
# CERTIFIED TESTER

# Advanced Level Syllabus

Version 2007

International Software Testing Qualifications Board

Deutschsprachige Ausgabe
German Testing Board e.V.
in Kooperation mit dem Swiss Testing Board
und dem Austrian Testing Board

Certified Tester

Advanced Level Syllabus
(Deutschsprachige Ausgabe)




Mitglieder der Advanced Level Arbeitsgruppe: Bernard Homès (Leitung), Graham Bath, Rex Black, Sigrid Eldh, Jayapradeep Jiothis, Paul Jorgensen, Vipul Kocher, Judy McKay, Klaus Olsen, Randy Rice, Jürgen Richter, Eric Riou Du Cosquer, Mike Smith, Geoff Thompson, Erik Van Veenendaal; 2006-2007.

**Übersetzung des englischsprachigen Lehrplans des International Software Testing Qualifications Board (ISTQB®), Originaltitel: Certified Tester, Advanced Level Syllabus.**


Übersetzung/Übertragung in die deutsche Sprache, 2007/2008: Horst Pohlmann (GTB), Thomas Müller (STB), Graham Bath (GTB, Leitung) mit Unterstützung durch das Übersetzungsbüro: Elke Bath und der Technical Writerin: Dagmar Boedicker.

Die Autoren danken den folgenden Reviewern: Petra Bukowski (GTB), Matthias Hamburg (GTB), Horst Pohlmann (GTB), Timea Illes-Seifert (GTB), Helmut Pichler (ATB), Thomas Müller (STB), Anton Schlatter (GTB), Maud Schlich (GTB), Stephanie Ulrich (GTB), Sabine Uhde (GTB), Uwe Hehn (GTB), Martin Klonk (ATB), Wiltrud Breuss (ATB), Harry Sneed (ATB) und,Kurt Aigner (ATB).

# Änderungsübersicht der deutschsprachigen Ausgabe

| Version | Datum | Bemerkungen |
|---|---|---|
| V2007Beta | 6. Juni 2008 | Certified Tester Advanced Level Syllabus Version 2007 (Lehrplan Aufbaukurs Certified Tester) |
| V2007Beta2 | 1.Oktober 2008 | Abgleich mit CTFL |
| V2007Beta3 | 25.August 2009 | Abgleich mit Glossar 2.0<br>Freigabeprüfung |
| V2007 | Januar 2010 | Freigabe durch D.A.CH (nach internem Review). |

# Inhaltsverzeichnis

# Dank

Dieses Dokument wurde von einem Kernteam der Arbeitsgruppe „Advanced Level Syllabus" (Lehrplan Aufbaukurs) des International Software Testing Qualifications Board erstellt. Dieser Arbeitsgruppe gehörten an: Bernard Homès (Vorsitzender), Graham Bath, Rex Black, Sigrid Eldh, Jayapradeep Jiothis, Paul Jorgensen, Vipul Kocher, Judy McKay, Thomas Mueller, Klaus Olsen, Randy Rice, Jürgen Richter, Eric Riou Du Cosquer, Mike Smith, Geoff Thompson, Erik Van Veenendaal.

Die Mitglieder der Arbeitsgruppe bedanken sich beim Reviewteam und bei den nationalen Testing-Boards für die konstruktiven Verbesserungsvorschläge und Beiträge.

Bei Fertigstellung des Advanced Level Lehrplans hatte die Arbeitsgruppe „Advanced Level" die folgenden Mitglieder (in alphabetischer Reihenfolge):

Graham Bath, Robert Bender, Rex Black, Chris Carter, Maria Clara Choucair, Sigrid Eldh, Dorothy Graham, Bernard Homès (Vorsitzender), Jayapradeep Jiothis, Vipul Kocher, Anastasios Kyriakopoulos, Judy McKay, Thomas Mueller, Klaus Olsen, Avinoam Porat, Meile Posthuma, Erkki Pöyhönen, Jürgen Richter, Eric Riou Du Cosquer, Jan Sabak, Hans Schaefer, Maud Schlich, Rajesh Sivaraman, Mike Smith, Michael Stahl, Geoff Thompson, Erik Van Veenendaal.

Folgende Personen haben an Review, Kommentierung und der Abstimmung über diesen Lehrplan mitgearbeitet:

Bernard Homès (Leitung)

Reto Armuzzi
Sue Atkins
Graham Bath
Paul Beekman
Armin Beer
Rex Black
Francisca Blunschi
Armin Born
Con Bracke
Chris Carter
Maria Clara Choucair
Robert Dankanin
Piet de Roo
Sigrid Eldh
Tim Edmonds
Erwin Engelsma
Graham Freeburn
Dorothy Graham
Brian Hambling
Jeff B Higgott
Bernard Homès
Rob Hendriks
Dr Suhaimi Ibrahim
Phillip Isles
Pr. Paul C. Jorgensen
Vipul Kocher
Anastasios Kyriakopoulos
Junfei Ma
Fergus McClachlan
Judy McKay
Don Mills
Gary Mogyorodi
Richard Morgan
Silvio Moser
Ernst Müller
Reto Müller
Thomas Müller
Peter Mullins
Beat Nagel
Richard Neeve
Klaus Olsen
Dale Perry
Helmut Pichler
Jörg Pietzsch
Avionam Porat
Iris Pinkster
Horst Pohlmann
Meile Posthuma
Eric Riou Du Cosquer
Stefan Ruff
Hans Schaefer
Maud Schlich
Rajesh Sivaraman
Mike Smith
Katja Stalder
Neil Thompson
Benjamin Timmermans
Chris van Bael
Jurian van de Laar
Marnix van den Ent
Mark van der Zwan
Stephanie van Dijck
Jan van Moll
Erik Van Veenendaal
Roland Weber
Phillip Whettlock
Derek Young
Mike Young
Wenqiang Zheng.

Dieses Dokument wurde von der Hauptversammlung des ISTQB® am 12. Oktober 2007 offiziell freigegeben.

# 0. Einführung in den Lehrplan

## 0.1 Das International Software Testing Qualifications Board

Das International Software Testing Qualifications Board (nachfolgend ISTQB® genannt) setzt sich zusammen aus den Mitgliedsboards verschiedener Länder und Regionen der ganzen Welt. Zum Zeitpunkt der Herausgabe der englischsprachigen Orginalausgabe des Dokuments (12, Oktober 2007) hatte das ISTQB® 33 Mitgliedsboards. Weitere Informationen über Aufbau und Mitgliedschaft im ISTQB® finden Sie unter www.ISTQB.org.

### *Zweck dieses Dokuments*

Dieser Lehrplan bildet die Grundlage für das Softwaretest-Qualifizierungsprogramm der Aufbaustufe (Advanced Level) des International Software Testing Qualifications Board. Das ISTQB® stellt den Lehrplan folgenden Adressaten zur Verfügung:

1. Nationalen/regionalen Boards zur Übersetzung in die jeweilige Landessprache und zur Akkreditierung von Ausbildungsanbietern. Die nationalen Boards können den Lehrplan an die eigenen sprachlichen Anforderungen anpassen sowie die Querverweise ändern und an die bei ihnen vorliegenden Veröffentlichungen angleichen.
2. Prüfungsinstitutionen zur Erarbeitung von Prüfungsfragen in der jeweiligen Landessprache, die sich an den Lernzielen der jeweiligen Module orientieren.
3. Ausbildungsanbietern zur Erstellung ihrer Kursunterlagen und zur Bestimmung einer geeigneten Unterrichtsmethodik.
4. Prüfungskandidaten zur Vorbereitung auf die Prüfung (als Teil des Ausbildungslehrgangs oder auch kursunabhängig).
5. Allen Personen, die im Bereich Software- und Systementwicklung tätig sind und die professionelle Kompetenz beim Testen von Software verbessern möchten, sowie als Grundlage für Bücher und Fachartikel.

Das ISTQB® kann auch anderen Personenkreisen oder Institutionen die Verwendung dieses Lehrplans für andere Zwecke genehmigen, wenn diese vorab eine entsprechende schriftliche Genehmigung einholen.

