Andreas Spillner · Thomas Roßner · Mario Winter · Tilo Linz

Praxiswissen Softwaretest

Testmanagement

Aus- und Weiterbildung
zum Certified Tester

- Advanced Level
- nach ISTQB-Standard

dpunkt.verlag

iSQI International Software Quality Institute

4., überarbeitete und erweiterte Auflage

Andreas Spillner ist Professor für Informatik an der Hochschule Bremen, Fakultät für Elektrotechnik und Informatik. Er war über 10 Jahre Sprecher der Fachgruppe TAV »Test, Analyse und Verifikation von Software« der Gesellschaft für Informatik e.V. (GI) und bis Ende 2009 Mitglied im German Testing Board e.V. 2007 ist er zum Fellow der GI ernannt worden. Seine Arbeitsschwerpunkte liegen im Bereich Softwaretechnik, Qualitätssicherung und Testen.

Thomas Roßner ist Mitgründer der imbus AG und in deren Vorstand verantwortlich für Forschung und Technologie des Unternehmens. In dieser Funktion leitete er in den vergangenen Jahren mehrere internationale Forschungsprojekte, u.a. zum Thema Softwarezuverlässigkeit und modellbasiertes Testen. Darüber hinaus arbeitet er aktiv in Testmanagementprojekten und Beratungsprojekten zum Thema Testprozessverbesserung.

Mario Winter ist Professor am Institut für Informatik der Fachhochschule Köln und dort Mitglied des Forschungsschwerpunktes »Software-Qualität«. Er ist Mitglied im German Testing Board e.V. und war von 2003 bis Anfang 2011 Sprecher der Fachgruppe »Test, Analyse und Verifikation von Software« im Fachbereich Softwaretechnik der Gesellschaft für Informatik (GI). Seine Lehr- und Forschungsschwerpunkte sind Softwareentwicklung und Projektmanagement, insbesondere die modellbasierte Entwicklung und Qualitätssicherung von Software.

Tilo Linz ist Vorstand der imbus AG, eines führenden Dienstleisters für Softwaretest. Er ist Leiter des German Testing Board e.V. und war von 2002 bis 2005 Vorsitzender des ISTQB. Zu seinen Arbeitsschwerpunkten zählen die Themen Berufsbild und Ausbildung im Softwaretest sowie die Optimierung von Softwaretestprozessen.

Andreas Spillner · Thomas Roßner · Mario Winter · Tilo Linz

Praxiswissen Softwaretest – Testmanagement

Aus- und Weiterbildung zum Certified Tester – Advanced Level nach ISTQB-Standard

4., überarbeitete u. erweiterte Auflage

dpunkt.verlag

Andreas Spillner
Andreas.Spillner@hs-bremen.de

Thomas Roßner
thomas.rossner@imbus.de

Mario Winter
mario.winter@fh-koeln.de

Tilo Linz
tilo.linz@imbus.de

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz & Herstellung: Birgit Bäuerlein
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: Media-Print Informationstechnologie, 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
Buch 978-3-86490-052-5
PDF 978-3-86491-515-4
ePub 978-3-86491-516-1

4., überarbeitete und erweiterte Auflage 2014

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Vorwort

Testmanagement reloaded

Nach den bisherigen Auflagen des Buches aus den Jahren 2006, 2008 und 2011 liegt nun die vierte Auflage von »Praxiswissen Softwaretest – Testmanagement« vor. Anlass war die Aufteilung und Erweiterung des Lehrplans zum ISTQB Certified Tester® »Advanced Level« in der 2012 veröffentlichten Version. Darüber hinaus haben auch die 2011 vom ISTQB veröffentlichten Lehrpläne zu den »Expert Level«-Modulen »Testmanagement« und »Improving the Testing Process« die erneute Überarbeitung des Buches motiviert. Überdies verdienen aktuelle Entwicklungen im Bereich internationaler Standards zum Softwaretest – Stichwort ISO 29119 – ihre Berücksichtigung. Und nicht zuletzt war auch die dritte Auflage des Buches erfreulich schnell vergriffen.

Was ist neu?

Der in der vierten Auflage abermals gestiegene Umfang sowie die Neustrukturierung des Buches sind hauptsächlich der neuen Version des Lehrplans geschuldet. Dieses Buch bezieht sich nun auf den Lehrplan zum ISTQB Certified Tester® – Advanced Level Testmanager von 2012, auch wenn es an einigen Stellen darüber hinausgeht und damit den Einstieg in die oben angesprochenen »Expert Level«-Module erleichtern kann.

Seit 2010 wurden im Glossar des ISTQB einige Begriffe ergänzt und bestehende Definitionen verfeinert. Auch diese Änderungen haben wir im Buch nachvollzogen. Zudem wurde das Quellenverzeichnis aktualisiert und neuere Veröffentlichungen aufgenommen sowie aktuelle Versionen der Standards berücksichtigt. Ebenso wurden die Angaben zu den Internetseiten (URLs) kontrolliert und ggf. aktualisiert bzw. ergänzt.

Webseite

Unsere Internetseite [URL: softwaretest-knowledge] informiert Sie weiterhin über Aktualisierungen, sei es bzgl. des Lehrplans, des Glossars oder aber ggf. hinsichtlich notwendiger Korrekturen und Ergänzungen zum Buchtext. Auf der Internetseite stehen die Vorworte der bisherigen Auflagen sowie das Geleitwort von Anton Schlatter zur 1. Auflage zur Verfügung.

Verbesserte Durchgängigkeit und Lesbarkeit

Wir hoffen, bei der notwendigen grundlegenden Überarbeitung des Buches die Durchgängigkeit und Lesbarkeit weiter verbessert zu haben, und freuen uns auf Ihr Feedback zum neuen »Praxiswissen Softwaretest – Testmanagement«!

Danksagung

An erster Stelle möchten wir wieder die vielen Leserinnen und Leser ansprechen, die uns in bester Tradition als Tester Fehler, Unstimmigkeiten und Verbesserungspotenziale des Buches gemeldet haben. Ihnen kommt unser ganz besonderer Dank zu! Weiterhin gilt unser Dank den Mitarbeiterinnen und Mitarbeitern des dpunkt.verlags, die einmal mehr dafür gesorgt haben, dass unsere Ausführungen in optisch und haptisch ansprechender Form zu Ihnen gelangen. Und nicht zuletzt gebührt unseren Familien großer Dank dafür, dass sie uns erneut Zeit und Raum gegeben haben, die nun vorliegende Auflage dieses Buches anzufertigen.

Wie schon in den vorangegangenen Auflagen wünschen wir Ihnen gutes Gelingen bei der Umsetzung der Testansätze in Ihrer Praxis und – sollte das Buch die Grundlage für die Vorbereitung der Prüfung zum »Certified Tester – Advanced Level Testmanager« sein – viel Erfolg bei der Prüfung!

Andreas Spillner, Thomas Roßner, Mario Winter, Tilo Linz
Bremen, Möhrendorf, Wuppertal
April 2014

Inhaltsübersicht

Inhaltsverzeichnis

Anhang

1 Einleitung

Große Abhängigkeit von Software

Unser Alltag ist wie nie zuvor abhängig von Software und softwarebasierten Systemen. Es gibt kaum noch Geräte, Maschinen oder Anlagen, deren Funktion oder Steuerung nicht über Software bzw. Softwareanteile realisiert wird. Aber auch Verwaltungsvorgänge in Industrie und Staat werden durch oft komplexe IT-Systeme getragen. Die Verwaltung von Versicherungspolicen, das Mautsystem »TollCollect«, biometrische Merkmale in Pass und Personalausweis oder die elektronische Gesundheitskarte sind hierfür Beispiele.

Testen von Software ist eine eigenständige Berufsdisziplin.

Diese starke Abhängigkeit von Software erfordert immer höhere Investitionen in qualitätssichernde Maßnahmen, damit die IT-Systeme möglichst zuverlässig ihre Aufgaben erfüllen. Das Testen von Software hat sich vor diesem Hintergrund zu einer spezialisierten, eigenständigen Fachrichtung und Berufsdisziplin der Informatik entwickelt. Dies belegen auch die Ergebnisse der 2011 in Deutschland, Österreich und der Schweiz durchgeführten Umfrage zu Entwicklungen und Trends im Bereich Testen und Qualitätssicherung in Unternehmen. Über 80% der Befragten befürworten spezielle Weiterbildungen im Bereich der Qualitätssicherung [URL: Softwaretest-Umfrage].

Testmanagement

Innerhalb der Disziplin Softwaretest hat das Thema »Testmanagement« besondere Bedeutung. Das Testmanagement umfasst klassische Methoden des Projektmanagements und des Risikomanagements sowie das Wissen um den zweckmäßigen Einsatz wohldefinierter Testentwurfsverfahren. Mit diesem Handwerkszeug ausgerüstet, kann der Testmanager[1] geeignete Maßnahmen zielgerichtet auswählen und umsetzen, die sicherstellen, dass eine bestimmte Mindestqualität des Produkts erreicht wird. Er verfolgt dabei ein ingenieurmäßiges Vorgehen.

1. Wir verwenden im Buch die männliche Form und wollen damit Frauen selbstverständlich nicht ausschließen bzw. ausgrenzen.

Ausbildung für Testmanager

Während die Ausbildung zum Projektmanager seit Langem etabliert ist und eine Vielzahl von Studiengängen, Ausbildungsprogrammen und Spezialliteratur existiert (s. beispielsweise [Hindel 09], [Spitczok von Brisinski 2010], [Pichler 07] oder [Pichler 11]), waren die Ausbildungsinhalte zum »Softwaretestmanager« lange Zeit kaum definiert oder gar standardisiert. Angesichts der steigenden Verantwortung, die Testmanager im Rahmen ihrer Tätigkeit übernehmen, war das ein unerfreulicher Zustand.

ISTQB Certified Tester – Advanced Level – Testmanager

Mit dem »ISTQB Certified Tester – Advanced Level – Testmanager« steht mittlerweile ein international anerkanntes Ausbildungsschema zur Verfügung, das auch für den Beruf des Testmanagers Lehrinhalte und Qualifizierungsmodule definiert. Das vorliegende Buch »Praxiswissen Softwaretest – Testmanagement« vermittelt diese Lehrinhalte und kann als Lehrbuch bei der Vorbereitung auf die entsprechende Zertifizierung dienen.

Foundation Level

Das »ISTQB Certified Tester«-Qualifizierungsprogramm ist dreistufig aufgebaut. Die Grundlagen des Softwaretests sind im Lehrplan »Foundation Level« beschrieben [URL: GTB CTF]. Dieser Lehrstoff ist im Buch »Basiswissen Softwaretest« (s. [Spillner 12]) ausführlich dargestellt.

Advanced Level

Der »Advanced Level«-Lehrplan [URL: GTB CTA] umfasst weiterführende Kenntnisse im Prüfen und Testen von Software und zeigt drei Spezialisierungsmöglichkeiten auf:

- die vertiefte Behandlung von verschiedenen Blackbox- und Whitebox-Testentwurfsverfahren in den »Advanced Level«-Modulen »Technical Test Analyst« und »Test Analyst« sowie
- die vertiefte Darstellung von Methoden und Techniken des Testmanagements im Modul »Testmanager«.

Diese Aufteilung entspricht auch der Struktur des Lehrstoffs, wie sie von vielen akkreditierten Weiterbildungsanbietern vorgenommen wird. Da der Lehrstoff des »Advanced Level« sehr umfassend ist, wird dieser im vorliegenden Buch nicht komplett behandelt, sondern ausschließlich das Modul »Advanced Level – Testmanager«.

Expert Level

Die dritte Stufe, »Expert Level«, richtet sich an erfahrene, professionelle Softwaretester und besteht aus einer Reihe von Modulen zu unterschiedlichen Spezialthemen. Seit 2011 sind die Lehrpläne zu den Modulen »Improving the Testing Process – Implementing Improvement and Change« [Bath 14] und »Test Management – Managing Testing, Testers, and Test Stakeholders« veröffentlicht [URL: GTB CTE]. Geplant sind weitere Themen wie Test Automation, Security Testing, TTCN-3 [URL: TTCN-3] u.a.

Add-on-Erweiterungen

Es ist geplant, aktuelle Themen oder branchenorientierte Spezialgebiete im »Certified-Tester«-Schema bedarfsorientiert und kurzfristig im Rahmen sogenannter »Extensions« der Foundation- und Advanced-Level-Lehrpläne zu berücksichtigen. Als erster Extension-Baustein wird in 2014 »Foundation Level Extension Syllabus Agile Tester« veröffentlicht.

International Software Testing Qualifications Board (ISTQB)

Das »ISTQB« [URL: ISTQB] sorgt weltweit für die Einheitlichkeit und Vergleichbarkeit der Lehr- und Prüfungsinhalte unter allen beteiligten Ländern. In ihm sind mittlerweile knapp 50 nationale Initiativen und Verbände aus über 70 Ländern zusammengeschlossen. Weitere nationale Boards werden hinzukommen.

