontext von DevOps
- Abnahmetestgetriebene Entwicklung (ATDD)
- User Stories
- Akzeptanzkriterien
- Iterations- und Releaseplanung bei agilen Projekten
- Testpyramide und Testquadrant

Exkurse sind nicht Teil des Lehrplans.

Bei der Überarbeitung des ISTQB®-Lehrplans wurden einige Themen auf höhere Ausbildungsstufen verschoben oder ganz weggelassen und sind somit nicht mehr Bestandteil des »Foundation Level«-Lehrplans. Wir haben diesen Schritt nicht strikt umgesetzt, sondern uns entschieden, die für die Praxis wichtigen Teile im Buch zu belassen. Diese sind als Exkurse farblich hervorgehoben (blaue Schrift). Wer das Buch nur zur Prüfungsvorbereitung nutzt, kann die Exkurse einfach überspringen.

Standard-Nachschlagewerk

Aus vielen Gesprächen mit Leserinnen und Lesern wissen wir, dass unser Buch als Nachschlagewerk für die tägliche (Test-)Arbeit genutzt wird. Deshalb haben wir neben den Inhalten des Lehrplans weitere grundlegende Testverfahren – als Exkurse – aufgenommen (z.B. kombinatorisches Testen, »Pairwise Testing«).

Das durchgehende Fallbeispiel und das Literaturverzeichnis wurden aktualisiert. Das Verzeichnis der Normen und Standards wurde ebenfalls überarbeitet. Die Angaben zu den Internetseiten (URLs) sind auf den aktuellsten Stand gebracht worden.

Webseite

Um die Leserinnen und Leser über zukünftige Aktualisierungen des Lehrplans und des Glossars zu informieren, betreiben wir die Internetseite »Softwaretest Knowledge« [URL: Softwaretest Knowledge]. Auf dieser Seite werden auch notwendige Korrekturen zum Buchtext aufgeführt. Neben Übungsaufgaben zu den einzelnen Buchkapiteln findet sich dort auch eine Cross-Referenz-Tabelle, in der zu jedem Lernziel des Lehrplans die entsprechenden Abschnitte des Buches aufgeführt sind, in denen das Lernziel ausführlich behandelt wird.

Ebenfalls sind dort das Vorwort zur 1. Auflage des Buches und die Geleitworte von Dorothy Graham, David Parnas und Martin Pol zu finden.

Danksagung

Erfolg hat meist viele Väter und Mütter – so auch hier. Allen Kolleginnen und Kollegen des GTB und des ISTQB® sei an dieser Stelle herzlich gedankt. Ohne ihr Engagement hätte das »Certified Tester«-Ausbildungsschema nicht den geschilderten Erfolg und die weltweite Akzeptanz erreicht. Ebenso möchten wir uns für die vielen Anmerkungen und Rezensionen unserer Leserinnen und Leser bedanken, die uns für unsere Arbeit am Buch sehr motiviert und zur Qualitätssteigerung beigetragen haben. Unserer Lektorin Christa Preisendanz und dem gesamten dpunkt.team danken wir für die langjährige und sehr gute Zusammenarbeit.

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

Andreas Spillner und *Tilo Linz*
Bremen, Möhrendorf
März 2024

Inhaltsübersicht

Inhaltsverzeichnis

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. Systeme mit künstlicher Intelligenz finden zurzeit eine rasend schnelle Verbreitung und wirken sich bereits heute in ganz vielen Bereichen unseres Lebens aus. 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 Qualitätsverbesserung 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 sowie Mitglieder in agilen Teams, 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 Trainings-

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

anbieter, Testexperten aus Industrie- und Beratungsunternehmen sowie Hochschullehrende organisiert. Wichtige Kompetenz bringen 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 besteht aus den Ausbildungsstufen »Foundation«, »Advanced« und »Expert«. Es wird ergänzt um Module für die Arbeit in agilen Teams sowie Spezialistenmodule. Details dazu finden sich auf der Webseite des ISTQB® [URL: ISTQB] und des GTB [URL: GTB]. Die Grundlagen zum Softwaretest – sowohl bei klassischer als auch bei agiler Entwicklung – 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. Ergänzend bietet das Schema »Specialist Module«, die spezielle oder domänenspezifische Methoden, Techniken und Prozesse vermitteln. Beispiele sind »Sicherheitstester«, »Usability Testing« und »Certified Tester AI Testing«. Die dritte Stufe, das »Expert Level«-Zertifikat, richtet sich an erfahrene, professionelle Softwaretester und umfasst die Module »Improving the Test Process« und »Test Management«.

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 auch auf erforderliche Kompetenzen beim Testen eingegangen.