### *Certified Tester Advanced Level im Bereich Softwaretesten*

Die Aufbaustufe (Advanced Level) des Certified Tester Ausbildungsprogramms richtet sich an Personen, die eine fortgeschrittene Stufe in ihrem beruflichen Werdegang im Bereich Softwaretesten erreicht haben. Dazu gehören Personen in Rollen wie Tester, Test Analysts, Testingenieure, Testberater, Testmanager, Abnahmetester und Softwareentwickler. Die Aufbaustufe des Certified Tester Ausbildungsprogramms ist auch für Personen geeignet, die ein grundlegendes Verständnis über das Thema Softwaretesten erwerben möchten, beispielsweise Projektleiter, Qualitätsmanager, Softwareentwicklungs-Manager, Fachanalytiker, IT-Leiter oder Managementberater. Für die Zertifizierung als ISTQB® Certified Tester Advanced Level müssen die Prüfungskandidatinnen und -kandidaten das Zertifikat ISTQB® Certified Tester Foundation Level vorweisen und bei der für sie zuständigen Prüfungsinstitution ausreichende praktische Erfahrung nachweisen, um als für die Aufbaustufe (Advanced Level) qualifiziert zu gelten. Bitte wenden Sie sich an die für Sie zuständige Prüfungsinstitution, dort erfahren Sie mehr über die spezifischen Kriterien zum Nachweis der notwendigen praktischen Erfahrung.

## *Lernziele/Kognitive Ebenen*

Die Lernziele der jeweiligen Kapitel dieses Lehrplans sind so unterteilt, dass sie den einzelnen Modulen eindeutig zugeordnet werden können. Weitere Details und Beispiele für Lernziele enthält Abschnitt 0.3.

Der Inhalt des Lehrplans, die Begriffe und die wichtigsten Elemente aller aufgeführten Normen und Standards sollen zumindest wiedergegeben werden können (K1), auch wenn dies in den Lernzielen nicht ausdrücklich erwähnt wird.

## *Prüfung*

Alle Prüfungen für das Advanced Level Certificate müssen sich auf den vorliegenden Lehrplan und den Foundation-Level-Lehrplan beziehen. Antworten auf die Prüfungsfragen können Stoff aus mehreren Kapiteln dieses Lehrplans und des Foundation-Level-Lehrplans voraussetzen. Grundsätzlich können alle Kapitel beider Lehrpläne geprüft werden.

Das Format der Prüfung ist in den Richtlinien Advanced Exam Guidelines des ISTQB® festgelegt. Einzelne nationale Boards können auf Wunsch auch andere Prüfungsformate anwenden.

Die Prüfungen können im Rahmen eines akkreditierten Ausbildungslehrgangs abgelegt werden oder kursunabhängig, beispielsweise in einem Prüfungszentrum. Die Prüfungen können in Papierform oder elektronisch absolviert werden, jedoch auf jeden Fall mit Prüfungsaufsicht (d.h. Überwachung der Prüfung durch eine vom nationalen Board oder von der Prüfungsinstitution beauftragte Person).

## *Akkreditierung*

Ausbildungsanbieter, deren Ausbildungsunterlagen auf diesem Lehrplan basieren, müssen durch ein vom ISTQB® anerkanntes nationales Testing-Board akkreditiert werden. Die entsprechenden Akkreditierungsrichtlinien können vom zuständigen nationalen Testing-Board angefordert werden oder von der Organisation, die die Akkreditierung im Auftrag des nationalen Boards erteilt. Akkreditierte Kurse sind als zu diesem Lehrplan konform anerkannt und dürfen als Bestandteil des Lehrgangs die ISTQB® Prüfung enthalten.

Weitere Anleitungen enthält Anhang C – Hinweise für die Ausbildungsanbieter

## *Detaillierungsgrad*

Der Detaillierungsgrad dieses Lehrplans erlaubt konsistentes Lehren und Prüfen auf internationaler Ebene. Um dieses Ziel zu erreichen, enthält der vorliegende Lehrplan

- allgemeine Lernziele, die die Absicht des Advanced Level (Aufbaustufe) beschreiben,
- Lernziele für jeden Wissensbereich, die das zu erzielende kognitive Lernergebnis und und die zu erzielende Einstellung der Teilnehmer beschreiben,
- eine Liste zu lehrender Inhalte mit ihrer Beschreibung und, wo notwendig, Verweise auf weiterführende Quellen,
- eine Liste mit Begriffen, die die Teilnehmer wiedergeben können und verstanden haben müssen,
- eine Beschreibung der wichtigsten zu lehrenden Konzepte, einschließlich der Quellen, wie anerkannte Fachliteratur oder Normen und Standards.

Der vorliegende Lehrplan ist keine vollständige Beschreibung des Wissensgebiets Softwaretesten. Er gibt an, wie detailliert der Stoff in den Lehrgängen des Advanced Level zu behandeln ist.

## *Aufbau des Lehrplans*

Der Lehrplan besteht aus 10 Hauptkapiteln, jeweils mit einem einführenden Abschnitt zu Beginn des Kapitels, in dem dargestellt wird, wie die jeweiligen Inhalte (Module) für die verschiedenen Testrollen relevant sind.

Die Abschnitte 0.3 bis 0.6 enthalten kapitelweise für jedes Modul die Lernziele für Trainingszwecke. In diesen Abschnitten ist auch angegeben, wie viel Unterrichtszeit für ein Thema mindestens aufzuwenden ist.

Es wird dringend empfohlen, mit einem Kapitel des Lehrplans parallel die jeweiligen Lernziele zu lesen. Diese Vorgehensweise ermöglicht es, die Anforderungen in einem Kapitel zu verstehen, und die wesentlichen Inhalte des jeweiligen Kapitels für jedes der drei Module zu erkennen.

## *Begriffe und Definitionen*

Viele Begriffe in der Software-Fachliteratur sind austauschbar. Die gültigen Definitionen für diesen Advanced Level Lehrplan sind im Standard-Glossar der Testbegriffe festgeschrieben, das vom ISTQB® veröffentlicht wurde.

## *Testansätze*

Es gibt verschiedene Ansätze für das Testen von Software, die auf vorliegenden Spezifikationen, Programmstrukturen, Daten, Risiken, Prozessen, Normen und Standards und Taxonomien basieren. Unterschiedliche Prozesse und Werkzeuge unterstützen die Testprozesse; darüber hinaus existieren Methoden für die Verbesserung bestehender Prozesse.

Der vorliegende Advanced Level Lehrplan ist gemäß den in ISO 9126 vorgegebenen Ansätzen aufgebaut, mit einer klaren Trennung zwischen funktionalen, nicht-funktionalen und unterstützenden Ansätzen. Er nennt unterstützende Prozesse und einige Verbesserungsmethoden. Organisation und Prozesse wurden nach freiem Ermessen ausgewählt und sollen Testern und Testmanagern eine gute Grundlage liefern.

# 0.2 Erwartungen

Prüfung und Zertifizierung in der Aufbaustufe (Advanced Level) sind im vorliegenden Lehrplan nach einer Gliederung in drei Hauptaufgabenbereiche beschrieben. Jede Aufgabenbeschreibung steht für bestimmte grundlegende Zuständigkeiten und Erwartungen in einer Organisation. Die Zuständigkeiten und die damit verbundenen Aufgaben können in einer Organisation auf mehrere Personen verteilt sein oder alle von einer einzelnen Person wahrgenommen werden. Die folgenden Abschnitte skizzieren diese Zuständigkeiten.

### *0.2.1 Testmanager Advanced Level*

Professionelle Testmanager des Advanced Level sollten

- Übergeordnete Testziele und –strategien für die zu testenden Systeme festlegen können;
- Aufgaben planen, Termine dafür festlegen und ihren Fortschritt verfolgen können;
- die notwendigen Aktivitäten beschreiben und organisieren können;
- ausreichende Ressourcen für die Aufgaben auswählen, beschaffen und zuordnen können:
- Mitglieder eines Testteams auswählen, Testteams organisieren und leiten können;
- die Kommunikation zwischen den Mitgliedern der Testteams untereinander sowie zwischen den Testteams und allen anderen Betroffenen organisieren können; und

- getroffene Entscheidungen begründen und, gegebenenfalls, geeignete Berichtsunterlagen erstellen können.