Nationale Testing Boards

Die nationalen Testing Boards sind in einem oder mehreren Ländern als unabhängige Expertengremien dafür zuständig, Ausbildung (Akkreditierung der Weiterbildungsanbieter) und Prüfungen (Zertifizierung durch eine unabhängige Institution) in den jeweiligen Ländern und Landessprachen zu ermöglichen und die Einhaltung der ISTQB-Standards zu überwachen.

Basiswissen wird vorausgesetzt.

Die drei ISTQB-Ausbildungsstufen bauen aufeinander auf. Das vorliegende Buch »Praxiswissen Softwaretest – Testmanagement« setzt den Stoff des »Foundation Level« voraus. Lesern, die neu in das Thema Softwaretest einsteigen, wird daher empfohlen, sich den Stoff des »Foundation Level« anzueignen. Dies kann durch den Besuch eines akkreditierten Seminars erfolgen oder durch das Durcharbeiten des Buches »Basiswissen Softwaretest« (s. [Spillner 12]). Im vorliegenden Buch werden lediglich knappe Wiederholungen der wichtigsten Grundlagen geboten.

1.1 Basiswissen – komprimiert

Im Folgenden wird der Inhalt des Lehrplans »Foundation Level« und somit auch das Buch »Basiswissen Softwaretest« kurz zusammengefasst.

Maßnahmen zur Verbesserung der Softwarequalität

Es gibt eine Vielzahl von Ansätzen und Vorschlägen, die Qualität der Software durch vorbeugende (konstruktive) Maßnahmen und den Einsatz von prüfenden (analytischen) Verfahren und Methoden zu verbessern. Zu den wichtigsten Maßnahmen gehören:

- Definierte Softwareentwicklungsprozesse (inkl. agiler Vorgehensweisen), die zu einer strukturierten und nachvollziehbaren Erstellung der Softwaresysteme beitragen.
- Ein wohldefinierter Testprozess und ein geordnetes Änderungs- und Fehlermanagement als Voraussetzungen, um die Testarbeiten wirtschaftlich und wirksam durchzuführen.

- Verwendung von Metriken und Qualitätskennzahlen, die helfen, Softwareprodukte und Entwicklungsprozesse objektiv zu bewerten, Verbesserungspotenziale aufzudecken und die Wirksamkeit von Korrektur- oder Verbesserungsmaßnahmen zu überprüfen.
- Der Einsatz von formalen Methoden, die eine präzise Formulierung der Entwicklungsdokumente und damit deren Überprüfbarkeit bzw. Auswertung durch Werkzeuge ermöglichen.
- Methoden zur systematischen Ermittlung und Durchführung von Testfällen, die für eine effiziente Erkennung von Fehlern und Unstimmigkeiten in den entwickelten Programmen sorgen.
- Methoden zur statischen Prüfung, in erster Linie Reviews, durch die Fehler und Mängel frühzeitig in den erstellten Entwicklungsdokumenten aufgedeckt werden.

Qualitätsziele und Qualitätsmerkmale

Testmanager müssen diese Methoden, Techniken und Prozesse beherrschen oder zumindest kennen, um im Projektverlauf die der jeweiligen Situation angemessenen Maßnahmen auswählen und anwenden zu können. Die Eignung von qualitätssichernden Maßnahmen ist aber auch abhängig von den jeweils gesetzten Qualitätszielen. Das geforderte Qualitätsniveau kann dabei anhand verschiedener Qualitätsmerkmale definiert werden. Einen Katalog solcher Qualitätsmerkmale (z.B. Funktionalität, Zuverlässigkeit oder Benutzbarkeit) definiert die Norm [ISO 9126][2] (s.a. [ISO 25010]).

Testorakel

Wann liegt ein Defekt oder Fehler vor und was ist unter diesen Begriffen zu verstehen? Eine Situation oder ein Ergebnis kann nur dann als fehlerhaft eingestuft werden, wenn vorab festgelegt wurde, wie die erwartete, korrekte Situation bzw. das erwartete Ergebnis aussieht. Wird eine →[3] Abweichung zwischen dem beobachteten Istverhalten und dem erwarteten Sollverhalten festgestellt, liegt ein Fehler vor. Um Sollwerte bzw. das Sollverhalten zu ermitteln, ist eine Testbasis bzw. ein sogenanntes Testorakel als Informationsquelle erforderlich. Anforderungsdokumente, eine formale Spezifikation oder auch das Benutzungshandbuch sind Beispiele für solche Informationsquellen.

Fehlerbegriff

Der Begriff »Fehler« ist unpräzise. Es ist zwischen Fehlhandlung (engl. *error*), Fehlerzustand (engl. *fault*) und Fehlerwirkung (engl. *failure*) zu unterscheiden. Eine Fehlhandlung einer Person führt beispielsweise zu einer fehlerhaften Programmierung. Dadurch enthält das Programm einen Fehlerzustand, der zu einer »von außen« sichtbaren

2. Die ISO-Norm 9126 ist durch die neue ISO-Norm 25010 abgelöst worden, wird aber zurzeit noch im »Foundation Level«-Lehrplan referenziert.
3. Mit dem Pfeilsymbol werden Begriffe gekennzeichnet, die im Glossar im Anhang des Buches erläutert werden.

Fehlerwirkung führen kann, aber nicht zwangsläufig führen muss. Meist kommt ein Fehlerzustand erst bei nicht alltäglichen Situationen zum Tragen, z. B. wirkt sich eine fehlerhafte Berechnung des Schaltjahrs erst am 29. Februar eines Schaltjahrs aus. Abbildung 1–1 soll den Zusammenhang zwischen Fehlhandlung, Fehlerzustand und Fehlerwirkung veranschaulichen und darstellen, welche Gegenmaßnahmen bzw. Methoden zur Aufdeckung angewendet werden können.

Testbegriff

Ähnlich dem Fehlerbegriff ist auch der Begriff »Testen« mit verschiedenen Bedeutungen belegt. Mit Testen wird oft der gesamte Prozess bezeichnet, ein Programm auf systematische Weise zu prüfen, um Vertrauen in die korrekte Umsetzung der Anforderungen[4] zu gewinnen und um Fehlerwirkungen nachzuweisen. Es ist auch ein Oberbegriff für alle Tätigkeiten und (Test-)Stufen im Testprozess. Jede einzelne Ausführung eines Testobjekts unter spezifizierten Bedingungen zum Zwecke der Überprüfung der Einhaltung der erwarteten Ergebnisse wird ebenso als Testen bezeichnet.

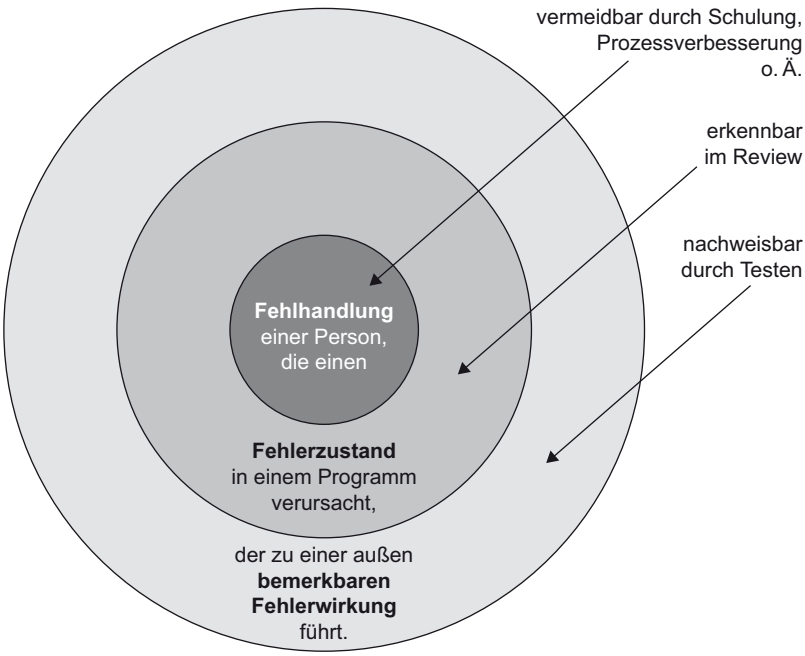

Abb. 1–1
Zusammenhang zwischen den Fehlerbegriffen

4. Mit Testen kann nicht nachgewiesen werden, dass die Anforderungen zu 100 % erfüllt sind, da Testen nur stichprobenartige Überprüfungen vornimmt.

Fundamentaler Testprozess

Testen umfasst eine Vielzahl von Einzelaktivitäten. Folgender fundamentaler Testprozess ist im Lehrplan »Foundation Level« definiert. Zum Prozess gehören folgende Aktivitäten:

- Testplanung und Steuerung,
- Testanalyse und Testentwurf,
- Testrealisierung und Testdurchführung,
- Bewertung von Endekriterien und Bericht,
- Abschluss der →Testaktivitäten.

Teststufen

Beim Testen kann das zu testende Produkt (Testobjekt) auf unterschiedlichen Abstraktionsebenen bzw. auf der Basis unterschiedlicher Dokumente und Entwicklungsprodukte betrachtet werden. Die entsprechende Bezeichnung ist →Teststufe. Es wird zwischen den Stufen Komponententest, Integrationstest, Systemtest und Abnahmetest unterschieden. Jede Teststufe zeichnet sich durch charakteristische Testziele, Testentwurfsverfahren und Testwerkzeuge aus.

Testarten

Daneben werden →Testarten unterschieden, die sich wie folgt abgrenzen lassen: funktionaler Test, nicht funktionaler Test, strukturbasierter Test und änderungsbezogener Test (s. [Spillner 12, Abschnitt 3.7]).

Statische und dynamische Prüfung

Beim Testen kann unterschieden werden, ob das Testobjekt zur Prüfung auf dem Rechner ausgeführt wird oder ob »nur« der zugehörige Programmtext, die zugrunde liegende Spezifikation oder Dokumentation geprüft wird. Im ersten Fall handelt es sich um sogenannte dynamische Prüfungen (mit den Vertretern Blackbox- und Whitebox-Testentwurfsverfahren, s. [Spillner 12, Kap. 5]), im zweiten Fall um statische Prüfungen (vertreten u.a. durch verschiedene Reviewarten und werkzeuggestützte statische Analysen, s. [Spillner 12, Kap. 4]).

Unabhängigkeit zwischen Test und Entwicklung

Unabhängig davon, welche Methoden zum Testen eingesetzt werden, sollen Entwicklung/Programmierung und Test organisatorisch möglichst getrennt bzw. unabhängig voneinander ablaufen. Denn ein Entwickler, der sein eigenes Programm testet, ist »blind« gegenüber eigenen Fehlhandlungen. Wer weist sich schon gerne seine eigenen Fehler nach?

Testwerkzeuge

Für das Testen von Software gibt es eine Vielzahl unterstützender Werkzeuge. Je nach Einsatzzweck werden verschiedene Werkzeugklassen unterschieden: u.a. Werkzeuge für Management und Steuerung von Tests, Werkzeuge zur Testspezifikation, zum statischen und dynamischen Test und für nicht funktionale Tests (s. [Spillner 12, Kap. 7]).

Testmanagement

Im »Foundation Level« werden auch schon die grundlegenden Aspekte des Testmanagements behandelt. Neben Testplanung, Teststeuerung und Berichtswesen gehören hierzu auch die Themen Fehler-, Änderungs- und Konfigurationsmanagement sowie das Thema Wirt-

schaftlichkeit des Testens (s. [Spillner 12, Kap. 6]). Das vorliegende Buch vertieft diese Aufgaben des Testmanagements.

Zur Veranschaulichung des Stoffs wird in diesem Buch das Fallbeispiel aus dem »Basiswissen«-Buch fortgesetzt:

Fallbeispiel »VirtualShow-Room« – VSR

Ein Automobilkonzern entwickelt ein neues elektronisches Verkaufssystem, genannt *VirtualShowRoom* (VSR). Das Softwaresystem soll in der Endausbaustufe weltweit bei allen Händlern installiert sein. Jeder Kunde, der ein Fahrzeug erwerben möchte, kann dann unterstützt durch einen Verkäufer oder vollkommen selbstständig sein Wunschfahrzeug am Bildschirm konfigurieren (Modellauswahl, Farbe, Ausstattung usw.).

Das System zeigt mögliche Modelle und Ausstattungsvarianten an und ermittelt zu jeder Auswahl des Kunden sofort den jeweiligen Listenpreis. Diese Funktionalität wird vom Teilsystem *DreamCar* realisiert.

Hat sich der Kunde für ein Fahrzeug entschieden, kann er am Bildschirm die für ihn optimale Finanzierung kalkulieren (*EasyFinance*), das Fahrzeug online bestellen (*JustInTime*) und bei Bedarf auch die passende Versicherung (*NoRisk*) abschließen. Das Teilsystem *ContractBase* verwaltet sämtliche Kundeninformationen und Vertragsdaten. Abbildung 1–2 zeigt eine schematische Darstellung des Systems.