Testen im Softwarelebenszyklus

Kapitel 3 stellt in der Softwareentwicklung gebräuchliche Lebenszyklusmodelle (sequenziell, iterativ-inkrementell, agil) kurz vor und erläutert, welche Rolle das Testen im jeweiligen Modell spielt. Wichtige Elemente agiler Vorgehensweisen (Backlogs, Whole-Team-Ansatz, User Stories, Continuous Integration etc.) werden erläutert und es wird aufgezeigt, welchen Einfluss sie auf die Gestaltung der Testaktivitäten haben. Die verschiedenen Teststufen und Testarten werden ausführlich erklärt und auf die Unterschiede beim funktionalen und nicht funktionalen Test eingegangen. Das Thema Regressionstest wird ebenfalls angesprochen. Wichtige Ansätze zur Verbesserung und Automatisierung

der Softwareentwicklung und des Testprozesses (CI/CD, frühes Testen/Shift-Left, testgetriebene Entwicklung, DevOps) werden kompakt vorgestellt und erläutert.

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, Rollen und Aufgaben beim Testmanagement zu berücksichtigen sind und welche Anforderungen an die Qualifikation der Mitarbeiter bestehen, wird in Kapitel 6 diskutiert. Welche Elemente zu einer Teststrategie gehören und wie diese durch eine fundierte Testplanung, Teststeuerung und Testüberwachung umgesetzt wird, wird ausführlich erklärt. Wichtige Verfahren zur Schätzung von Testaufwand und Testkosten werden vorgestellt und es wird erläutert, welchen Beitrag das Testen leistet, um Risiken zu mindern, und wie risikobasiertes Testen funktioniert. Abschließend werden Anforderungen an das Fehlermanagement und Konfigurationsmanagement sowie das Thema Wirtschaftlichkeit des Testens besprochen.

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.

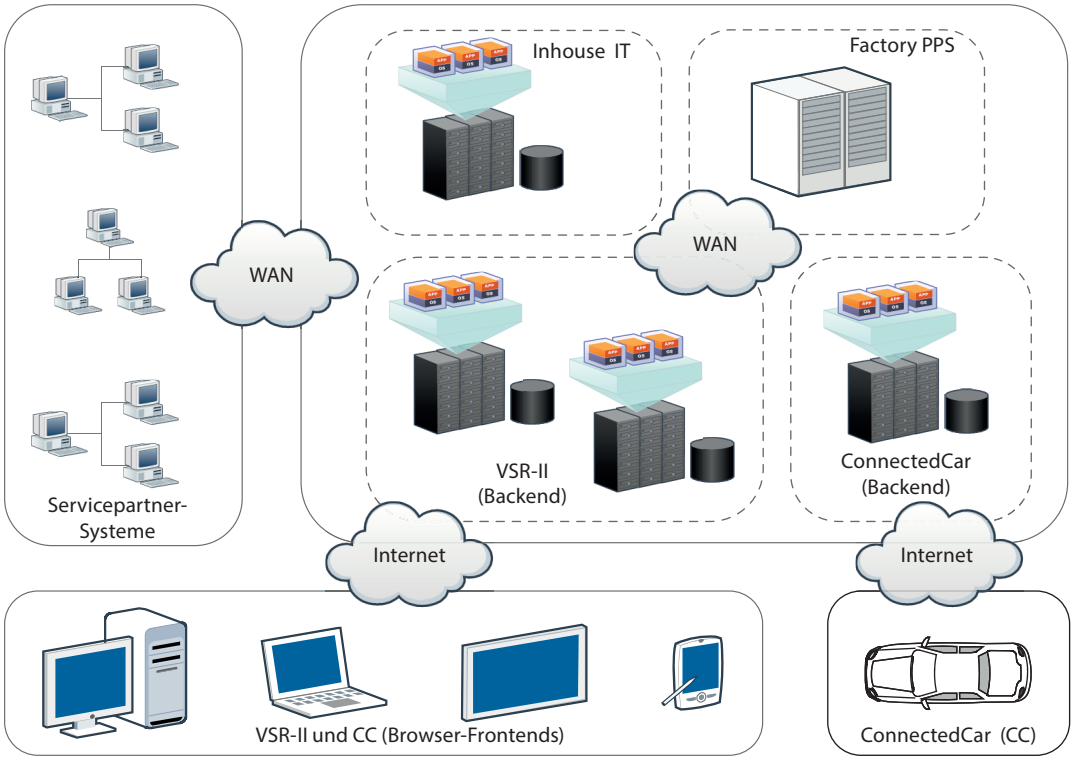


Abb. 1–1 *VSR-II in der Übersicht*

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, unterschiedliche Denkweisen und erforderliche Kompetenzen beim Testen eingegangen.

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 (bzw. Kontrolle der Qualität) und verringert die Risiken beim Einsatz der Software, da mögliche Fehler während des Testens – und damit vor dem Einsatz des Softwaresystems – aufgedeckt werden können. Testen trägt somit dazu bei, die vereinbarten Ziele (s. Abschnitt 2.1.2) innerhalb des vereinbarten Umfangs zu erreichen sowie die festgelegten Zeit-, Qualitäts- und Budgetvorgaben einzuhalten.