### *0.2.2 Test Analyst Advanced Level*

Test Analysts des Advanced Level sollten

- die in der Teststrategie definierten Aufgaben nach den Anforderungen des Geschäftsbereichs strukturieren können;
- das System detailliert genug analysieren können, um die Erwartungen der Anwender an die Qualität zu erfüllen;
- die Systemanforderungen bewerten und damit die Gültigkeit für einer Fachdomäne überprüfen können;
- geeignete Maßnahmen vorbereiten und durchführen, sowie über ihren Fortschritt berichten können;
- die notwendigen Nachweise für Auswertungen bereitstellen können und
- die notwendigen Werkzeuge und Techniken zum Erreichen der definierten Testziele implementieren können.

### *0.2.3 Technical Test Analyst Advanced Level*

Technical Test Analysts des Advanced Level sollten

- die in der Teststrategie definierten Aufgaben hinsichtlich der technischen Anforderungen strukturieren können;
- die systeminterne Struktur detailliert genug analysieren können, um Testfälle zu entwerfen, die die erwartete Qualität nachweisen können;
- das System hinsichtlich seiner technischen Qualitätsmerkmale, wie Leistung, Sicherheit, usw. bewerten können;
- geeignete Maßnahmen vorbereiten und durchführen, sowie über ihren Fortschritt berichten können;
- technische Testaktivitäten durchführen können;
- die notwendigen Nachweise für Auswertungen bereitstellen können und
- die notwendigen Werkzeuge und Techniken zum Erreichen der definierten Testziele implementieren können.

## 0.3 Lernziele/Kognitive Ebenen des Wissens

Die nachfolgend definierten Lernziele bilden die Grundlage des Lehrplans. Jedes Thema des Lehrplans wird anhand des zugeordneten Lernziels geprüft.

**Kognitive Ebene 1: Kennen (K1)**

Der oder die Lernende kann Begriffe oder Konzepte erkennen, sich an sie erinnern und sie wiedergeben.

Schlüsselbegriffe: erinnern, erkennen, kennen, wiedergeben.
Beispiel:
Sie können die Definition von Fehlerwirkung erkennen als

- „das Nichterbringen einer definierten Leistung gegenüber einem Anwender oder anderen Betroffenen", oder
- „die tatsächliche Abweichung einer Komponente oder eines Systems von der erwarteten oder vereinbarten Auslieferung, Dienstleistung oder vom erwarteten Ergebnis".

**Kognitive Ebene 2: Verstehen (K2)**

Der oder die Lernende kann Aussagen zu einem Thema begründen und erklären. Er kann Sachverhalte, Testkonzepte und Testvorgehen zusammenfassen, unterscheiden, klassifizieren und anhand von Beispielen erläutern. Für Sachverhalte kann er beispielsweise Fachbegriffe vergleichen und für Testvorgehen den Ablauf erklären.
Schlüsselbegriffe: darstellen, erläutern anhand von Beispielen, gegenüberstellen, interpretieren, kategorisieren, klassifizieren, schlussfolgern, übertragen, vergleichen, zuordnen, zusammenfassen
Beispiele:
Erklären Sie, weshalb Tests so früh wie möglich entworfen werden sollten:

- Um Fehler zu finden, solange deren Behebung noch kostengünstiger ist.
- Um die wichtigsten Fehler zuerst zu finden.

Erklären Sie Gemeinsamkeiten und Unterschiede von Integrations- und Systemtests:

- Gemeinsamkeiten: Es wird mehr als eine Komponente getestet, nicht-funktionale Aspekte können getestet werden.
- Unterschiede: Integrationstests konzentrieren sich auf Schnittstellen und Interaktionen; Systemtests auf die Aspekte des Gesamtsystems, wie End-to-End-Abläufe.

**Kognitive Ebene 3: Anwenden (K3)**

Der oder die Lernende kann die korrekte Anwendung eines Testkonzepts oder eines Testverfahrens auswählen und auf einen gegebenen Kontext anwenden. K3 bezieht sich normalerweise auf prozedurales Wissen. Kreative Aufgaben, wie die Bewertung einer Softwareanwendung oder das Erstellen eines Modells für eine bestimmte Software, gehören nicht dazu. Wenn ein gegebenes Modell vorliegt, und im Lehrplan das schrittweise Vorgehen zur Erstellung eines Testfalls vom Modell abgedeckt wird, dann gehört es zu K3.

Schlüsselbegriffe: anwenden, durchführen, implementieren, eine Vorgehensweise befolgen, ein Verfahren ausführen
Beispiele:

- Identifizieren Sie die Grenzwerte für gültige und ungültige Äquivalenzklassen.
- Befolgen Sie die allgemeine Vorgehensweise für das Erstellen von Testfällen, und wählen Sie aus einem vorgegebenen Zustandsübergangsdiagramm (und Testfällen) die notwendigen Testfälle aus, um alle Statusübergänge abzudecken.

**Kognitive Ebene 4: Analysieren (K4)**

Der oder die Lernende kann Informationen zu bestimmten Testszenarien oder Testverfahren zum besseren Verständnis in ihre Bestandteile zerlegen und zwischen Sachverhalten und abgeleiteten Schlussfolgerungen unterscheiden. Typische Anwendungen sind die Analyse eines Dokuments, einer Software oder Projektsituation und der Vorschlag angemessener Maßnahmen zur Problemlösung.[1]

Schlüsselbegriffe: analysieren, auswählen, beurteilen, bewerten, entwerfen, erstellen, fokussieren, generieren, Hypothesen aufstellen, konstruieren, koordinieren, planen, produzieren, strukturieren, synthetisieren, überwachen, unterscheiden, zerlegen, zurückführen.
Beispiele:

- Analysieren Sie Produktrisiken und schlagen Sie vorbeugende oder korrigierende Maßnahmen vor.
- Beschreiben Sie, welche Teile eines Abweichungsberichts einen Sachverhalt darstellen und bei welchen es sich um Schlussfolgerungen aus den Ergebnissen handelt.

**Literatur**

(zum Thema Kognitive Ebenen von Lernzielen)

Bloom, B. S. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain,* David McKay, Co. Inc.
Anderson, L. W. and Krathwohl, D. R. (Hg.) (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Allyn & Bacon.

## 0.4 Lernziele für Testmanager

Dieser Abschnitt enthält die detaillierten Lernziele für das Testmanager-Modul.

Für alle Teile dieses Lehrplans gilt, dass sie auf der Wissensstufe K1 geprüft werden können. Dies bedeutet, dass der oder die Lernende den entsprechenden Begriff oder das Konzept erkennt, sich daran erinnert und es wiedergeben kann.

Die folgende Auflistung enthält deshalb nur die Lernziele der Wissensstufen K2, K3 und K4.

**Einführung in den Lehrplan Testmanager – [60 Minuten]**

(einschließlich Wiederholung des Lehrplans zum ISTQB® Foundation Level)

**Kapitel 1: Grundlegende Aspekte des Softwaretestens – [150 Minuten]**

1.2 Testen im Softwarelebenszyklus

LO-1.2.1 (K2) Sie können beschreiben, wie das Testen Bestandteil aller Softwareentwicklungs- und Wartungsaktivitäten ist.
LO-1.2.2 (K4) Sie können Softwarelebenszyklusmodelle analysieren und eine kurze Übersicht über die entsprechenden Aufgaben/Testaktivitäten geben, die durchzuführen sind, und dabei zwischen Test- und Entwicklungsaktivitäten unterscheiden.

[1] Die kognitive Ebene K4 wird hier in einem erweiterten Sinn verwendet und schließt Fähigkeiten des Beurteilens (Evaluate) und Erschaffens (Create) mit ein. Dies im Gegensatz zur angegebenen Literatur, die diese Fähigkeiten gesondert ausweist.


1.3 Spezifische Systeme

LO-1.3.1 (K2) Sie können anhand von Beispielen die spezifischen Aspekte beim Testen von Multisystemen erklären.
LO-1.3.2 (K2) Sie können erklären, weshalb die drei wichtigsten Ergebnisse beim Testen von sicherheitskritischen Systemen die Einhaltung der Vorschriften (Konformität) nachweisen müssen.

1.4 Metriken und Messung

LO-1.4.1 (K2) Sie verstehen testbezogene Standardmetriken und können sie miteinander vergleichen.
LO-1.4.2 (K3) Sie können Testaktivitäten durch Messung des Testobjekts oder der Testobjekte und des Testprozesses überwachen.