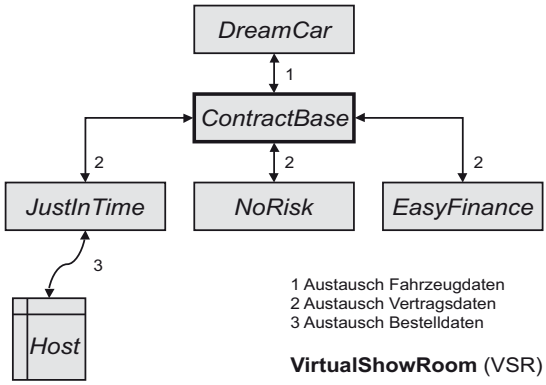

Abb. 1–2
Architektur des VSR-Systems

Jedes Teilsystem wird von einem eigenen Entwicklungsteam separat entworfen und entwickelt. Insgesamt sind ca. 50 Entwickler und weitere Mitarbeiter aus den jeweils betroffenen konzerninternen Fachabteilungen an dem Projekt beteiligt sowie externe Softwarefirmen.

Im »Basiswissen«-Buch wurden die verschiedenen →Testentwurfsverfahren und Vorgehensweisen beschrieben, um das System gründlich zu testen, bevor das VSR-System in Betrieb geht.

Die Entwicklung des *VSR-2* folgt einem iterativen Entwicklungsprozess. Aus dem vorhandenen *VSR-1* soll mit vier aufeinanderfolgenden Iterationen der *VSR-2* entstehen. Dafür ist eine Entwicklungsdauer von einem Jahr vorgesehen. Es wird also etwa quartalsweise eine Zwischenversion geben.

Jede neue Version soll die Funktionalität der Vorgängerversion weiterhin korrekt bereitstellen. Allerdings kann der eine andere, vielleicht bessere oder effizientere Implementierung zugrunde liegen. Zusätzlich implementiert jede Version erstmalig einen Satz neuer Funktionen.

Der Produktmanager erwartet vom Testmanager daher zweierlei:

- Zum einen muss das Testteam sicherstellen, dass jede *VSR-2*-Version die bisherige Altfunktionalität korrekt enthält.
- Zum anderen soll das Testteam möglichst schnell eine objektive Beurteilung abgeben, ob bzw. wie gut ein neues *Feature* umgesetzt ist.

Die Aufgaben, die bei einer solchen Problemstellung vom Testmanager zu erfüllen sind, werden in den folgenden Kapiteln behandelt und anhand obigen Beispiels jeweils verdeutlicht.

1.2 Praxiswissen Testmanagement – Übersicht

Praxiswissen – Kapitelübersicht

Die Themen des Buches und die Inhalte der einzelnen Kapitel sind im Folgenden kurz beschrieben. Die Marginalien zitieren die im Übersichtsdokument zu den Lehrplänen des Certified Tester – Advanced Level angegebenen Punkte zum geschäftlichen Nutzen von Testmanagern:

Ein erfolgreicher Testmanager kann [CTAL 12] …

… ein Testprojekt leiten und die für die Testorganisation festgelegten Aufgaben, Ziele und Testprozesse umsetzen;

- In Kapitel 2 wird der grundlegende Testprozess erörtert. Die wesentlichen Aktivitäten des Testmanagements im Testprozess werden ausführlich beschrieben. Das Kapitel geht insbesondere näher auf die Testplanung ein, eine wichtige, wenn nicht sogar die wichtigste Aufgabe des Testmanagers. Die Planung muss während des Projekts angepasst werden.
- Wie das Testen in Verbindung zum Softwarelebenszyklus steht, wird in Kapitel 3 dargestellt. Unterschiedliche Vorgehensmodelle der Softwareentwicklung werden diskutiert und die jeweilige Bedeutung des Testens im Modell bewertet.

… Risikoidentifizierung und -analysesitzungen organisieren und leiten, und deren Ergebnisse für die Planung, Aufwandsschätzung, Überwachung und Steuerung der Testaktivitäten verwenden;

- Identifikation und Analyse der Risiken sowie risikoorientierte Tests sind für das Testmanagement wichtige Instrumente zur Verteilung der beschränkten Testkapazitäten und dienen zur risikomindernden Steuerung des →Testprojekts. In Kapitel 4 sind ent-

sprechende Hinweise zum Vorgehen sowie zu anderen Testansätzen zu finden.

- Kapitel 5 erläutert das grundlegende Vorgehen sowie einige Techniken zur Aufwandsschätzung, die die zielgenaue Zeit- und Ressourcenplanung unterstützen.

… Testkonzepte erstellen und umsetzen, die der Richtlinie und der Teststrategie der Organisation entsprechen;

- Dokumente sind ein zentraler Bestandteil des Testprozesses. Planung und Status der Tests werden in zentralen Dokumenten festgehalten und aktualisiert. Kapitel 6 stellt einen Überblick über Arten und Zusammenhänge der wichtigsten Testdokumente dar und erläutert die für das Testmanagement relevanten Dokumente im Detail.

… die Testaktivitäten zur Erreichung der Projektziele kontinuierlich überwachen und steuern;

- Testmetriken erlauben quantitative Aussagen bezüglich der Produktqualität, des aktuellen Projektstands und der Reife des Entwicklungs- und Testprozesses und helfen, Kriterien für die Beendigung des Testens festzulegen. Maßtheoretische Grundlagen und konkrete Beispiele hierfür werden in Kapitel 7 gegeben.

… den relevanten Teststatus bewerten und den Projektbeteiligten zeitgerecht darüber berichten;

- Die Steuerung des Testprozesses auf Grundlage der Messwerte in den Berichten über den Testfortschritt ist für den Testmanager eine entscheidende Maßnahme, um den Testprozess erfolgreich durchführen zu können. Kapitel 8 geht auf diesen Aspekt ein.

… wirtschaftliche Argumente für Testaktivitäten vorbringen und darlegen, welche Kosten und Nutzen zu erwarten sind;

- Da Testen bei vielen Stakeholdern nur mit Kosten assoziiert wird, zeigt Kapitel 9 den Mehrwert des Testens auf, der aus dem investierten Testaufwand gezogen werden kann.

… die angemessene Kommunikation zwischen den Mitgliedern des Testteams untereinander sowie zwischen Testteam und anderen Projektbeteiligten sicherstellen;

- Die Einbindung des Testteams in die Aufbauorganisation des Unternehmens – von einzelnen Testern im Projekt bis hin zum verteilten Testen – sowie die damit verknüpften Koordinations- und Kommunikationsaufgaben sind Gegenstand von Kapitel 10.
- In Kapitel 11 werden für das Testmanagement relevante Normen und Standards vorgestellt und diskutiert.

- Reviews zur Qualitätssicherung von Dokumenten werden in vielen Unternehmen mit sehr gutem Erfolg angewendet. Die unterschiedlichen Vorgehensweisen werden ausführlich in Kapitel 12 beschrieben.
- Wie ist mit den beim Testen gefundenen Abweichungen und Fehlerwirkungen umzugehen? Antworten hierzu gibt Kapitel 13.

… sich an Initiativen zur Testprozessverbesserung beteiligen und diese leiten;

- Auch der Entwicklungs- und Testprozess selbst kann und soll regelmäßig bewertet und verbessert werden. Welche Verfahren und Vorgehensweisen dazu anzuwenden sind, wird in Kapitel 14 beschrieben.
- Mit entsprechender Werkzeugunterstützung lässt sich der Testprozess meist effizienter durchführen. Welche Werkzeugtypen generell sinnvoll im Testprozess eingesetzt werden können und wie der Testmanager passende Werkzeuge auswählt und einführt, wird in Kapitel 15 beschrieben.

… Qualifikationen und unzureichende Ressourcen im Testteam identifizieren und bei der Beschaffung angemessener Ressourcen mitwirken und die für das Testteam benötigte Entwicklung von Qualifikationen identifizieren und planen;

- Ohne Mitarbeiter mit den erforderlichen Fähigkeiten und Qualifikationen – ohne Berücksichtigung des »Faktors Mensch« – kann der Testmanager die Testaufgaben nicht erfolgreich durchführen. In Kapitel 16 wird beschrieben, was bei der Zusammenstellung des Testteams zu berücksichtigen ist.

Selbstverständlich muss das Buch nicht in dieser linearen Reihenfolge gelesen werden, sondern kann auch punktuell als Nachschlagewerk oder anhand der Querverweise als Hypertext »explorative« Verwendung finden.

Eine weitere Reihenfolge, in der die Kapitel gelesen werden können, ergibt sich aus der folgenden, oft in der Praxis zu beobachtenden Handlungskette:

- Worum geht es im Kern? Um den fundamentalen Testprozess (Kap. 2).
- Wie sehen die Randbedingungen dazu aus? Der Kontext des Testmanagements (Kap. 3).
- Was bringt das Testen denn? (Eine sehr oft gestellte Frage, mit der Testmanager oft konfrontiert werden!) Der Mehrwert des Testens (Kap. 9).
- Wie organisiere ich als Testmanager das Testen? Testorganisation (Kap. 10).

- ... und welche Leute brauche ich dazu? Kompetenzen und Teamzusammensetzung (Kap. 16).
- Natürlich mache ich mir Gedanken zur Dokumentation, bevor es mit dem Test losgeht! Testdokumentation (Kap. 6).
- ... und selbstverständlich müssen jegliche Dokumentation und alle anderen Ergebnisse geprüft werden. Reviews, Audits, Assessments (Kap. 12).
- ... und wie aufwendig das Testen ist, muss vorab klar sein. Testaufwandsschätzung (Kap. 5).
- Na, dann kann es ja losgehen mit dem Test. Risikoorientierte und andere Testverfahren (Kap. 4).
- ... und natürlich finden wir Fehler! Fehlermanagement (Kap. 13).
- Wie können Testmanager managen? Testmetriken definieren (Kap. 7) und Testmetriken anwenden (Kap. 8).
- Worauf müssen Testmanager noch achten? Normen und Standards (Kap. 11).
- Was kann langfristig verbessert werden? Bewertung und Verbesserung des Testprozesses (Kap. 14).
- Und wie sieht es mit Werkzeugen aus? Werkzeuge zur Unterstützung des Testprozesses (Kap. 15).

Das Glossar enthält alle hier im Buch verwendeten Begriffe. Weitere Glossareinträge aus dem Buch »Basiswissen Softwaretest« (s. [Spillner 12]) finden Sie auch unter [URL: Glossar GTB] oder als mehrsprachiges Glossar unter [URL: Glossar imbus].

2 Fundamentaler Testprozess

In diesem Kapitel wird der fundamentale Testprozess mit seinen einzelnen Aktivitäten aus Sicht des Testmanagements vorgestellt[1].

In den meisten Entwicklungsmodellen (s. Kap. 3) wird das Testen nur sehr allgemein dargestellt. Um Tests strukturiert durchzuführen, reicht eine solche grobe Darstellung nicht aus. Neben der Einordnung des Testens in den Entwicklungsprozess ist ein detailliertes Vorgehensmodell für die Testarbeiten erforderlich. Dem Testmanagement obliegt dabei die Verwaltung des Testprozesses, der Testinfrastruktur und der Testmittel (engl. *testware*).

Prozessphasen

Die Entwicklungsaufgabe »Testen« wird dazu in folgende Arbeitsabschnitte bzw. Prozessphasen unterteilt: Testplanung, -überwachung und -steuerung, Testanalyse und Testentwurf, Testrealisierung und Testdurchführung, Bewertung von Endekriterien und Bericht sowie Abschluss der Testaktivitäten (s. Abb. 2–1).

Obgleich die Darstellung und die Beschreibung der einzelnen Aufgaben eine rein sequenzielle Bearbeitung im Testprozess suggeriert, können die einzelnen Aktivitäten sich je nach Entwicklungsmodell überschneiden und teilweise auch parallel und wiederholt durchgeführt werden. So ist beispielsweise eine Überlappung der Analyse und des Entwurfs von Testfällen und deren Durchführung möglich, um anhand der Erfahrungen mit den durchgeführten Testfällen weitere ergänzende Testfälle zu spezifizieren.

1. Die Beschreibung konzentriert sich auf die Aspekte des Testprozesses, die im »Advanced Level« beschrieben sind. Eine Einführung in den grundlegenden Testprozess ist in [Spillner 12, Abschnitt 2.2] zu finden.