Der Beitrag des Testens zum Erfolg ist nicht auf die Aktivitäten des Testteams – wenn es ein solches Team überhaupt gibt – beschränkt. Jeder am Projekt Beteiligte – ebenso die Stakeholder – kann und soll seine Fähigkeiten zum Testen einsetzen, damit das Projekt erfolgreich und mit der gewünschten Qualität durchgeführt und beendet wird.

Das (statische und dynamische, s. Kap. 4 und 5) Testen von Komponenten, Systemen und der zugehörigen Dokumentation erkennt Fehler (Fehlerzustände bzw. Fehlerwirkungen) in der Software. Dem Testen von Software kommt 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 mehr als nur die Ausführung von Testfällen. Neben dieser eher technischen Aktivität sind alle weiteren Aktivitäten rund um das Testen sorgfältig zu planen und zu verwalten. Auch

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.).

ist der zu veranschlagende Aufwand für das Testen vorab zu schätzen und dann zu überwachen und ggf. anzupassen. Diese unterschiedlichen Aktivitäten werden oft als ein Prozess – der Testprozess – zusammengefasst. 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. 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 den Abschnitten 2.3 und 6.3.

Testen im Softwareentwicklungslebenszyklus

Die Testaktivitäten werden je nach Entwicklungslebenszyklus der Software unterschiedlich organisiert und durchgeführt. Testen ist in verschiedenen Abschnitten auch ein Mittel zur direkten Bewertung der Qualität eines Testobjekts. So kann beispielsweise der Übergang zu einer nächsten Phase der Softwareentwicklung oder die Planung der nächsten Iteration von den Ergebnissen der Tests abhängen (z.B. Entscheidung über die Freigabe des Testobjekts).

Testen ist eine intellektuell herausfordernde Tätigkeit.

Viele der Testaktivitäten werden durch Werkzeuge unterstützt oder auch erst durch sie ermöglicht (s. Kap. 7). Testen ist aber eine weitgehend intellektuelle Tätigkeit, die von den ausführenden Personen Fachwissen, analytische Fähigkeiten und kritisches Hinterfragen sowie Systemdenken erfordert (s. [Myers 11], [Roman 18]).

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 aus der Perspektive der späteren Nutzer

Eine Aufgabe des Testens ist somit die Sicherstellung, dass die Anforderungen der späteren Nutzer während des gesamten Entwicklungszyklus berücksichtigt werden. Eine repräsentative Gruppe von Nutzern in das Entwicklungsprojekt einzubinden, ist sicherlich eine gute Alternative, die aber in der Regel aufgrund der hohen Kosten und der mangelnden Verfügbarkeit geeigneter Nutzer meist nicht möglich ist. Testen bein-

haltet somit auch die Validierung des Systems (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. Der englische Begriff hierfür ist »Error«.

Fehlhandlungen können aus vielfältigen Gründen entstehen. Einige typische Fehlhandlungen bzw. Gründe (bzw. Hauptursachen) 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.
- Oder die Projektbeteiligten sind abgelenkt, unkonzentriert oder einfach müde.

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.

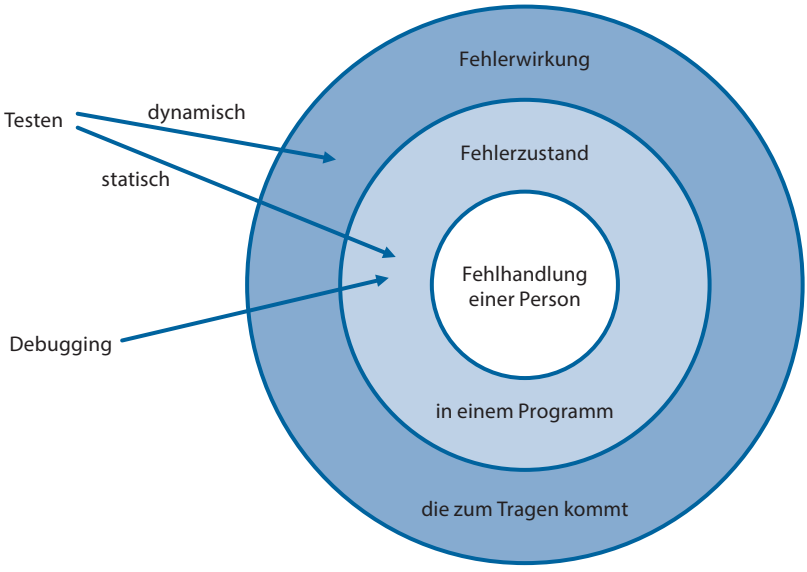


Abb. 2–1
Zusammenhang zwischen Fehlhandlung, Fehlerzustand und Fehlerwirkung

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[3], 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.