**Kapitel 2: Testprozess – [120 Minuten]**

2.3 Testplanung und Teststeuerung

LO-2.3.1 (K2) Sie können anhand von Beispielen erläutern, wie sich Teststrategien auf die Testplanung auswirken.
LO-2.3.2 (K2) Sie können Testarbeitsergebnisse vergleichen und anhand von Beispielen die Zusammenhänge zwischen Arbeitsergebnissen in der Entwicklung und beim Test erläutern.
LO-2.3.3 (K2) Sie können die Aktivitäten zur Teststeuerung klassifizieren, mit denen nachgewiesen werden soll, ob übergeordnete Testzielsetzung, Teststrategien und Testziele erreicht wurden.

2.5 Testrealisierung und Testdurchführung

LO-2.5.1 (K2) Sie können die Voraussetzungen für die Testdurchführung erläutern.
LO-2.5.2 (K2) Sie können anhand von Beispielen die Vor- und Nachteile einer frühzeitigen Testrealisierung erläutern, und dabei unterschiedliche Testmethoden (wie in Kapitel 4 aufgeführt) berücksichtigen.
LO-2.5.3 (K2) Sie können erläutern, warum Nutzer und/oder Kunden an der Durchführung der Tests teilnehmen können.
LO-2.5.4 (K2) Sie können darstellen, wie der Umfang der Testprotokollierung je nach Teststufe variieren kann.

2.6 Testauswertung und -bericht

LO-2.6.1 (K2) Sie können zusammenfassen, welche Informationen im Lauf des Testprozesses für eine korrekte Testberichterstattung und Bewertung der Testendekriterien zu sammeln sind.

2.7 Abschluss der Testaktivitäten

LO-2.7.1 (K2) Sie können die vier Gruppen von Testabschlussaktivitäten zusammenfassen.

LO-2.7.2 (K3) Sie können die in der Testabschlussphase gewonnenen Erkenntnisse verallgemeinern, um die Bereiche für eine Verbesserung oder Wiederholung zu identifizieren.

**Kapitel 3: Testmanagement – [1120 Minuten]**

3.2 Testmanagement-Dokumentation

LO-3.2.1 (K4) Sie können einige Testmanagement-Dokumente erläutern, wie Testkonzept, Testentwurfsspezifikation und Testvorgehen gemäß IEEE Standard 829.
LO-3.2.2 (K2) Sie können mindestens vier wichtige Elemente einer Teststrategie oder eines Testansatzes darstellen und angeben, welche Dokumente gemäß IEEE Standard 829 Teststrategieelemente enthalten.
LO-3.2.3 (K2) Sie können darstellen, wie und warum Abweichungen von der Teststrategie in den anderen Testmanagement-Dokumenten behandelt werden.

3.3 Dokumentvorlagen für Testkonzepte

LO-3.3.1 (K2) Sie können den Aufbau eines Mastertestkonzepts gemäß IEEE Standard 829 zusammenfassen.
LO-3.3.2 (K2) Sie können die Themen eines gemäß IEEE Standard 829 aufgebauten Testkonzepts umschreiben und interpretieren. Dabei können Sie eingehen auf die Anpassung des Testkonzepts an ein Unternehmen/eine Organisation, auf das Produktrisiko sowie auf Risiko, Größe und Formalität eines Projekts.

3.4 Testschätzung

LO-3.4.1 (K3) Sie können den Testaufwand für ein kleines Beispielsystem unter Anwendung einer metrikbasierten und einer erfahrungsbasierten Vorgehensweise schätzen; dabei können Sie die Einflussfaktoren aus Kosten, Aufwand und Zeitdauer berücksichtigen.
LO-3.4.2 (K2) Sie können anhand von Beispielen die im Lehrplan aufgeführten Faktoren erläutern, die zu ungenauen Testschätzungen führen können.

3.5 Zeitliche Testplanung

LO-3.5.1 (K2) Sie können anhand von Beispielen die Vorteile einer frühzeitigen und iterativen Erstellung eines Testkonzepts erläutern.

3.6 Testfortschritt überwachen und steuern

LO-3.6.1 (K2) Sie können die verschiedenen Verfahren zur Steuerung des Testfortschritts vergleichen.
LO-3.6.2 (K2) Sie können mindestens fünf konzeptionell unterschiedliche Beispiele nennen, wie die Ergebnisse der Testfortschrittsüberwachung den Verlauf des Testprozesses beeinflussen.
LO-3.6.3 (K4) Sie können Ergebnisse aus der Überwachung und Steuerung des Testfortschritts dazu verwenden, Maßnahmen und einen Aktionsplan auszuarbeiten, mit denen der aktuelle Testprozess verbessert werden kann. Sie können Verbesserungen vorschlagen.

LO-3.6.4 (K4) Sie können Testergebnisse analysieren und den Testfortschritt beurteilen, wie er mit allen Berichtsdimensionen in Testfortschrittsbericht und Testabschlussbericht dokumentiert ist.

## 3.7 Geschäftswert des Testens

LO-3.7.1 (K2) Sie können Beispiele (Maßnahmen) für alle vier Kategorien nennen, die die Qualitätskosten bestimmen.
LO-3.7.2 (K3) Sie können die relevanten quantitativen und/oder qualitativen Werte für einen gegebenen Kontext nennen.

## 3.8 Verteiltes Testen, Outsourcing und Insourcing

LO-3.8.1 (K2) Sie können Risiken, Gemeinsamkeiten und Unterschiede der drei Personalorganisations-Modelle (verteiltes Testen, Outsourcing, Insourcing) nennen.

## 3.9 Risikoorientiertes Testen

### 3.9.1 Einführung in das risikoorientierte Testen

LO-3.9.1.1 (K2) Sie können anhand von Beispielen erläutern, auf welche unterschiedliche Art und Weise risikoorientiertes Testen auf Risiken reagiert.
LO-3.9.1.2 (K4) Sie können die Risiken eines Projekts und eines Produkts identifizieren und eine geeignete Teststrategie und ein Testkonzept für diese Risiken bestimmen.

### 3.9.2 Risikomanagement

LO-3.9.2.1 (K3) Sie können eine Risikoanalyse für ein Produkt aus Sicht der Tester durchführen und dabei die FMEA-Vorgehensweise befolgen.
LO-3.9.2.2 (K4) Sie können Ergebnisse aus der Sicht verschiedener wichtiger Betroffener zusammenfassen. Sie können deren kollektive Beurteilung dazu nutzen, geeignete Testaktivitäten zur Risikobeherrschung zu skizzieren.

### 3.9.3 Risikomanagement im Softwarelebenszyklus

LO-3.9.3.1 (K2) Sie können die Merkmale des Risikomanagements darstellen, die ursächlich dafür sind, dass Risikomanagement ein iterativer Prozess ist.
LO-3.9.3.2 (K3) Sie können eine gegebene risikoorientierte Teststrategie in konkrete Testaktivitäten umsetzen und deren Auswirkungen beim Testen überwachen.
LO-3.9.3.3 (K4) Sie können die Testergebnisse analysieren und dokumentieren und Restrisiken bestimmen oder benennen, um so dem Projektmanagement intelligente Release-Entscheidungen zu ermöglichen.

## 3.10 FMEA (Fehler-Möglichkeits- und Einfluss-Analyse)

LO-3.10.1 (K2) Sie können das Konzept der FMEA beschreiben und anhand von Beispielen ihre Anwendung in Projekten und den Nutzen für die Projekte erläutern.


3.11 Besonderheiten beim Testmanagement

LO-3.11.1 (K2) Sie können die Besonderheiten des Testmanagements beim explorativen Testen, beim Test von Multisystemen und beim Test sicherheitskritischer Systeme vergleichen bezüglich Teststrategie, Vor- und Nachteilen und Angemessenheit und bezüglich der Auswirkungen auf Testplanung, Überdeckung, Überwachung und Steuerung.

**Kapitel 4: Testverfahren – [0 Minuten]**
Kein Lernziel dieses Kapitels (auf keiner der kognitiven Ebenen) betrifft die Rolle „Testmanager".

**Kapitel 5: Test der Softwareeigenschaften – [0 Minuten]**
Kein Lernziel dieses Kapitels (auf keiner der kognitiven Ebenen) betrifft die Rolle „Testmanager".

**Kapitel 6: Review – [120 Minuten]**

6.2 Grundsätze von Reviews

LO-6.2.1 (K2) Sie können die Vorteile von Reviews im Vergleich zu dynamischen und weiteren statischen Testverfahren erläutern.