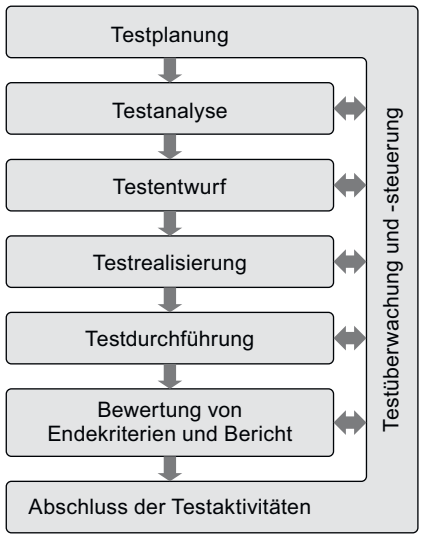

Abb. 2–1
Fundamentaler Testprozess

Testprozess für das jeweilige Projekt umsetzen

Für Testmanager ist das Wissen über den Testprozess von zentraler Bedeutung, denn ihre Aufgabe ist es, notwendige Anpassungen und Ausgestaltungen des Testprozesses projektspezifisch vorzunehmen und zu optimieren. Das hier beschriebene abstrakte Testprozessmodell soll dabei eine Hilfestellung geben.

Weitere Testprozessmodelle

Weitere Testprozessmodelle und Testprozessverbesserungsmodelle, wie beispielsweise TMMi (*Test Maturity Model Integration*), CTP (*Critical Testing Processes*) oder STEP (*Systematic Test and Evaluation Process*) und TPI Next *(Test Process Improvement Next)*, sind in Kapitel 14 beschrieben und geben Testmanagern zusätzliche nützliche Anregungen.

2.1 Testplanung

Festlegungen im Testkonzept dokumentieren

Die Planung einer so umfangreichen Aufgabe wie des Testens soll so früh wie möglich beginnen, am besten gleich zu Anfang des Softwareentwicklungsprojekts. Aufgaben und Zielsetzung der Tests müssen ebenso festgelegt werden wie die benötigten Ressourcen. Dazu gehören die erforderlichen Mitarbeiter zur Durchführung der Aufgaben, die zu veranschlagende Zeit sowie die notwendigen Hilfsmittel und Werkzeuge. Die entsprechenden Festlegungen sind im →Testkonzept (engl. *test plan*) zu dokumentieren (s. Abb. 2–2). Eine Organisationsstruktur mit dem entsprechenden Testmanagement soll vorhanden sein und ist ggf. anzupassen.

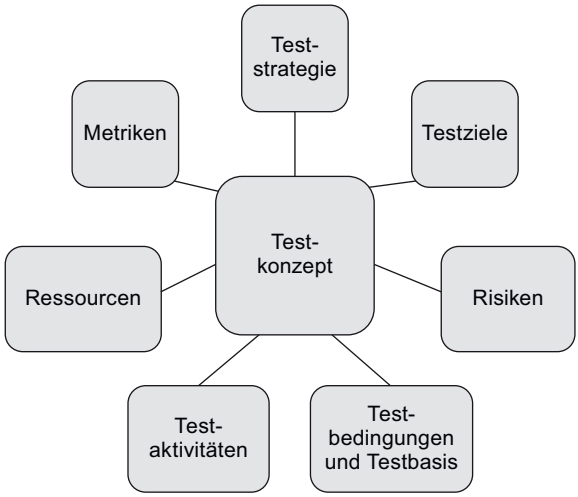

Abb. 2–2
Aspekte des Testkonzepts

2.1.1 Definition der Teststrategie

Festlegung der Teststrategie

Die Teststrategie (auch Testvorgehensweise genannt) bildet den roten Faden von den Anforderungen an das zu testende System hin zu den einzelnen Testaktivitäten. Sie definiert die zur Prüfung der Testobjekte einzusetzenden Testverfahren, die Verteilung der zur Verfügung stehenden Ressourcen auf die zu testenden Systemteile und die jeweiligen Qualitätsmerkmale. Ebenso wird die Reihenfolge der durchzuführenden Aktivitäten in der Teststrategie festgelegt.

Von systematischen bis hin zu reaktiven Strategien – der richtige Mix macht's!

In der Praxis werden je nach Organisations- und Projektkontext unterschiedliche Teststrategien verfolgt. Der Lehrplan benennt hier u.a. analytische, auf einer Testbasis fußende Strategien wie z.B. das risikoorientierte Testen (s.u.) ebenso wie modellbasierte Strategien, die auf Modellen der Systemnutzung oder des Systems aufsetzen. Reaktive Strategien dagegen konzentrieren sich auf manuelle Tests und kommen erst zum Einsatz, wenn ein ausführbares System vorliegt. Nähere Ausführungen zu den unterschiedlichen Teststrategien finden sich in Kapitel 4.

Allgemeine und projektspezifische Teststrategie

Die Teststrategie kann auf zwei Ebenen formuliert werden: Sie kann allgemein und damit unabhängig vom konkreten Testprojekt als Teil eines Testhandbuches (s. Abschnitt 6.2.2) festgelegt werden oder aber auf projektspezifischer Ebene formuliert werden. In reifen Testorganisationen sind beide Ebenen abgedeckt, wobei möglichst viele Bestandteile der Teststrategie übergreifend im Testhandbuch formuliert werden.

Bei der Bestimmung der Teststrategie sind gemäß dem Risiko unterschiedliche Testentwurfsverfahren und Endekriterien festzulegen. Kritische Systemteile müssen intensiv getestet werden. Bei den weniger kritischen Teilen reicht ein nicht so umfangreicher Test oder sogar ein Verzicht auf das Testen aus. Die Festlegungen müssen sehr fundiert getroffen werden, um eine bestmögliche und nachvollziehbare Verteilung der Testaktivitäten auf die »richtigen« Stellen bzw. Teile des Softwaresystems zu erreichen.

Risikoorientiertes Testen

Risikoorientiertes Testen (s. Abschnitt 4.2) dient dazu, die identifizierten Risiken zu reduzieren. Wenn beispielsweise bei mehreren Projekten festgestellt wird, dass schwerwiegende Abweichungen an einer ungenauen oder fehlerhaften Entwurfsspezifikation liegen, dann ist der Testprozess zu ergänzen und bei der Testprozessplanung zusätzlich →statische Tests (Reviews) der Entwurfsspezifikation einzufordern, um die Qualität der Dokumente zu erhöhen.

Endekriterien festlegen

Die Intensität des Tests wird bestimmt durch die eingesetzten Testentwurfsverfahren und den jeweiligen angestrebten Überdeckungsgrad bei der Durchführung der Testfälle. Der Überdeckungsgrad ist eines von verschiedenen Kriterien, anhand derer das mögliche Ende des Tests bestimmt wird. Hierfür sind entsprechende Metriken zu definieren (Kap. 7).

Priorisierung der Tests

Softwareprojekte geraten oft unter Zeitdruck, dies muss bei der Planung vorausschauend berücksichtigt werden. Eine Priorisierung der durchzuführenden Tests bewirkt, dass die kritischsten Softwareteile zuerst getestet werden, falls wegen Zeit- oder Ressourcenbeschränkungen nicht alle geplanten Tests durchgeführt werden können.

Die Priorisierung der Tests hängt dabei eng mit der gewählten Teststrategie zusammen. Bei einer risikoorientierten Teststrategie erfolgt sie gemäß der identifizierten Risiken, bei anderen Ansätzen z. B. gemäß der Priorität der Anforderungen oder unter Berücksichtigung eines Nutzungsprofils des Systems (Kap. 4).

Zeitpunkt des Beginns der Testaktivitäten

Zunächst wird beim Entwurf der Strategie betrachtet, zu welchem Zeitpunkt im Verlauf des Gesamtentwicklungsvorhabens das Testen beginnt und welche Aktivität in welcher Phase des Projekts durchzuführen ist. Danach wird zwischen präventiven und reaktiven Strategiebestandteilen unterschieden. Präventive Strategien zielen darauf ab, durch frühzeitigen Beginn der Testplanung und des Testentwurfs die Entstehung von Fehlern zu minimieren bzw. Fehler möglichst frühzeitig zu finden. Bei reaktiven Strategien beginnt die Testplanung und insbesondere der Testfallentwurf erst bei Vorliegen des lauffähigen Systems.

Beispiele für präventive Strategiebestandteile

Typische Bestandteile einer präventiven Teststrategie sind:

- eine risikobasierte Testpriorisierung,
- die Auswahl von Testentwurfsverfahren anhand der zu testenden Qualitätsmerkmale,
- modellbasierte Strategien, die auf Anforderungs- oder Entwurfsmodellen des zu testenden Systems basieren,
- eine prozess- oder standardorientierte Vorgehensweise.

Gemeinsames Merkmal der präventiven Strategien ist, dass sie auf Informationen basieren, die vor Verfügbarkeit des implementierten Systems vorliegen.

Beispiele für reaktive Strategiebestandteile

Als reaktive Strategiebestandteile können dagegen gesehen werden:

- dynamisches und heuristisches Vorgehen wie z. B. fehlerbasiertes Testen,
- exploratives Testen,
- Regressionsteststrategien, z. B. Testautomatisierung.

Im Allgemeinen ist keine Teststrategie rein reaktiv oder rein präventiv; der Testmanager soll einen Mix aus Strategiebestandteilen wählen, der zum Projekt, der Organisation, der vorhandenen Technologie und den einsetzbaren Ressourcen passt.

Abgrenzung der Aufgaben des Testprojekts

Im Normalfall ist der Testmanager mit seinem Team nicht die einzige Instanz, die sich mit Qualitätssicherung und Test im Entwicklungsprojekt beschäftigt. Eine klare Abgrenzung der Aufgaben der einzelnen Instanzen sorgt dafür, dass weder wichtige Themen liegen bleiben noch dass zu viele Redundanzen auftreten. Bei der Teilaufgabe Testplanung ergibt sich oftmals erhöhter Kommunikations- und Koordinationsbedarf zwischen Testmanagern verschiedener Teststufen oder Teilprojekte, dem Projektleiter und dem Entwicklungsleiter.

Zu testende und nicht zu testende Komponenten des Systems

Die Testobjekte sowie die Komponenten, aus denen die Testobjekte bestehen, müssen klar definiert werden. Dies umfasst auch Angaben zu den Versionen der Testobjekte. Ebenso ist die Art der Übergabe der Testobjekte von der Entwicklung an den Test festzulegen – erfolgt diese z. B. per Installation in einer Cloud-Umgebung, per Download, per CD/DVD oder anderem Medium?

Wenn einzelne Bestandteile des Systems explizit keine Testobjekte sind, jedoch zum Testbetrieb der Testobjekte erforderlich sind, soll dies ebenfalls erwähnt werden. Einige Systembestandteile werden beispielsweise ggf. von Zulieferern erstellt und in getestetem Zustand abgegeben, sodass sie innerhalb des Projekts nicht nochmals getestet werden müssen.

Test- und Qualitätsziele

Identifikation der Testziele

Projektziele bei der Entwicklung eines Systems umfassen Ziele, die auf das Testprojekt Auswirkungen haben. Hierzu gehören der geplante Freigabetermin, das Gesamt- und das dem Test zugewiesene Teilbudget etc. sowie das angestrebte Qualitätsniveau, also die Art und Umsetzung der einzelnen Qualitätsmerkmale des Systems.

Das Testkonzept stellt in Bezug auf die Qualitätsmerkmale eine Art Kontrakt zwischen Projektleitung und Testmanagement dar. Es legt fest, welche der Qualitätsmerkmale in welcher Weise im Test nachgewiesen werden, welches also die Testziele des Projekts sind.

Um eine nachvollziehbare und über alle Teststufen einheitliche Zuordnung von Qualitätsmerkmalen zu Testzielen und den zu deren Prüfung geeigneten Testverfahren zu erreichen, wird am besten auf ein wohldefiniertes Qualitätsmodell wie die ISO/IEC-Norm 25010 zurückgegriffen (s. [ISO 25010]). Diese Norm aktualisiert die bewährte ISO-Norm 9126 und gliedert die Qualitätsmerkmale von Softwaresystemen ebenfalls in drei Untergruppen (s. Abb. 2–3): Merkmale der anwendungsbezogenen Qualität (Gebrauchsqualität, engl. *quality in use*) sowie in externe und interne Qualitätsmerkmale (engl. *external and internal quality*).

Abb. 2–3
Gruppen der Qualitätsmerkmale von Software nach ISO 25010

Für jeden dieser drei Bereiche von Qualitätsmerkmalen werden entsprechende Untermerkmale festgelegt. Diese können anhand bestimmter Messverfahren (Metriken, s. Kap. 7) definiert und somit quantifizierbar als Testziele vorgegeben und am Testobjekt gemessen werden. Abbildung 2–4 zeigt die Merkmale der anwendungsbezogenen Qualität, Abbildung 2–5 die externen und internen Qualitätsmerkmale.