Beispiel: Unklare Anforderung als Ursache von Softwarefehlern

Über das Teilsystem VSR-II-*EasyFinance* kann sich der Kunde verschiedene Optionen zur Finanzierung seines neuen Fahrzeugs berechnen und vorschlagen lassen. Der bei einer Kreditfinanzierung vom System verwendete Zinssatz wird aus einer im System hinterlegten Zinssatztabelle entnommen. Für Fahrzeuge, die im Rahmen von werblichen Sonderaktionen angeboten und verkauft werden, können davon abweichend andere Zinssätze gelten.

Im neuen VSR-II soll zusätzlich folgende Anforderung realisiert werden:

REQ: Wenn der Kunde der »Online-Bonitätsprüfung« zugestimmt und diese absolviert hat, dann reduziert VSR-II-*EasyFinance* den Kreditzinssatz gemäß der im System hinterlegten Zinssatz-Bonus-Tabelle.

Der Autor der Anforderung hat allerdings vergessen klarzustellen, dass diese Reduktion nicht erlaubt ist, wenn es um ein Fahrzeug geht, das in einer Sonderaktion verkauft wird. Entsprechend wurde dieser Fall in den Tests des ersten Release nicht überprüft. Kunden aus Sonderaktionen erhielten daher online Kreditangebote mit zu niedrigem Zinssatz angeboten und beschwerten sich, als die erste Kreditabrechnung einen höheren Zinssatz auswies.

3. Im Lehrplan als Grundursachenanalyse bezeichnet, s.a. Glossareintrag »Grundursache«.

2.1.2 Testbegriff

Testen ist nicht Debugging.

Um einen Fehlerzustand korrigieren zu können, muss der Fehlerzustand im Softwareprodukt lokalisiert werden. Bekannt ist zunächst nur seine Wirkung, aber nicht die genaue Stelle in der Software, die den Fehlerzustand beinhaltet. Das Lokalisieren und Beheben des Fehlerzustands ist Aufgabe des für die betroffene Codekomponente verantwortlichen Softwareentwicklungsteams und wird auch als »Debugging« (Fehlerbereinigung, Fehlerkorrektur) bezeichnet. Debugging und Testen werden oft gleichgesetzt, sind aber zwei völlig unterschiedliche und getrennte Aufgaben. Während Debugging das Ziel hat, Defekte bzw. Fehlerzustände zu lokalisieren, ist es Aufgabe des Tests, Fehlerwirkungen gezielt aufzudecken (s.a. Abb. 2–1).

Fehlernachtest

Die Behebung des Fehlerzustands führt zur Qualitätsverbesserung des Produkts, da in den meisten Fällen keine neuen Fehlerzustände durch die Änderung hinzugefügt werden. Tests, die prüfen, ob die Korrekturmaßnahmen erfolgreich waren, werden als Fehlernachtests bezeichnet. Oft werden dieselben Personen, die auch die Tests zur Aufdeckung der betreffenden Fehler durchgeführt haben, mit den Fehlernachtests beauftragt oder auch – insbesondere bei agiler Entwicklung – die Person, die den Fehler im Code korrigiert. Wenn ein automatisierter Test vorhanden ist, dann ist der Fehlernachtest nichts anderes als das erneute Ausführen des/der automatisierten Tests, die jetzt aber mit »passed« durchlaufen sollten, statt mit »failed«.

In der Praxis kommt es aber leider vor, dass durch die Korrektur eines Fehlerzustands ein oder sogar mehrere neue Fehlerzustände »hineinprogrammiert« werden. Der neue Fehlerzustand kann dann bei einer ganz anderen Eingabekonstellation zur Wirkung kommen. Solche unbeabsichtigten Seiteneffekte erschweren den Test und bedingen, dass nach Änderungen nicht nur die Tests zu wiederholen sind, die den Fehlerzustand zur Wirkung gebracht haben, sondern auch weitere Tests, die mögliche Seiteneffekte aufdecken könnten (Regressionstest, s. Abschnitt 3.6.3).

Testziele

Testen – und hier sind sowohl statisches als auch dynamisches Testen gemeint – verfolgt mehrere Ziele:

- Die qualitative Bewertung von Arbeitsergebnissen wie Anforderungsspezifikation, User Stories, Design und Programmtext
- Der Nachweis, dass alle spezifischen Anforderungen vollständig umgesetzt sind und dass das Testobjekt so funktioniert, wie es die Nutzer und andere Interessenvertreter (Stakeholder) erwarten
- Informationen zur Verfügung stellen, damit die Stakeholder die Qualität des Testobjekts fundiert einschätzen können und somit Vertrauen in die Qualität des Testobjekts schaffen[4]