6.4 Einführung von Reviews

LO-6.4.1 (K2) Sie können Review-Arten miteinander vergleichen und deren relative Stärken, Schwächen und Einsatzbereiche aufzeigen.

LO-6.4.2 (K3) Sie können ein Review-Team durch ein formales Review mit den notwendigen Schritten führen.

LO-6.4.3 (K4) Sie können einen Review-Plan als Teil des Qualitäts-/Testkonzepts eines Projekts skizzieren, der die Review-Techniken unter Berücksichtigung der aufzudeckenden Fehler und der Fähigkeiten des verfügbaren Personals einsetzt und auf die geeigneten dynamischen Testansätze abstimmt.

6.5 Erfolgsfaktoren für Reviews

LO-6.5.1 (K2) Sie können die möglichen Risiken erläutern, wenn technische, organisatorische und personenbezogene Faktoren beim Durchführen von Reviews nicht berücksichtigt werden.

**Kapitel 7: Fehler- und Abweichungsmanagement – [80 Minuten]**

LO-7.1 (K3) Sie können einen Abweichung nach dem in IEEE Standard 1044-1993 festgelegten Abweichungsmanagement-Lebenszyklusmodell bearbeiten.

LO-7.2 (K3) Sie können Fehlerberichte auf Basis des IEEE Standard 1044-1993 und der angewendeten Fehlertaxonomie bewerten, um deren Qualität zu verbessern.

LO-7.3 (K4) Sie können die im Lauf der Zeit erstellten Fehlerberichte analysieren und die Fehlertaxonomie aktualisieren.

**Kapitel 8: Standards im Testverbesserungs-Prozess – [120 Minuten]**

LO-8.1 (K2) Sie können Quellen der Softwarestandards zusammenfassen und Sie können die Nützlichkeit der Softwarestandards für das Softwaretesten erklären.

8.4 Testverbesserungs-Prozess

LO-8.4.1 (K3) Sie können einen Testverbesserungsplan erstellen und testen und dabei die allgemeinen Prozessschritte einsetzen und die richtigen Personen beteiligen.
LO-8.4.2 (K2) Sie können die in den Modellen TMM, TPI, CTP, STEP definierten Testverbesserungs-Prozesse zusammenfassen sowie die Prozessbereiche Verifizierung und Validierung nach CMMI.
LO-8.4.3 (K2) Sie können die Bewertungskriterien der Testverbesserungs-Modelle TMM, TPI, CTP, STEP und die Prozessbereiche Verifizierung und Validierung nach CMMI erklären.

**Kapitel 9: Testwerkzeuge und Automatisierung – [90 Minuten]**

9.2 Testwerkzeugkonzepte

LO-9.2.1 (K2) Sie können die Grundgedanken und Wesensmerkmale in jedem der folgenden Überlegungen zu Testwerkzeugen miteinander vergleichen: „Kosten, Nutzen und Risiken von Testwerkzeugen und Automatisierung“, „Testwerkzeugstrategien“, „Integration und Informationsaustausch“, „Automatisierungssprachen“, „Testorakel“, „Testwerkzeuge in Betrieb nehmen“, „Open Source-Werkzeuge“, „Eigene Testwerkzeuge entwickeln“ sowie „Testwerkzeuge klassifizieren“.
LO-9.2.2 (K2) Sie können beschreiben, warum und wann es wichtig ist, eine Strategie für den Einsatz von Testwerkzeugen zu erstellen.
LO-9.2.3 (K2) Sie verstehen die verschiedenen Phasen der Testwerkzeug-Implementierung.

9.3 Testwerkzeugkategorien

LO-9.3.1 (K2) Sie können die Testwerkzeugkategorien nach Zielsetzung, Verwendungszweck, Stärken, Risiken und mit Beispielen zusammenfassen.
LO-9.3.2 (K2) Sie können die spezifischen Anforderungen an Testwerkzeuge und Open Source-Testwerkzeuge zusammenfassen, die beim Testen sicherheitskritischer Systeme eingesetzt werden.
LO-9.3.3 (K2) Sie können wichtige Aspekte und Konsequenzen unterschiedlicher Testwerkzeuge und deren Implementierung, Verwendung und ihre Auswirkung auf den Testprozess beschreiben.
LO-9.3.4 (K2) Sie können beschreiben, wann und warum die Entwicklung eines eigenen Werkzeugs in Frage kommt, sowie die damit verbundenen Vorteile, Risiken und Folgen.

**Kapitel 10: Soziale Kompetenz und Teamzusammensetzung – [240 Minuten]**

10.2 Individuelle Fähigkeiten

LO-10.2.1 (K3) Sie können einen vorgegebenen Fragebogen anwenden, um Stärken und Schwächen von Teammitgliedern festzustellen bezogen auf die Anwendung von Softwaresystemen, Kenntnissen des Geschäftsbereichs/der Branche, den Bereich der Systementwicklung, des Softwaretestens und den zwischenmenschlichen Fähigkeiten.

10.3 Dynamik im Testteam

LO-10.3.1 (K3) Sie können eine Soll/Ist-Analyse durchführen, um die benötigten technischen Fähigkeiten oder erforderliche soziale Kompetenz für die offenen Positionen einer Organisation zu bestimmen.

10.4 Testen in der Organisationsstruktur etablieren

LO-10.4.1 (K2) Sie können die verschiedenen organisatorischen Optionen bzgl. der Unabhängigkeit des Testens charakterisieren und mit Insourcing, Outsourcing und Off-Shoring vergleichen.

10.5 Motivieren

LO-10.5.1 (K2) Sie können Beispiele anführen für die Tester motivierende und demotivierende Faktoren.

10.6 Kommunizieren

LO-10.6.1 (K2) Sie können anhand von Beispielen eine professionelle, objektive und effektive Kommunikation in einem Projekt aus Sicht des Testers beschreiben. Stellen Sie dabei auch Risiken und Chancen dar.

## 0.5 Lernziele für Test Analysts

Dieser Abschnitt enthält die detaillierten Lernziele für das Modul Test Analyst.

Für alle Teile dieses Lehrplans gilt, dass sie auf der Wissensstufe K1 geprüft werden können. Dies bedeutet, dass der oder die Lernende den entsprechenden Begriff oder das Konzept erkennt, sich daran erinnert und es wiedergeben kann.

Die folgende Auflistung enthält deshalb nur die Lernziele der Wissensstufen K2, K3 und K4.

**Einführung in den Lehrplan Test Analyst – [60 Minuten]**
(einschließlich Wiederholung des Lehrplans zum ISTQB® Foundation Level)

**Kapitel 1: Grundlegende Aspekte des Softwaretestens – [30 Minuten]**

**Kapitel 2: Testprozesse – [180 Minuten]**

2.4 Testanalyse und Testentwurf

LO-2.4.1 (K2) Sie können erklären, warum funktionale Tests in bestimmten Lebenszyklusphasen einer Anwendung stattfinden.
LO-2.4.2 (K2) Sie können anhand von Beispielen die Kriterien erläutern, die bei der Entwicklung von Testbedingungen Struktur und Tiefe beeinflussen.
LO-2.4.3 (K2) Sie können beschreiben, inwiefern Testanalyse und Testentwurf statische Testtechniken sind, die zur Aufdeckung von Fehlern eingesetzt werden können.
LO-2.4.4 (K2) Sie können anhand von Beispielen das Konzept der Testorakel erläutern und wie ein Testorakel in Testspezifikationen eingesetzt werden kann.

2.5 Testrealisierung und Testdurchführung

LO-2.5.5 (K2) Sie können die Voraussetzungen für die Testdurchführung beschreiben, einschließlich Testmitteln, Testumgebung, Konfigurationsmanagement und Abweichungsmanagement.

2.6 Testendekriterien auswerten, Bericht

LO-2.6.2 (K3) Sie können mit vorgegebenen Metriken bewerten, ob ein bestimmtes Testendekriterium erfüllt ist.

**Kapitel 3: Testmanagement – [120 Minuten]**

3.9.2 Risikomanagement

LO-3.9.2.3 (K3) Sie können die Auswahl von Testfällen, Testüberdeckung und Testdaten auf Basis des Risikos priorisieren und das angemessen in einem Testkonzept und einer Testspezifikation dokumentieren.
LO-3.9.2.4 (K2) Sie können die Aktivitäten bei einem risikoorientierten Testansatz für Planung und Durchführung von fachlichen Tests umreißen.