Anwendungsbezogene Qualität				
Effektivität	**Effizienz**	**Zufriedenheit**	**Risiko-abwesenheit**	**Kontext-abdeckung**
Aufgaben-erfüllung innerhalb der Genauigkeits- und Voll-ständigkeits-grenzen	Aufgabenerfül-lung innerhalb der Aufwands-grenzen für Benutzer (Zeit, Kosten, ...)	Verwendbar-keit Vertrauen Begeisterung Bequem-lichkeit	Wirtschaftliche Risiken Gesundheits- und Sicher-heitsrisiken Umgebungs- und Umwelt-risiken	Kontextuelle Vollständigkeit Flexibilität

Abb. 2–4
Anwendungsbezogene Qualitätsmerkmale nach ISO 25010

Externe Qualität					
Funktionalität	**Performanz**	**Kompatibilität**	**Benutzbarkeit**	**Zuverlässig-keit**	**Sicherheit**
Vollständigkeit Richtigkeit Angemessen-heit	Zeitverhalten Ressourcen-gebrauch Kapazität	Koexistenz Interopera-bilität	Angemessen-heit Erkennbarkeit Erlernbarkeit Bedienbarkeit Fehlertoleranz Ästhetik Zugänglichkeit	Reife Verfügbarkeit Fehlertoleranz Wiederherstell-barkeit	Vertraulichkeit Integrität Nachweis-barkeit Verantwortlich-keit Authentifizier-barkeit

Interne Qualität	
Wartbarkeit	**Portierbarkeit**
Modularität Wiederver-wendbarkeit Analysier-barkeit Modifizier-barkeit Testbarkeit	Anpassbarkeit Installierbarkeit Austausch-barkeit

Abb. 2–5
Interne und externe Qualitätsmerkmale nach ISO 25010

Durch dieses Schema sind die (funktionalen und nicht funktionalen) Eigenschaften der Software gut beschreibbar; insbesondere die Angabe geeigneter Prüfverfahren und die Quantifizierung von Akzeptanzkriterien werden durch diese Norm unterstützt (s.a. Abschnitt 7.7). Vor allem aber wird das Augenmerk auf die in der Praxis oft vernachlässigten nicht funktionalen Qualitätsmerkmale wie z.B. Zuverlässigkeit und

Nicht funktionale Qualitätsmerkmale werden oft vernachlässigt.

Benutzbarkeit gelenkt. Da die nicht funktionalen Anforderungen vielfach schlecht oder gar nicht explizit als Anforderungen spezifiziert sind, fällt dem Testmanagement folgende Aufgabe zu: Die impliziten Erwartungshaltungen möglichst aller verschiedenen →*Stakeholder* des Systems (z. B. der Endanwender, des Betreibers, der Wartungsmitarbeiter) während der Testplanung weitgehend zu konkretisieren und in eine adäquate Planung nicht funktionaler Tests umzusetzen. Dabei ist zu berücksichtigen, dass erst durch das Vorliegen erster Ergebnisse dieser Tests die vagen Anforderungen konkretisiert werden können und anschließend ggf. weitere nicht funktionale Tests erforderlich werden.

Planung frühzeitiger Reviews von Anforderungsdokumenten

Das »nachträgliche« *Requirements Engineering*, das Testmanager oft betreiben müssen, um aussagekräftige Testziele zu erhalten, lässt sich vermeiden oder deutlich reduzieren, wenn frühzeitig auf Umfang und Qualität der nicht funktionalen Anforderungen geachtet wird. Testmanager können dies durch die Planung und Durchführung von Reviews der Anforderungen erreichen. Dabei ist eine Review-Checkliste hilfreich, die folgende Punkte enthält, auf die die Reviewer besonders achten sollen:

- Die Anforderungen müssen in möglichst einfacher und klarer Sprache gehalten sein; bestimmte Schlüsselwörter wie »soll« (kennzeichnet eine notwendige Eigenschaft des Systems) und »sollte« (kennzeichnet eine wünschenswerte Eigenschaft des Systems) müssen einheitlich verwendet werden. Alle Anforderungen sollen in einem einheitlichen Format dokumentiert werden.
- Die Leser von Anforderungsdokumenten haben einen unterschiedlichen fachlichen und technischen Hintergrund. Beim Review ist möglichst ein Vertreter jeder Stakeholder-Gruppe einzubinden.
- Quantitative Angaben für das geforderte Verhalten des Systems sind qualitativen Angaben vorzuziehen. Am besten wird bei der Formulierung der Anforderung bereits eine Metrik und die verschiedenen Akzeptanzniveaus angegeben.

Beispiel: Testziele des VSR-Testkonzepts

Das Testkonzept für VirtualShowRoom (VSR) enthält beispielsweise für das Teilsystem *DreamCar* die folgenden Aussagen:

- Als Testziel zur »Funktionalität« wird festgehalten, dass alle Anwendungsfälle des Systems mit möglichst allen zur Verfügung stehenden Fahrzeug-, Sondermodell- und Zubehördaten kombinatorisch durch Tests abzudecken sind. Da hierbei die Exaktheit der berechneten Fahrzeugpreise ein zentrales Leistungsmerkmal ist, werden insbesondere frühzeitige Tests der Richtigkeit der Berechnungen im Komponententest vorgesehen.
- Als typisches Mehrbenutzersystem wird *DreamCar* während des Betriebs wechselnden Lastanforderungen gerecht werden müssen. Um diese adäquat testen zu können, werden im Testkonzept Analyse-, Realisierungs-

und wiederholte Durchführungsphasen für Lasttests vorgesehen. Dabei sollen die Nutzungsprofile der zukünftigen Anwender im Hinblick auf Nutzerzahl, Nutzungshäufigkeit der einzelnen Features und deren zeitliche Verteilung auf einen typischen Arbeitstag in Form eines Lastprofils modelliert und anschließend mit einem Performanztestwerkzeug automatisiert durchgeführt. Durch diese Tests werden Qualitätsziele im Bereich »Effizienz«, aber auch im Bereich »Zuverlässigkeit« abgedeckt. Die Ergebnisse der Lasttests werden mit dem Auftraggeber und Vertretern der künftigen Endanwender diskutiert, um festzustellen, ob das Verhalten des Systems unter Last für den Praxiseinsatz ausreichend ist.

- Da *DreamCar* von Autohändlern und Kaufinteressenten gleichermaßen ohne vorherige Einarbeitung bedient werden soll, wird dem Qualitätsmerkmal »Benutzbarkeit« ebenfalls ein hoher Stellenwert eingeräumt, der allerdings erst im Rahmen des Systemtests adressiert werden soll.[2] Dort soll eine Benutzbarkeitsprüfung unter Beteiligung von mindestens fünf Repräsentanten von Fahrzeughändlern sowie fünf zufällig gewählten »Leuten von der Straße« durchgeführt werden.

Qualitätsziele, die nicht getestet werden

Identifikation der Nicht-Ziele des Testprojekts

Es ist eine wichtige Aufgabe von Testmanagern, klarzustellen, welche Projektziele explizit im Testkonzept berücksichtigt werden, d.h. aber auch, welche Qualitätsmerkmale nicht einem Test unterzogen werden können oder sollen. Dies unterstützt die klare Aufteilung der Verantwortlichkeiten zwischen dem Test und anderen Formen der Qualitätssicherung, z.B. die konstruktive Berücksichtigung bestimmter Qualitätsanforderungen durch entsprechende Entwurfsmethoden.

Beispiel: Nicht-Testziele bei VSR DreamCar

Das Testkonzept für VSR sieht zum Beispiel für das Teilsystem *DreamCar* Folgendes vor:

- Änderbarkeit und Übertragbarkeit müssen durch Reviews oder andere statische Qualitätssicherungsmaßnahmen geprüft werden.
- Der Ressourcenverbrauch der Implementierung von *DreamCar* spielt keine wesentliche Rolle und wird nicht explizit getestet.
- Die Datenbank mit den Fahrzeugdaten wird von einem Unterauftragnehmer zugeliefert, der diese als Standardprodukt vermarktet; daher werden lediglich Regressionstests der Funktionalität dieser Datenbank als Eingangsprüfung bei Lieferung neuer Versionen vorgesehen.

2. Da vorher bereits ein Benutzungsprototyp erstellt wurde, kennen die späteren Anwender schon die Benutzungsoberfläche. Ansonsten wäre ein erster Test der Benutzbarkeit auf der Systemteststufe sehr riskant, weil es durch den Test zu erheblichen Änderungen des Systems kommen kann.

2.1.2 Art und Umfang der Tests

Nach der Frage der Abgrenzung, also dem »Was« des Testprojekts, kann das »Wie« festgelegt werden. Dieser Teil des Testkonzepts fordert von Testmanagern Fertigkeiten, die über klassisches Projektmanagement hinausgehen. Sie müssen Qualitätsmerkmale und -anforderungen verstehen und bewerten sowie mit verschiedensten Testverfahren vertraut sein. Bei der Ausarbeitung dieses Teils der Teststrategie ist eine enge Zusammenarbeit mit den Testanalysten und technischen Testanalysten zu empfehlen. Gemeinsam werden dann in der Testanalyse aus den Qualitäts- und Testzielen →Testbedingungen abgeleitet, die in der Phase des Testentwurfs die Leitlinie für die Entstehung der einzelnen Testfälle bilden.

Strategie zur Auswahl geeigneter Testverfahren

Die Prüfung der für das System geforderten Qualitätsmerkmale erfordert je nach Merkmal völlig unterschiedliche Vorgehensweisen und Testverfahren, die auf den verschiedenen Teststufen angewendet werden müssen. Testen ist im Kontext der ISO/IEC 25010 ein Vorgang, mit dem der Grad der Erfüllung der Qualitätsmerkmale durch Anwendung der definierten Messverfahren ermittelt werden soll. Die Norm liefert eine allgemeine Definition für diesen Bewertungsprozess (s. Abb. 2–6).

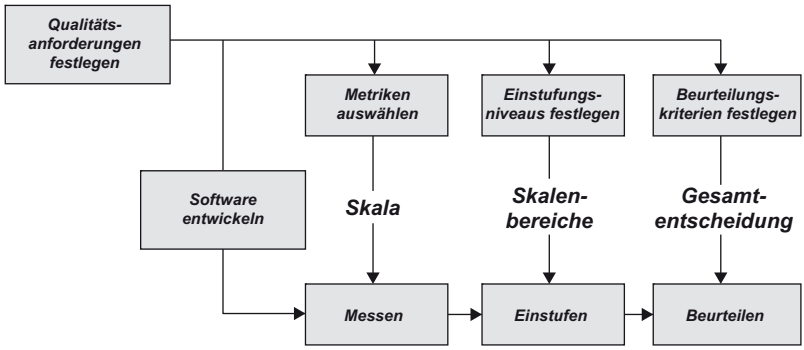

Abb. 2–6 *Bewertungsprozess für Software*

Testen kann in diesem Prozess aufgefasst werden als Abfolge der Tätigkeiten:

- Metriken auswählen (d.h. die dazu geeigneten Testverfahren festlegen),
- Einstufungsniveaus festlegen (d.h. Akzeptanzkriterien definieren),
- Messen (d.h. Tests durchführen),
- Einstufen (d.h. Vergleich des Istverhaltens mit den Akzeptanzkriterien).

Die Teststrategie ordnet jedem zu testenden Leistungsmerkmal eine Spezifikation der anzuwendenden Verfahren zu, mit denen dieses Leistungsmerkmal angemessen geprüft wird, sowie das zur Akzeptanz des Merkmals nachzuweisende Einstufungsniveau.

Viele Testverfahren sind grundsätzlich skalierbar, d.h., durch Investition eines höheren Aufwands kann eine höhere Testabdeckung erreicht werden. Im Normalfall muss diese Skalierbarkeit bei der Verteilung der zur Verfügung stehenden Ressourcen auf die Testaktivitäten genutzt werden, da nicht alle möglichen Testaufgaben gleich intensiv bearbeitet werden können.

In einer reifen Testorganisation kann diese Aufgabe auf der Ebene des Testhandbuches (s. Abschnitt 6.2.2) erledigt werden – dieses enthält dann Informationen über die allgemeine Testvorgehensweise und ordnet hierfür die allgemeinen Qualitätsmerkmale und die im Unternehmen verfügbaren Testverfahren einander zu.

Zur Vermeidung von systematischen Schwächen einer methodisch aufgebauten Teststrategie sowie bei fehlenden Anforderungsdokumenten kann exploratives Testen (s. Abschnitt 3.5.2) als ergänzender Bestandteil der Teststrategie sinnvoll sein.

Tipp

Wiederverwendbarkeit von Teststrategien

Die Ausarbeitung einer passenden Teststrategie ist ein aufwendiger und komplexer Vorgang, der – wie erwähnt – den iterativen Abgleich von Testzielen, Aufwand und Organisationsstrukturen erfordert. Umso wichtiger ist es, dass eine so gewonnene Strategie gut dokumentiert und – von projektspezifischen Anteilen befreit – zur Wiederverwendung in das Testhandbuch der Organisation(seinheit) aufgenommen wird. Kandidaten für eine solche Wiederverwendungsstrategie sind zumindest:

- die Definition von Testentwurfsverfahren und deren Zuordnung zu Qualitätsmerkmalen sowie
- das Verfahren zur Ableitung der Testtiefe und Testreihenfolge auf Basis von Risikofaktoren.