**Kapitel 4: Testverfahren – [1080 Minuten]**

4.2 Spezifikationsorientierten Testverfahren

LO-4.2.1 (K2) Sie können Beispiele für typische Fehlerzustände anführen, die sich mit den jeweiligen spezifikationsorientierten Testverfahren identifizieren lassen, und den entsprechenden Überdeckungsgrad angeben.

LO-4.2.2 (K3) Sie können Testfälle aus vorgegebenen Softwaremodellen erstellen und dabei folgende Testentwurfsverfahren anwenden. Die Tests sollen dabei eine gegebene Modellüberdeckung erzielen:

- Äquivalenzklassenbildung
- Grenzwertanalyse
- Entscheidungstabellentest
- Zustandsbasierter Test
- Klassifikationsbaumverfahren
- Paarweises Testen
- Anwendungsfallbasiertes Testen

LO-4.2.3 (K4) Sie können ein System oder dessen Anforderungsspezifikation analysieren und festlegen, welche spezifikationsorientierten Testverfahren für bestimmte Zielsetzungen anzuwenden sind, und Sie können eine Testspezifikation nach IEEE Standard 829 skizzieren, mit dem Schwerpunkt auf funktionalen und fachlichen Testfällen und Testverfahren.

4.4 Fehlerbasierte und erfahrungsbasierte Testverfahren

LO-4.4.1 (K2) Sie können Prinzip und Gründe für das fehlerbasierte Testentwurfsverfahren beschreiben und gegen den Einsatz spezifikationsorientierter und strukturorientierter Testverfahren abgrenzen.

LO-4.4.2 (K2) Sie können anhand von Beispielen Fehlertaxonomien und deren Einsatz erläutern.

LO-4.4.3 (K2) Sie können das Prinzip der erfahrungsbasierten Testentwurfsverfahren und die Gründe für ihren Einsatz verstehen und wann sie genutzt werden.

LO-4.4.4 (K3) Sie können Tests nach dem explorativen Testentwurfsverfahren spezifizieren, durchführen und darüber berichten.

LO-4.4.5 (K2) Sie können Fehlerzustände gemäß den Zielen verschiedener Fehlerangriffe klassifizieren.

LO-4.4.6 (K4) Sie können ein System analysieren, um festzulegen, welche spezifikationsorientierten, fehlerbasierten oder erfahrungsbasierten Testverfahren für bestimmte Ziele einzusetzen sind.

**Kapitel 5: Test der Softwareeigenschaften – [210 Minuten]**

5.2 Qualitätsmerkmale bei fachlichen Tests

LO-5.2.1 (K4) Sie können anhand von Beispielen erläutern, welche der in Kapitel 4 erwähnten Testverfahren geeignet sind, um Richtigkeit, Angemessenheit, Interoperabilität, funktionale Sicherheit und Zugänglichkeit eines Systems zu testen.

LO-5.2.2 (K3) Sie können Benutzbarkeitstests skizzieren, entwerfen, spezifizieren und durchführen. Sie können dabei die geeigneten Verfahren einsetzen und vorgegebene Testziele und zu findende Fehlerzustände berücksichtigen.

5.3 Qualitätsmerkmale beim technischen Testen

LO-5.3.1 (K2) Sie können erklären, warum Effizienztests, Zuverlässigkeitstests und technische Sicherheitstests Teil einer Teststrategie sein sollten, und Beispiele für Fehlerzustände anführen, die damit gefunden werden sollen.

LO-5.3.2 (K2) Sie können nicht-funktionale Testarten für das technische Testen anhand typischer Fehlerzustände charakterisieren, die gezielt angegriffen werden (Fehlerangriffe) die typische Anwendung im Lebenszyklus einer Anwendung, und die für das Testdesign geeigneten Testverfahren.

**Kapitel 6: Review – [180 Minuten]**

6.5 Erfolgsfaktoren für Reviews

LO-6.5.1 (K3) Sie können eine Review-Checkliste verwenden, um Programmcode und Architektur aus Sicht eines Testers zu verifizieren.

LO-6.5.2 (K3) Sie können eine Review-Checkliste verwenden, um Anforderungen und Anwendungsfälle aus Sicht des Testers zu verifizieren.

LO-6.5.3 (K2) Sie können Review-Arten miteinander vergleichen und deren relative Stärken, Schwächen und Einsatzbereiche aufzeigen.

**Kapitel 7: Fehler- und Abweichungsmanagement – [120 Minuten]**

7.4 Pflichtfelder für die Erfassung von Fehler- und Abweichungen

LO-7.4.1 (K4) Sie können funktionale und nicht-funktionale Fehlerwirkungen analysieren, klassifizieren, in verständlichen Abweichungsberichten und beschreiben.

**Kapitel 8: Standards im Testverbesserungs-Prozess – [0 Minuten]**
Kein Lernziel dieses Kapitel (auf keiner der kognitiven Ebenen) betrifft die Test Analysts.

**Kapitel 9: Testwerkzeuge und Automatisierung – [90 Minuten]**

9.2 Testwerkzeugkonzepte

LO-9.2.1 (K2) Sie können die Elemente und Aspekte in jedem der folgenden Testwerkzeugkonzepte vergleichen: „Kosten, Nutzen und Risiken von Testwerkzeugen und Automatisierung“, „Testwerkzeugstrategien“, „Integration und Informationsaustausch“, „Automatisierungssprachen“, „Testorakel“, „Testwerkzeuge in Betrieb nehmen“, „Open Source-Werkzeuge“, „Eigene Testwerkzeuge entwickeln“ sowie „Testwerkzeuge klassifizieren“.

9.3 Testwerkzeugkategorien

LO-9.3.1 (K2) Sie können die Testwerkzeugkategorien nach Zielsetzung, Verwendungszweck, Stärken und Risiken zusammenfassen, sowie Beispiele nennen.
LO-9.3.5 (K2) Sie können die Werkzeuge der verschiedenen Werkzeugkategorien den unterschiedlichen Teststufen und Testarten zuordnen.

**Kapitel 10: Soziale Kompetenz und Teamzusammensetzung – [30 Minuten]**

10.6 Kommunizieren

LO-10.6.1 (K2) Sie können anhand von Beispielen eine professionelle, objektive und effektive Kommunikation in einem Projekt aus Sicht des Testers beschreiben Stellen Sie dabei Risiken und Chancen dar.

## 0.6 Lernziele für Technical Test Analysts

Dieser Abschnitt enthält die detaillierten Lernziele für das Modul Technical Test Analyst.

Für alle Teile dieses Lehrplans gilt, dass sie auf der Wissensstufe K1 geprüft werden können. Dies bedeutet, dass der oder die Lernende den entsprechenden Begriff oder das Konzept erkennt, sich daran erinnert und es wiedergeben kann.

Die folgende Auflistung enthält deshalb nur die Lernziele der Wissensstufen K2, K3 und K4.

**Einführung in den Lehrplan Technical Test Analyst – [60 Minuten]**
(einschließlich Wiederholung des Lehrplans zum ISTQB® Foundation Level)

**Kapitel 1: Grundlegende Aspekte des Softwaretestens – [30 Minuten]**

**Kapitel 2: Der Testprozess – [180 Minuten]**

2.4 Testanalyse und Testentwurf

LO-2.4.1 (K2) Sie können die Stufen im Softwarelebenszyklus einer Anwendung erklären, in denen sich nicht-funktionale Tests und architekturorientierte Tests einsetzen lassen. Sie können erklären, warum nicht-funktionale Tests nur in bestimmten Stufen des Lebenszyklus stattfinden.

LO-2.4.2 (K2) Sie können anhand von Beispielen die Kriterien erläutern, die bei der Entwicklung von Testbedingungen Struktur und Tiefe beeinflussen.

LO-2.4.3 (K2) Sie können beschreiben, inwiefern Testanalyse und Testentwurf statische Testtechniken sind, die zur Aufdeckung von Fehlern eingesetzt werden können.

LO-2.4.4 (K2) Sie können anhand von Beispielen das Konzept der Testorakel erläutern, und wie ein Testorakel in Testspezifikationen eingesetzt werden kann.

2.5 Testrealisierung und Testdurchführung

LO-2.5.5 (K2) Sie können die Voraussetzungen für die Testdurchführung beschreiben, einschließlich Testmittel, Testumgebung, Konfigurationsmanagement und Abweichungsmanagement.

2.6 Testendekriterien auswerten, Bericht

LO-2.6.2 (K3) Sie können mit vorgegebenen Metriken bewerten, ob ein bestimmtes Testendekriterium erfüllt ist.

**Kapitel 3: Testmanagement – [120 Minuten]**

3.9.2 Risikomanagement

LO-3.9.2.5 (K2) Sie können die Aktivitäten bei einem risikoorientierten Testansatz für Planung und Durchführung von fachlichen Tests umreißen.

**Kapitel 4: Testverfahren – [930 Minuten]**

4.2 Spezifikationsoriente Verfahren

LO-4.2.1 (K2) Sie können Beispiele für typische Fehlerzustände anführen, die sich mit den jeweiligen spezifikationsorientierten Testverfahren identifizieren lassen.

LO-4.2.2 (K3) Sie können Testfälle aus vorgegebenen Softwaremodellen erstellen und dabei folgende Testentwurfsverfahren anwenden, wobei die Tests eine gegebene Modellüberdeckung erzielen sollen:

- Äquivalenzklassenbildung
- Grenzwertanalyse
- Entscheidungstabellentest
- Zustandsbasierter Test

LO-4.2.3 (K4) Sie können ein System oder dessen Anforderungsspezifikation analysieren und festlegen, welche spezifikationsorienten Testverfahren für bestimmte Zielsetzungen anzuwenden sind, und Sie können eine Testspezifikation nach IEEE Standard 829, mit dem Schwerpunkt auf Testfällen für Komponententests und nicht-funktionale Tests und Testverfahren skizzieren.

4.3 Strukturorientierte Testverfahren

LO-4.3.1 (K2) Sie können Beispiele für typische Fehler anführen, die sich mit den jeweiligen strukturorientierten Testverfahren identifizieren lassen.

LO-4.3.2 (K3) Sie können mit den folgenden Testentwurfsverfahren echte Testfälle entwerfen, wobei die Tests eine vorgegebene Modellüberdeckung erzielen sollen:

- Anweisungsüberdeckungstest
- Entscheidungsüberdeckungstest
- Definierter Bedingungsüberdeckungstest
- Mehrfachbedingungsüberdeckungstest

LO-4.3.3 (K4) Sie können ein System analysieren, um festzulegen, welches strukturorientierte Testverfahren für bestimmte Zielsetzungen anzuwenden ist.

LO-4.3.4 (K2) Sie können jedes der strukturorientierten Testentwurfsverfahren und die zugehörigen Überdeckungsgrade verstehen sowie, wann welches Verfahren verwendet wird.

LO-4.3.5 (K4) Sie können vergleichen und analysieren, welches strukturorientierte Testentwurfsverfahren in unterschiedlichen Situationen einzusetzen ist.

4.4 Fehlerbasierte und erfahrungsbasierte Testverfahren

LO-4.4.1 (K2) Sie können Prinzip und Gründe für das fehlerbasierte Testentwurfsverfahren beschreiben und gegen den Einsatz spezifikationsorientierter und strukturorientierter Testverfahren abgrenzen.

LO-4.4.2 (K2) Sie können anhand von Beispielen Fehlertaxonomien und deren Einsatz erläutern.

LO-4.4.3 (K2) Sie können Prinzip und Gründe für den Einsatz erfahrungsbasierter Testentwurfsverfahren verstehen und wann sie genutzt werden.

LO-4.4.4 (K3) Sie können Tests unter Nutzung explorativen Testens spezifizieren, durchführen und darüber berichten.

LO-4.4.6 (K2) Sie können Tests nach den Zielen verschiedener Fehlerangriffe spezifizieren.

LO-4.4.7 (K4) Sie können ein System analysieren, um festzulegen, welche spezifikationsorientierten, fehlerbasierten oder erfahrungsbasierten Testverfahren für bestimmte Ziele anzuwenden sind.

4.5 Statische Analyse

LO-4.5.1 (K3) Sie können die Algorithmen „Kontrollflussanalyse" und „Datenflussanalyse" anwenden, um zu verifizieren, dass der Programmcode keine Kontroll- oder Datenflussanomalie hat.
LO-4.5.2 (K4) Sie können die Ergebnisse der Kontrollfluss- und Datenflussanalyse interpretieren, die ein Werkzeug liefert, und bewerten, ob eine Kontroll- oder Datenflussanomalie vorliegt.
LO-4.5.3 (K2) Sie können den Einsatz von Aufrufgraphen zur Bewertung der Architekturqualität erklären. Dazu gehören die zu identifizierenden Fehlerzustände, die Anwendung für Testentwurf und Testplanung und die möglicherweise eingeschränkte Gültigkeit der Ergebnisse.

4.6 Dynamische Analyse

LO-4.6.1 (K2) Sie können erklären, wie eine dynamische Analyse des Programmcodes durchgeführt wird, und zusammenfassen, welche Fehlerzustände mit diesem Verfahren aufgedeckt werden können und wo die Grenzen dieses Verfahrens liegen.

**Kapitel 5: Test der Softwareeigenschaften – [240 Minuten]**

5.3 Qualitätsmerkmale beim technischen Testen

LO-5.3.2 (K2) Sie können nicht-funktionale Testarten für das technische Testen anhand typischer Fehlerzustände charakterisieren, die gezielt angegriffen werden (Fehlerangriffe), die typische Anwendung im Lebenszyklus einer Anwendung, und die für das Testdesign geeigneten Testverfahren.
LO-5.3.3 (K2) Sie können verstehen und erklären, in welchen Lebenszyklusphasen einer Software oder eines Systems Sicherheits-, Zuverlässigkeits- und Effizienztests angewendet werden (einschließlich der entsprechenden Teilmerkmale nach ISO 9126)
LO-5.3.4 (K2) Sie können die Arten von Fehlern unterscheiden, die durch Sicherheits-, Zuverlässigkeits- und Effizienztests aufgedeckt werden (einschließlich der entsprechenden Teilmerkmale nach ISO 9126)
LO-5.3.5 (K2) Sie können Testansätze für das Testen auf Sicherheits-, Zuverlässigkeits- und Effizienzmerkmale und deren entsprechende Teilmerkmale nach ISO 9126 charakterisieren.
LO-5.3.6 (K3) Sie können Testfälle für das Testen auf Sicherheits-, Zuverlässigkeits- und Effizienzmerkmale und deren entsprechende Teilmerkmale nach ISO 9126 spezifizieren.
LO-5.3.7 (K2) Sie können verstehen und erläutern, warum Wartbarkeit, Portabilität und Zugänglichkeitstest Teil einer Teststrategie sein sollten.
LO-5.3.8 (K3) Sie können Testfälle für nicht-funktionale Tests auf Änderbarkeit und Portabilität spezifizieren.

**Kapitel 6: Review – [180 Minuten]**

6.4 Einführung von Reviews

LO-6.4.1 (K2) Sie können Review-Arten miteinander vergleichen und deren relative Stärken, Schwächen und Einsatzbereiche aufzeigen.

6.5 Erfolgsfaktoren für Reviews

LO-6.5.3 (K4) Sie können eine Review-Checkliste skizzieren, um typische Fehlerzustände aufzudecken, die bei Reviews von Code und Architektur gefunden werden.

**Kapitel 7: Fehler- und Abweichungsmanagement – [120 Minuten]**

7.4 Pflichtfelder für die Erfassung von Fehler- und Abweichungsmeldungen

LO-7.4.2 (K4) Sie können nicht-funktionale Abweichungen in verständlichen Abweichungsberichten analysieren, klassifizieren und beschreiben.

**Kapitel 8: Standards und Testverbesserungs-Prozess – [0 Minuten]**
Kein Lernziel dieses Kapitel (auf keiner der kognitiven Ebenen) betrifft die Technical Test Analysts.

**Kapitel 9: Testwerkzeuge und Automatisierung – [210 Minuten]**

9.2 Testwerkzeugkonzepte

LO-9.2.1 (K2) Sie können die Elemente und Aspekte in jedem der folgenden Testwerkzeugkonzepte vergleichen: „Kosten, Nutzen und Risiken von Testwerkzeugen und Automatisierung“, „Testwerkzeugstrategien“, „Integration und Informationsaustausch“, „Automatisierungssprachen“, „Testorakel“, „Testwerkzeuge in Betrieb nehmen“, „Open Source-Werkzeuge“, „Eigene Testwerkzeuge entwickeln“ sowie „Testwerkzeuge klassifizieren“.