Beispiele für Testentwurfsverfahren

Folgende zwei Beispiele illustrieren diese Zuordnung von Merkmalen, Verfahren und Einstufungsniveaus:

1. **»Richtigkeit«**
   Das funktionale Qualitätsmerkmal »Richtigkeit« (*functional correctness*) ist definiert als »der Grad, zu dem das System korrekte Ergebnisse mit der benötigten Präzision liefert« [ISO 25010].

   Geeignetes Testentwurfsverfahren ist auf der Ebene des Systemtests der anwendungsfallbasierte Test (s. [Spillner 12, Abschnitt 5.1.5]) unter ergänzender Anwendung von Äquivalenzklassenanalyse und Grenzwertanalyse (s. [Spillner 12, Abschnitt 5.1.1 und 5.1.2]).

Geeignete Metrik zur Bestimmung des Einstufungsniveaus ist beispielsweise die Abweichung zwischen berechnetem und erwartetem Wert eines Ausgabeparameters.

Denkbare Einstufungsniveaus für die Akzeptanz:

- 0 .. 0.0001: akzeptabel
- > 0.0001: nicht akzeptabel

2. **»Benutzbarkeit«**
Das nicht funktionale Qualitätsmerkmal »Benutzbarkeit« (*usability*) ist definiert als »das Ausmaß, in dem ein Produkt von spezifizierten Benutzern in einem spezifizierten Gebrauchskontext verwendet werden kann, um spezifizierte Ziele effektiv, effizient und zufriedenstellend zu erreichen« [ISO 25010].

   Geeignetes Testentwurfsverfahren hierfür ist ein Usability-Test, bei dem die Hauptanwendungsfälle des Systems durchgespielt werden.

   Geeignete Metrik ist die Messung des mittleren Zeitaufwands für jeden dieser Hauptanwendungsfälle für unterschiedliche Klassen von Benutzern (z.B. Neuanwender, Profis).

   Die Einstufungsniveaus sind dann sinnvollerweise spezifisch für jeden Anwendungsfall festzulegen; beispielsweise könnte für den Anwendungsfall »neuen Fahrzeugtyp in der *DreamCar*-Konfiguration anlegen« angegeben werden:

   - < 1 Minute: hervorragend geeignet
   - 1 – 2 Minuten: akzeptabel
   - 2 – 3 Minuten: bedingt akzeptabel
   - > 3 Minuten: nicht akzeptabel

2.1.3 Priorisierung

Definition von Testreihenfolge, Testaufwand und Testtiefe

Typischerweise stehen Testmanager bei der Testplanung vor dem Problem, dass nicht alle Testziele mit gleicher Intensität verfolgt werden können. Es besteht ein hohes Risiko, dass gegen Ende des geplanten Testzeitraums Aufgaben unerledigt bleiben, weil sich Zulieferungen verzögern, geplante Ressourcen nicht zur Verfügung stehen oder das zu testende System so instabil ist, dass der Testbetrieb verlangsamt wird. Die Aufgaben im Testprojekt müssen also priorisiert werden.

Hierzu bedarf es zweier Betrachtungen:

- Welche Faktoren werden herangezogen, um die Priorität einer Aufgabe zu definieren? Nachdem Testen ein risikominderndes Verfahren darstellt, ist der wesentliche Faktor das Produktrisiko. In den Abschnitten 4.2 und 4.3 wird darauf im Detail eingegangen.

- Welche Steuergrößen im Testprojekt sollen durch die Priorisierung wie beeinflusst werden? Grundsätzlich können folgende Parameter gesteuert werden:
  1. Die Reihenfolge, in der die Testziele abgearbeitet werden sollen. Prinzipiell ist es aus fachlicher Sicht sinnvoll, diejenigen Testaktivitäten zuerst durchzuführen, die die wichtigsten Eigenschaften des Systems prüfen. Zusätzlich gibt es aber meistens technische Abhängigkeiten, die die Reihenfolgeplanung beeinflussen, und nicht zuletzt bildet die Reihenfolge der Lieferung der Leistungsmerkmale seitens der Entwicklung eine wichtige Planungsgrundlage.
  2. Der Aufwand, der für die einzelnen Testziele angesetzt werden soll. Dies umfasst Aufwände für Testvorbereitung und -durchführung. Bei der Berücksichtigung der hochprioren Testziele mit hohem Aufwand ist in der Planung zu bedenken, dass auch andere Faktoren wie technische Randbedingungen, Komplexität und Testbarkeit der zu testenden Funktionalitäten etc. einen Einfluss haben.
  3. Die Testentwurfsverfahren, die eingesetzt werden sollen. Wie im vorigen Abschnitt erläutert, eignen sich verschiedene Testentwurfsverfahren unterschiedlich gut zur Prüfung unterschiedlicher Qualitätsmerkmale. Außerdem sind die meisten Testentwurfsverfahren skalierbar, d.h., sie können mit mehr oder weniger Aufwand eingesetzt werden, und erzeugen dadurch eine höhere oder niedrigere Testintensität.

Eine Priorisierung, die ausschließlich die Reihenfolge der Aktivitäten bestimmt, kann dazu führen, dass bei Zeitmangel Testziele komplett unbearbeitet bleiben. Um dies zu vermeiden, kann jedem Testziel der (maximal) zu erbringende Aufwand zugeordnet werden. Dies erfolgt z.B., indem die zur Verfügung stehenden Ressourcen proportional zum ermittelten Risikoindex (ermittelt durch das Schaefer- oder Gutjahr-Verfahren, s. Abschnitte 4.3.2 und 4.3.3) auf die Testziele verteilt werden. Auch niedrig priorisierte Testziele erhalten damit ein Mindestbudget. Zur Feinjustierung empfiehlt es sich schließlich, gestaffelt nach Priorität die Anwendung bzw. Ausprägung der einzusetzenden Testentwurfsverfahren festzuschreiben. Hoch priorisierten Testzielen werden aufwendigere Testentwurfsverfahren zugeordnet, was erfahrungsgemäß zu einer höheren Fehlerfindung führt.

Beispiel: Priorisierung bei VSR DreamCar

Für *DreamCar* werden mit dem Verfahren nach Hans Schaefer (s. Abschnitt 4.3.2) die Risiken der wichtigsten Anwendungsfälle analysiert. Es entsteht folgende Priorisierung:

Anwendungsfall	Priorität	Reihenfolge	Budget (Stunden)	Testentwurfsverfahren
Fahrzeugmodell anlegen	322	1	9	exploratives Testen
Sondermodell anlegen	122	2	3	Smoke-Test
Zubehörteil anlegen	78	3	2	Smoke-Test
Fahrzeug konfigurieren	677	4	18	Äquivalenzklassenkombination

Bei der Testplanung kann die Priorität nicht als alleiniges Kriterium zur Reihenfolgenbildung herangezogen werden, weil die Funktionen aufeinander aufbauen (so muss z.B. zuerst ein Fahrzeugmodell angelegt sein, damit es bei der Konfiguration eines Fahrzeugs verwendet werden kann). Daher entschließt sich der Testmanager, zusätzlich den maximal zu erbringenden Testaufwand zu planen (damit die Tester bei den weniger riskanten Funktionen nicht zu viel Zeit verbringen) und die anzuwendenden Testentwurfsverfahren zu definieren (um die Verwendung des Zeitbudgets sinnvoll zu steuern).

2.1.4 Planung und Koordination der Teststufen

Die Teststrategie und die →Testaufwandsschätzung (vgl. Kap. 5) müssen von Testmanagern auf die Teststufen heruntergebrochen und zu einem detaillierten Projektplan weiterentwickelt werden. Ergebnis dieser Detailplanung kann die Erkenntnis sein, dass die zur Verfügung stehende Zeit oder die Ressourcen nicht ausreichen, um die Teststrategie komplett abzuarbeiten. In diesem Fall müssen ausgehend von den Prioritäten an geeigneter Stelle Kürzungen vorgenommen, Termine verschoben oder weitere Ressourcen angefordert werden.

Ende- und Abnahmekriterien

Endekriterien und Metriken gehören zusammen.

Für jede Teststufe und für das gesamte Testprojekt muss klar geregelt werden, wann ein Testobjekt für den Test als tauglich befunden werden kann und die entsprechenden Testaktivitäten beginnen können. Ebenso ist festzulegen, wann es als ausreichend getestet zu betrachten ist und somit die zugehörigen Testaktivitäten beendet werden können. Hierfür sind adäquate Eingangs- und Endekriterien erforderlich. Auf höheren Teststufen kommen auch Abnahmekriterien zum Tragen, die

ein System oder eine Komponente erfüllen muss, um eine Abnahme durch den Benutzer, Kunden oder eine bevollmächtigte Instanz erfolgreich abschließen zu können. Diese Kriterien sind für jeden Testfall auf entsprechende »bestanden/nicht bestanden«-Kriterien abzubilden.

Grundsätzlich ist diese Frage mit der Definition und Anwendung passender Metriken verknüpft; Kapitel 7 nennt entsprechende Metriken und Kapitel 8 erläutert deren Anwendung im Rahmen der Teststeuerung und -berichterstattung.

Die Eingangs-, Ende- und Abnahmekriterien sollen sich, ebenso wie die Teststrategie, am Projekt- und Produktrisiko des zu testenden Systems orientieren. Dies betrifft sowohl die Schärfe der anzuwendenden Kriterien (also der erforderlichen Exaktheit der Metriken) als auch die Definition der Freigabeschwellen auf Basis der gewählten Metriken (also der zu erzielenden Messwerte, die für ein Testende bzw. eine Abnahme vorliegen müssen).

Manchmal ist es sinnvoll, zwischen Endekriterien und Abnahmekriterien wie folgt zu unterscheiden: Endekriterien werden in jedem Testzyklus angewendet, um dessen Ende zu bestimmen. Sie beziehen sich im Allgemeinen ausschließlich auf den Testfortschritt, aber nicht auf die einzelnen Ergebnisse der Tests. Abnahmekriterien dienen dagegen dazu, die Lieferfähigkeit der Testobjekte an die jeweils nächste Teststufe bzw. an den Kunden zu bestimmen. Diese umfassen ggf. deutlich weiter greifende Kriterien, wie beispielsweise die Ergebnisse der Tests, das Vorhandensein von Releasedokumenten, offizielle Freigabeunterschriften etc.

Abbruchkriterien und Wiederaufnahmebedingungen

Ist ein Testobjekt nicht reif genug, um eine Teststufe vollständig zu durchlaufen (d.h. die Endekriterien können nicht erreicht werden), so besteht die Gefahr, dass innerhalb der Teststufe bei der Abarbeitung des Stufentestplans sinnlos Ressourcen verschwendet werden. Sinnvollerweise ist in einer solchen Situation das Testobjekt zurückzuweisen und für eine Nachbesserung durch Entwicklung und vorgelagerte Teststufen zu sorgen.

Testabbruchkriterien und Testeingangskriterien ergänzen sich.

Eine Teillösung dieses Problems besteht in der Definition und Anwendung von Abbruchkriterien. Ein Beispiel hierfür ist das Überschreiten einer bestimmten Anzahl auftretender Fehlerwirkungen in der Testeingangsprüfung (*Smoke-Test*). Die Abbruchkriterien werden ergänzt durch die Eingangskriterien für die jeweilige Teststufe, durch deren Anwendung das Eintreten einer solchen Situation verhindert oder die Wahrscheinlichkeit dafür zumindest reduziert wird. Eingangs-

kriterien sind ebenso als Wiederaufnahmekriterien einer unterbrochenen Teststufe geeignet (s. Abschnitt 10.3 für weitere Informationen).

2.1.5 Zeit- und Aktivitätenplanung

Zunächst wird ein grober Zeitplan aller Testaktivitäten auf Meilensteinebene, der sogenannte →Mastertestplan, angelegt. Die Meilensteine umfassen geplante Fertigstellungstermine für Dokumente, Zieltermine für (Teil-)Freigaben der Testobjekte und deren Übergabetermine von der Entwicklung an den Test.

Vom Meilensteinplan (Ziel) wird der Stufentestplan (Umsetzung) abgeleitet.

Ausgehend von diesen Meilensteinen wird dann durch Hinzunahme der Ergebnisse der Aufwandsabschätzung für jede Teststufe eine erste Version des ressourcenbezogenen Zeit- und Aktivitätenplans (im Folgenden kurz →Stufentestplan genannt) ausgearbeitet. Die Ergebnisse dieser Planung müssen wiederum so früh wie möglich in den Gesamtplan des Projektmanagers Eingang finden.

Abstimmung zwischen Entwicklungs- und Testplan

Eine regelmäßige Abstimmung zwischen Entwicklungsleitern und Testmanagern muss stattfinden: Testmanager müssen über Plan- und Isttermine der Entwicklung sowie Inhalte der Lieferstände informiert werden und auf diese in der Detailtestplanung reagieren. Entwicklungsleiter wiederum müssen auf Testergebnisse reagieren und ggf. Meilensteine verschieben, wenn zusätzliche Korrekturzyklen vorgesehen werden müssen. Entwicklungsplan und (Stufen-)Testplan sollen unter Einbindung des jeweiligen Gegenparts regelmäßig korrigiert, detailliert und gereviewt werden. Beispielsweise kann die Planungsinformation aus dem Test genutzt werden, um die Lieferreihenfolge von Komponenten des zu testenden Systems optimal auf die geplante Strategie beim Integrationstest dieser Komponenten abzustimmen. Auf diese Weise wird der →Testplan zu einem effektiven Kommunikationsmittel.

Typische Inhalte eines Stufentestplans

Typischerweise findet ein Stufentestplan seine Realisierung in einem mit Zusatzinformationen hinterlegten →Gantt-Diagramm, das folgende Themen abhandelt:

Inhalte

- Zuordnung der inhaltlichen Schwerpunkte zu den Testzyklen, abhängig vom voraussichtlich implementierten Funktionsumfang des jeweiligen Testobjekts. Hier wird i. Allg. eine Mischung aus erstmaligem Test neuer Funktionalitäten, Fehlernachtests von in den vorigen Zyklen fehlerbehafteten Bereichen und Regressionstests der übrigen Bereiche angestrebt.
- Schrittweise Verfeinerung der zugewiesenen Schwerpunkte pro Zyklus zu einer Liste aller geplanten Tests, ausgehend von den im Testkonzept genannten Testzielen. In längeren zyklisch ablaufenden Testprojekten wird sich diese Verfeinerung nicht nur auf die

Durchführung von Tests, sondern auch auf die zeitgleiche Spezifikation und ggf. Automatisierung von Tests für diejenigen Testfälle erstrecken, die in den jeweils folgenden Testzyklen zur Durchführung bereitgestellt werden sollen.

- Ergänzende Planung von Phasen, in denen ad hoc bzw. explorativ getestet werden soll, soweit die zur Verfügung stehenden Ressourcen dies zulassen und der Mehrwert der explorativen Tests darstellbar ist.
- Regelung von Zuständigkeiten im Testprojekt wie z.B. das Management, die Spezifikation, die Vorbereitung, Durchführung und Nachbereitung der Tests. Weitere Zuständigkeiten betreffen die testprojektinterne Qualitätssicherung, das Konfigurationsmanagement und die Bereitstellung und Wartung der Testumgebung. Die organisatorische Einbindung des Testpersonals in das Gesamtprojekt, die Weisungsbefugnis im Testteam und ggf. die Aufteilung/Organisation des Testteams in verschiedene Testgruppen und/oder Teststufen müssen hier geregelt werden.

Tipp

Zuständigkeiten umfassen auch das Dokumentenmanagement.

Meistens sind diese Aktivitäten mit der Erzeugung von Dokumenten verbunden (s. Kap. 6). Werden die Zuständigkeiten nur auf der Ebene »Personen X, Y und Z haben die Rolle R« geregelt, dann reicht dies nicht aus, um die termingerechte Entstehung und die adäquate Qualität der Dokumente sicherzustellen. Zusätzlich muss für jedes Dokument explizit personell geregelt werden,

- wer verantwortlicher Autor ist und wer in welcher anderen Rolle am Dokument mitwirkt,
- wie, wann und durch wen die Freigabe des Dokuments erfolgt,
- wie, wann sowie an wen es nach Freigabe zu verteilen ist.

Zeiten

- Angabe von Beginn und Ende der einzelnen Testzyklen, abhängig von den entwicklungsseitigen Lieferdaten für die zu testenden »Builds« der Testobjekte.
- Zuordnung von Terminen für die einzelnen Testaufgaben bzw. Phasen innerhalb der Testzyklen, z.B. der →Eingangstests bei der Übernahme der Testobjekte aus der vorgelagerten Teststufe, der →Freigabetests vor der Weitergabe an die nachfolgende Teststufe, der übrigen Tests in der Reihenfolge ihrer Priorität, die voraussichtlich durchzuführenden Fehlernachtests etc.

Ressourcen

- Zuordnung der Aufgaben zu den Mitgliedern des Testteams unter Berücksichtigung und Dokumentation des jeweiligen anwendungs- oder testaufgabenspezifischen Know-hows, z.B. für explorative oder auch Ad-hoc-Tests, Lasttests etc.

- Angabe des Arbeitsaufwands für die Aktivität.
- Belegungsplan für die Testinfrastruktur (Hard- & Software, Testmittel wie Lastgeneratoren, Usability-Labor etc.).

Abhängigkeiten

- Definition einer effizienten Testreihenfolge zur Minimierung von redundanten Aktionen (beispielsweise durch Ausführen von Testfällen, die Datenobjekte kreieren, vor solchen Testfällen, die diese Daten manipulieren oder löschen; so dienen die ersten Testfälle als Vorbedingungen für die weiteren).
- Nennung von notwendigen externen Zulieferungen wie Testdatenbanken etc.

2.1.6 Sicherstellen der Rückverfolgbarkeit

Frage: Welche Information wird wo berücksichtigt?

Insbesondere zur Überwachung und Steuerung des Tests stehen Testmanager immer wieder vor Fragen wie »Welcher Stakeholder hat wann eine bestimmte Anforderung formuliert?«, »Welche Anforderungen und welche Produktrisiken sind durch eine bestimmte Testbedingung berücksichtigt?« oder »Welche Testfälle beziehen sich auf eine bestimmte Anforderung?«. Zurückführen lassen sich solche Fragen oft auf die generelle Fragestellung: »Wie hängt eine bestimmte Information bzw. ein bestimmtes Dokument (oder »Artefakt«) mit anderen Dokumenten zusammen?« Hierzu müssen die Zusammenhänge aller Entwicklungsartefakte bekannt sein, also ihre jeweiligen Ursprünge und weiteren Verwendungen.

Antwort: Rückverfolgbarkeit sicherstellen

Das Identifizieren aufeinander aufbauender bzw. (inhaltlich) zusammenhängender Artefakte wird als →Rückverfolgbarkeit (engl. *traceability*, auch: Verfolgbarkeit, Nachvollziehbarkeit) bezeichnet (vgl. [IEEE 610.12], [URL: SEVOCAB]). Hierfür ist festzuhalten, welche Anforderungen wann von wem formuliert, mittels welcher fachlichen und technischen Entwurfsdokumente spezifiziert, durch welche Programmteile realisiert und aufgrund welcher Testbedingungen mit welchen Testfällen getestet wurden. Erst mit diesen Informationen können z.B. entsprechende Abdeckungsmaße für Endekriterien vorgegeben und ermittelt werden.

Verfolgbarkeit wird bezüglich der zeitlichen und inhaltlichen Abhängigkeit in die horizontale und die vertikale Verfolgbarkeit unterschieden (vgl. [Davis 90]).

Horizontale Verfolgbarkeit

Die horizontale Verfolgbarkeit setzt einerseits Elemente der gleichen Entwicklungstätigkeit in einen (zeitlichen) Bezug. In diesem Zusammenhang wird auch von Versionen eines Artefakts gesprochen. Andererseits beschreibt die horizontale Verfolgbarkeit, wie bestimmte Teile von Artefakten innerhalb ein- und derselben Entwicklungstätigkeit durch andere verfeinert bzw. präzisiert werden.

Vertikale Verfolgbarkeit = Rückverfolgbarkeit

Die vertikale Verfolgbarkeit betrachtet die Relationen zwischen Elementen verschiedener Entwicklungstätigkeiten und gibt Antwort auf die Frage, wie ein bestimmtes Element (z.B. eine Klasse im Code) aus einem anderen (z.B. einer Entwurfsklasse) hervorgegangen ist oder mit welchen Testfällen es geprüft wurde. Während die horizontale Verfolgbarkeit Gegenstand des Konfigurations- und Versionsmanagements ist, steht die vertikale Verfolgbarkeit im Zentrum obiger Fragen des Testmanagements. Rückverfolgbarkeit meint also in der Regel die vertikale Verfolgbarkeit.

Richtungen der Rückverfolgbarkeit

Relativ zu den Anforderungen lassen sich mit der obigen Definition der IEEE 830 die in Abbildung 2–7 gezeigten vier Richtungen der (vertikalen) Verfolgbarkeit ableiten (nach [Davis 90]). Bei den ersten beiden Richtungen wird auch von Vor-Verfolgbarkeit (*pre-traceability*) gesprochen, bei den letzten beiden dementsprechend von Nach-Verfolgbarkeit (*post-traceability*).

Erfolgt die Dokumentation und Verwaltung der Anforderungen über einfache Werkzeuge z.B. zur Textverarbeitung oder Tabellenkalkulation, muss die Verfolgbarkeit manuell über entsprechende Hyperlinks realisiert werden, was in großen Projekten schnell zu Konsistenzproblemen führt. Spezielle Werkzeuge zum Anforderungsmanagement und zum Testmanagement bieten in der Regel komfortablere Möglichkeiten zur Verlinkung von Anforderungen mit Spezifikationen und weiteren Artefakten, über die die Rückverfolgbarkeit realisiert wird.

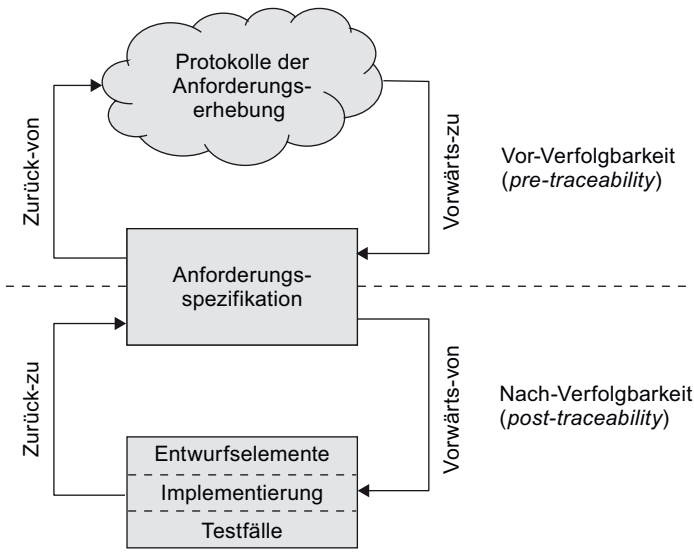

Abb. 2–7
Richtungen der Rückverfolgbarkeit

Technisch kann die Rückverfolgbarkeit prinzipiell auf drei unterschiedliche Arten realisiert werden:

- Durch eine Rückverfolgbarkeitsmatrix,
- durch entsprechende Schlüssel in einer Datenbank bzw. einem Wiki[3]-basierten Dokumentationssystem und
- in einem Werkzeug zum Testmanagement oder zum Anforderungsmanagement.

Matrixbasierte Rückverfolgbarkeit

Die einfachste und nur für kleinere Projekte zu empfehlende Möglichkeit bietet eine z. B. im Tabellenkalkulationsprogramm zu pflegende Matrix, in der für jede Anforderung eine Zeile angelegt wird. In den Spalten werden sowohl die Vorgängerdokumente als auch die darauf aufbauenden Artefakte aus Entwicklung und Test aufgeführt. Tabelle 2–1 zeigt eine solche Rückverfolgbarkeitsmatrix im VSR-Projekt. Vor den Referenzen auf abhängige Dokumente ist in der Spalte »Status« der Bearbeitungszustand der jeweiligen Anforderung aufgeführt.

Tab. 2–1 *Matrix zur Rückverfolgbarkeit im VSR-Projekt*

Anf.-ID	Beschreibung	Status	Quell-dokument	Review-protokolle	Entwurfs-komponenten	Code-einheiten	Testziele	Testbe-dingungen	Testfälle	Test-protokolle
F01.001	Fahrzeugmodell anlegen	Komp.getestet	ANFWS001	RA130218 RA130315 RC130322 RA130403	K01	C01_001 C01_002	TZ01_001	TB01_001 TB01_002	TF01_001 TF01_002 TF01_003	TP130503
F01.002	Fahrzeugmodell ändern	Komp.getestet	ANFWS001	RA130218 RA130315 RA130403	K01	C01_002	TZ01_001	TB01_001 TB01_003	TF01_001 TF01_002	TP130503
F01.003	Fahrzeugmodell löschen	Komp.getestet	ANFWS001	RA130218 RA130315 RA130403	K01	C01_001	TZ01_001	TB01_001 TB01_002 TB01_005	TF01_001 TF01_004	TP130503
F01.004	Fahrzeugmodell suchen	Integrat.test	ANFWS001 ANFWS002	RA130218 RA130315 RA130403	K01	C01_004	TZ01_001	TB01_001 TB01_004	TF01_005 TF01_006	TP130504
F02.001	Sondermodell anlegen	Implementierung	ANFWS001	RA130218 RA130315	K02	C02_001 C02_002	TZ02_001			
F02.002	Sondermodell ändern	Implementierung	ANFWS001	RA130218 RA130315	K02	C02_002	TZ02_001			
F02.003	Sondermodell löschen	Implementierung	ANFWS001	RA130218 RA130315	K02	C02_001	TZ02_001			
F02.004	Sondermodell suchen	Implementierung	ANFWS001	RA130218 RA130315	K02	C02_004	TZ02_001			
F03.001	Zubehörteil anlegen	Spezifiziert	ANFWS002	RA130219						
F03.002	Zubehörteil ändern	Spezifiziert	ANFWS002	RA130219						
F03.003	Zubehörteil löschen	Spezifiziert	ANFWS002	RA130219						
F03.004	Zubehörteil suchen	Spezifiziert	ANFWS002	RA130219						
F03.005	Zubehörteil zuordnen	Spezifikation	ANFWS002	RA130219						
F04.001	Fahrzeug konfigurieren	Entwurf	ANFWS002 ANFWS003	RA130219	K04		TZ04_001			
F04.002	Fahrzeug speichern	Entwurf	ANFWS002 ANFWS003	RA130219	K04		TZ04_002			
F04.003	Fahrzeug laden	Entwurf	ANFWS002 ANFWS003	RA130219	K04		TZ04_002			
F04.004	Fahrzeug ändern	Entworfen	ANFWS002	RA130219	K04		TZ04_001			
F04.005	Fahrzeug löschen	Entworfen	ANFWS002	RA130219	K04		TZ04_002			
…										

3. Ein Wiki (auch WikiWeb) ist ein Hypertextsystem, dessen Inhalte über Schlüsselworte automatisch verlinkt werden [URL: Wiki].

Schlüsselbasierte Rückverfolgbarkeit

In größeren Projekten empfiehlt es sich, die Informationen zur Rückverfolgbarkeit in einer speziellen Datenbank zu pflegen, sodass bessere Abfragemöglichkeiten realisiert werden können. Die Rückverfolgbarkeit wird dann durch Fremdschlüssel auf Datenbankeinträge der abhängigen Artefakte ermöglicht. In Wiki-basierten Dokumentationssystemen kann die Rückverfolgbarkeit über eindeutige Bezeichner der Artefakte realisiert werden, die dann jeweils als Metadaten (*tag*) den abhängigen Artefakten hinzugefügt werden.

Werkzeuggestützte Rückverfolgbarkeit

Letztendlich unterstützen auch Werkzeuge zum Testmanagement (s. Kap. 15) sowie zum Anforderungsmanagement die Rückverfolgbarkeit. Neben dem automatischen Erzeugen entsprechender eindeutiger Bezeichner werden auch die Zuordnung abhängiger Artefakte sowie Konsistenz- und Vollständigkeitsprüfungen der Abhängigkeiten unterstützt. Darüber hinaus bieten die meisten Werkzeuge auch entsprechende Metriken und Vorlagen zur Erstellung von Berichten zum Status der Artefakte sowie der Rückverfolgbarkeit selbst an.

2.1.7 Definition der Testumgebung

Testplattform = Basis-SW und -HW und sonstige Voraussetzungen

Um dynamische Tests auf Testobjekten ausführen zu können, wird eine Umgebung benötigt, die Hardware, Instrumentierung, →Simulatoren, Softwarewerkzeuge und andere unterstützende Hilfsmittel umfasst. Das Testkonzept muss darlegen,

- welche Anforderungen an die Testplattform, z. B. in Form von Hardware, Betriebssystem und sonstiger Basissoftware, bestehen,
- in welchem Umfang entsprechende Ressourcen bereitzustellen oder zu beschaffen sind und
- welche sonstigen Voraussetzungen, z. B. Referenzdatenbanken oder Konfigurationsdaten, für den Test erforderlich sind.

Testschnittstellen und Werkzeuge

Darüber hinaus müssen auch die folgenden Themen behandelt werden:

- Welches sind die »Points of Control« und die »Points of Observation«, d. h., über welche Schnittstellen wird das Testobjekt bei der Testdurchführung stimuliert bzw. seine Reaktionen überprüft?
- Welche sonstige Soft- und Hardware ist dafür erforderlich? Hierzu gehören beispielsweise Monitoring- und Protokollumgebungen, Simulatoren, Debugger, Signalgeneratoren sowie ggf. eine Testumgebung, die der künftigen →Produktionsumgebung möglichst weitgehend entspricht (für Last- und Performanztests). Auch ein Usability-Labor für Bedienbarkeitstests mag notwendig sein. Zeitbedarf und Kosten für Anschaffung, Einführung und Betrieb dieser Hard-

und Software und die ggf. notwendige Ausbildung des Personals müssen in die Aktivitätenplanung und in die Aufwandsabschätzung (s. Kap. 5) einfließen. Die Einführung oder ggf. Eigenentwicklung von Werkzeugen muss wegen des potenziell hohen Aufwands strategisch verfolgt werden (s. Abschnitte 15.3 und 15.4).

- Welche Ansätze zur Testautomatisierung sind in dieser Umgebung möglich und sollen ggf. genutzt werden? Für datengetriebene Tests ist beispielsweise der Zugriff auf die Testdatenbank wichtig, für schlüsselwortgetriebene Tests die Verwaltung der Schlüsselwort-Bibliotheken.
- Sind zur Beschaffung der notwendigen Testdaten besondere Maßnahmen erforderlich? Für Tests mit produktionsnahen Daten spielen beispielsweise Datenschutzaspekte eine wichtige Rolle. In solchen Fällen ist ggf. eine werkzeuggestützte Anonymisierung der Echtdaten oder die Generierung künstlicher Datenbestände einzuplanen.

Die Antworten auf diese Fragen können für das gleiche Testobjekt je nach Teststufe durchaus unterschiedlich ausfallen. So wird die Software eines eingebetteten Systems während des Komponententests evtl. in einer Hostentwicklungsumgebung unter Zuhilfenahme einer typischen Komponententestumgebung wie CPPUnit [URL: CPPUnit] oder Tessy [URL: Tessy] getestet. Dagegen wird im Systemtest aber auf der Target-Hardware integriert in einem speziell konfigurierten Testrahmen unter Verwendung typischer Umgebungen für den Test eingebetteter Systeme wie z.B. Simulatoren oder Prüfstände. Die Definition der Testumgebungen wird dabei zum einen beeinflusst durch das technisch Machbare, zum anderen aber auch durch die Testaufgaben und daraus resultierenden Testentwurfsverfahren und zu guter Letzt auch durch die zur Verfügung stehende und budgetierbare Hard- und Software sowie die darüber vorhandenen Kenntnisse der Testmitarbeiter.

Ressourcenmanagement für die Bestandteile der Testumgebung

Eine Ressourcenverwaltung (*assett management*) – sei es über einen einfachen Belegungsplan in Form einer Tabellenkalkulation, eine Ressourcennutzungsdatenbank oder die Abbildung in einem Projektmanagementwerkzeug – ist bei komplexen Testumgebungen angeraten. Teil des Risikomanagements (s. Abschnitt 4.2) müssen dann auch Strategien zur Behandlung fehlender Ressourcen sein. Gerade die produktionsnahe Testumgebung für nicht funktionale Tests ist ggf. sehr teuer und daher zeitlich nur begrenzt verfügbar. Eventuell steht gar keine eigene Testumgebung zur Verfügung, sodass der Zugriff auf die Produktionsumgebung eingeplant werden muss. Auch die Werkzeuge für den Lasttest sind ein hoher Kostenfaktor, da die sogenannten → »virtuellen Benutzer«, die zur Lastgenerierung benötigt werden, von

den Werkzeugherstellern meistens nach Anzahl der parallelen simulierten Benutzer und der Dauer der Nutzung lizenziert werden.

Testumgebung und Testmittel dem Konfigurationsmanagement unterwerfen

Das Management der Testumgebung sowie der Testmittel erfordert neben der Ressourcenkoordination auch die Einbindung ins testseitige Versions- und Konfigurationsmanagement. Nur zu oft ist die Lauffähigkeit der Testumgebungen von einer exakten Kombination bestimmter Versionen von Betriebssystem, Testautomatisierungswerkzeug, Analysewerkzeug und dem Stand des Testobjekts abhängig. In einigen Branchen wie Medizintechnik, Automobil-, Luftfahrt- und Raumfahrtindustrie oder auch Bahntechnik erfordern strenge Nachweispflichten die Aufbewahrung aller Testumgebungsbestandteile sowie der Testmittel für zehn oder mehr Jahre (s. hierzu Abschnitt 11.4). Auch dies ist im Testkonzept zu berücksichtigen.

Tipp

Ein effizientes Management von Testplattformen ist ein leicht umzusetzender Faktor zum Effizienzgewinn. Durch projektübergreifende Bereitstellung von schnell einzuspielenden »Images« für Testhardware mit oft benötigten Kombinationen aus Betriebssystem, Testsoftware und anderen Plattformbestandteilen (z. B. Office-Paket) und den Einsatz einer entsprechenden Imaging-Software und Image-Management-Umgebung [URL: Imaging] lassen sich in Testcentern die Rüstzeiten für Testläufe signifikant reduzieren. In Kombination mit Virtualisierungslösungen wie VMWare oder Plex86 [URL: Virtualization] lassen sich diese Standard-Images auf virtuelle Hardware abbilden und somit weitgehend unabhängig von der kurzlebigen Hardwareentwicklung im PC-Sektor machen. Die Technologie kann außerdem eingesetzt werden, um *snapshots* des Rechnerzustands vor Beginn längerer Testläufe anzufertigen und so Wiederaufsetzpunkte zu erzeugen, auf die bei Scheitern des Testlaufs zurückgefallen werden kann.

Frühzeitig an Testinfrastruktur denken

Die Testumgebung ist Teil der gesamten Testinfrastruktur, die alle Ressourcen umfasst, die für die Durchführung des Tests benötigt werden. Hierzu gehören z. B. Testwerkzeuge, Büroräume und Drucker- und Kopiergeräte, Kommunikationseinrichtungen, Datenbanken und alles, was sonst zur Durchführung der Testfälle notwendig ist, ebenso wie bestimmte regulatorisch geforderte Verfahren wie brandgesicherte Archive oder sichere Zugriffsberechtigungen. Auch diese sind bei der Testplanung zu berücksichtigen. Oft erfolgt dies im Rahmen der Managements der gesamten IT-Infrastruktur, wobei Standards wie ITIL oder ISO 20000 hilfreiche Referenzen darstellen ([URL: ITIL], [ISO 20000]).

2.1.8 Vorteile frühzeitiger Testplanung

»Moment of involvement«

Die Testplanung kann – und muss – sehr früh beginnen, und zwar parallel zu den frühen Phasen der Softwareentwicklung, d. h. der Anforderungsdefinition und dem funktionalen Systementwurf, auch wenn zu diesem Zeitpunkt noch nicht alle Informationen vorliegen, die zum Abschluss der Planung erforderlich sind. Sowohl moderne Entwicklungs- und Testvorgehensmodelle wie das W-Modell (s. Abschnitt 3.3.2) als auch anerkannte Prozessverbesserungsmodelle wie TPI® Next (s. Abschnitt 14.3.2) betonen die Wichtigkeit dieses frühzeitigen »moment of involvement«[4].

Zeitgewinn

Da der Test in der Regel mehrere Testzyklen und damit auch Planungsphasen umfasst, kann die erste Testgrobplanung stufenweise ergänzt und konkretisiert werden. Basierend auf den bereits stabilen Teilplandaten kann dazu mit Aktivitäten wie der Erstellung der Testspezifikation, dem Einrichten der Testumgebung und sogar der teilweisen Automatisierung von Tests begonnen werden. Das Testteam gewinnt Zeit und kann idealerweise bereits bei Vorliegen der ersten Testobjekte mit der Testdurchführung beginnen. Das Testen befindet sich damit wesentlich kürzer auf dem kritischen Pfad des Gesamtvorhabens.

Frühere Fehlerfindung

Außerdem werden oftmals grundlegende Fehler in den Testobjekten nicht erst bei der Durchführung, sondern bereits während der Testanalyse und des Testentwurfs entdeckt. Die Behebungskosten für solche Fehler sind umso niedriger, je früher dies geschieht.

Risikosenkung

Die frühzeitige Planung hilft, Projektrisiken zu senken:

- Der Testplan kann frühzeitig durch Reviews anderer Projektbeteiligter geprüft werden. Potenzielle Inkonsistenzen, vergessene Themen etc. werden so schneller adressiert.
- Für den Test notwendige Beistellungen oder Mitwirkungsleistungen werden rechtzeitig kommuniziert.

Rechtzeitiges Signalisieren von Ressourcenbedarf

- Nicht zuletzt erleichtert eine rechtzeitig vorliegende Testplanung die Beschaffung der notwendigen Ressourcen, insbesondere wenn der Bedarf höher ausfällt, als bei Projektbeginn abzusehen war.

4. Begriff aus TPI Next® [van Ewijk 11].