9.3 Testwerkzeugkategorien

LO-9.3.1 (K2) Sie können die Testwerkzeugkategorien nach Zielsetzung, Verwendungszweck, Stärken, Risiken und mit Beispielen zusammenfassen.
LO-9.3.5 (K2) Sie können die Werkzeuge der verschiedenen Werkzeugkategorien den unterschiedlichen Teststufen und Testarten zuordnen.

9.3.7 Schlüsselwortgetriebene Testautomatisierung

LO-9.3.7.1 (K3) Sie können Schlüsselwort-/Aktionswort-Tabellen mit Hilfe des Schlüsselwort-Algorithmus erstellen, den ein Testausführungs-Werkzeug verwenden wird.

9.3.8 Performanztestwerkzeuge

LO-9.3.8.1 (K3) Sie können Performanztests entwerfen und planen, um Systemmerkmale messen zu können.

**Kapitel 10: Soziale Kompetenz und Teamzusammensetzung – [30 Minuten]**

10.6 Kommunizieren

LO-10.6.1 (K2) Sie können anhand von Beispielen eine professionelle, objektive und effektive Kommunikation in einem Projekt aus Sicht des Testers beschreiben. Stellen Sie dabei Risiken und Chancen dar.

# 1. Grundlegende Aspekte des Softwaretestens

*Begriffe:*

ethische Leitlinien, Messung, Metrik, Multisysteme, sicherheitskritische Systeme, Softwarelebenszyklus.

## 1.1 Einführung

Dieses Kapitel enthält zentrale Testthemen, die von allgemeiner Relevanz für alle Mitarbeiter im Bereich Testen sind, d.h. für Testmanager, Test Analysts und Technical Test Analysts. Die Ausbildungsanbieter werden diese allgemeinen Themen im Kontext des jeweiligen Moduls behandeln und durch relevante Beispiele ergänzen. Im Modul Technical Test Analyst gibt es beispielsweise zum allgemeinen Thema Metriken und Messung (Abschnitt 1.4) Beispiele für technische Metriken, wie Performanzmessungen.

Abschnitt 1.2 betrachtet den Testprozess als Teil des gesamten Softwarelebenszyklus. Das Thema baut auf den grundlegenden Konzepten auf, wie sie im Foundation-Level-Lehrplan eingeführt wurden. Es legt besonderen Wert auf das Angleichen des Testprozesses an den Softwarelebenszyklus und andere IT-Prozesse.

Systeme können verschiedene Ausprägungen annehmen, die die Vorgehensweise beim Testen erheblich beeinflussen können. Abschnitt 1.3 stellt zwei spezifische Systemtypen vor, die allen Testern bekannt sein müssen: Multisysteme und sicherheitskritische Systeme.

Fortgeschrittene Tester sind mit einigen schwierigen Aufgaben konfrontiert, wenn sie die in diesem Lehrplan beschriebenen unterschiedlichen Testaspekte in ihren Organisationen, Teams und Aufgabenbereichen einführen.

## 1.2 Testen im Softwarelebenszyklus

Testen ist integraler Bestandteil verschiedener Software-Entwicklungsmodelle:

- sequenziell (Wasserfallmodell, V-Modell und W-Modell)
- iterativ (Rapid Application Development (RAD) und Spiralmodell)
- inkrementell (evolutionäre und agile Methoden)

Das langfristige Einbinden des Testens in den Softwarelebenszyklus sollte als Teil der Teststrategie betrachtet und beschrieben werden. Dazu gehören Organisation, Prozessbeschreibungen sowie Auswahl von Werkzeugen und Methoden.

Testprozesse werden nicht isoliert ausgeführt, sondern stehen in Zusammenhang mit anderen Prozessen und beziehen sich auf diese, wie:

- Anforderungsanalyse und -management
- Projektmanagement
- Konfigurations- und Änderungsmanagement
- Softwareentwicklung
- Softwarewartung
- technischer Support
- Erstellen technischer Dokumentation

In sequenziellen Softwareentwicklungs-Modellen steht die frühzeitige Testplanung in einem engen Zusammenhang mit der späteren Testdurchführung. Die Testaktivitäten können sich zeitlich überlappen und/oder gleichzeitig stattfinden.

Änderungs- und Konfigurationsmanagement sind wichtige unterstützende Aufgaben beim Softwaretesten. Ohne geeignetes Änderungsmanagement lassen sich die Auswirkungen von Änderungen auf das System nicht beurteilen. Ohne Konfigurationsmanagement besteht die Gefahr, dass parallele Entwicklungen verloren gehen oder schlecht verwaltet werden.

Abhängig vom Projektkontext können zusätzlich zu den im Lehrplan definierten Teststufen noch weitere definiert werden, beispielsweise:

- Hardware-Software Integrationstest
- Systemintegrationstest
- Interaktionstest der Funktionen
- Produktintegrationstest beim Kunden

Jede Teststufe lässt sich wie folgt kennzeichnen:

- Testziele
- Testumfang
- Rückverfolgbarkeit zur Testbasis
- Eingangsbedingungen und Testendekriterien
- Test-Arbeitsergebnisse und Berichterstattung
- Testverfahren
- Maße und Metriken
- Testwerkzeuge
- Einhaltung von organisationsinternen oder anderen Standards

Je nach Kontext lassen sich Ziele und Umfang der Teststufen isoliert oder auf Projektebene betrachten (beispielsweise, um unnötiges Wiederholen ähnlicher Tests in unterschiedlichen Teststufen zu vermeiden).

Testaktivitäten müssen an das ausgewählte Softwarelebenszyklusmodell angepasst werden, das sequenziell sein kann (beispielsweise Wasserfall, V-Modell, W-Modell), iterativ (beispielsweise Rapid Application Development (RAD), Spiralmodell) oder inkrementell (beispielsweise evolutionäre und agile Methoden) .

Im V-Model lässt sich beispielsweise der fundamentale Testprozess des ISTQB® auf der Stufe Systemtest folgendermaßen anpassen:

- Der Systemtest wird gleichzeitig mit dem Projekt geplant, und die Teststeuerung und -überwachung dauert an, bis die Testdurchführung und der Abschluss der Testaktivitäten erfolgt sind.
- Systemtestanalyse und -entwurf finden parallel zum Erstellen von Anforderungsspezifikation, System- und (abstraktem) Architekturentwurf statt, und auf einer niedrigeren Ebene gleichzeitig mit dem Komponentenentwurf.
- Die Bereitstellung der Testumgebung (beispielsweise Testrahmen) kann während des funktionalen Systementwurfs beginnen, obwohl der größte Aufwand normalerweise gleichzeitig mit der Implementierung und dem Komponententest anfällt. Die Arbeit an der Testrealisierung dauert dagegen oft bis wenige Tage vor Beginn der Testdurchführung.
- Die Testdurchführung beginnt, wenn alle Eingangsbedingungen für den Systemtest erfüllt sind (oder auf sie verzichtet wird), was normalerweise bedeutet, dass mindestens der Komponententest und oft auch der Integrationstest abgeschlossen sind. Die Testdurchführung dauert, bis die Testendekriterien erfüllt sind.

- Während der gesamten Testdurchführung werden die Testendekriterien bewertet und die Testergebnisse berichtet, normalerweise mit größerer Häufigkeit und Dringlichkeit, je näher der Projektendtermin rückt.
- Die Testabschlussaktivitäten folgen, wenn die Testendekriterien erfüllt sind und die Testdurchführung als abgeschlossen erklärt wird. Sie können aber manchmal verschoben werden, bis auch der Abnahmetest abgeschlossen ist und alle Projektaktivitäten beendet sind.

Für jede Teststufe und für jede ausgewählte Kombination von Softwarelebenszyklus und Testprozess muss der Testmanager diese Anpassung während der Testplanung und/oder Projektplanung vornehmen. Bei besonders komplexen Projekten, beispielsweise Multisystem-Projekten (verbreitet beim Militär und in Großfirmen), müssen die Testprozesse nicht nur angepasst, sondern auch je nach dem Kontext des Projekts modifiziert werden (beispielsweise wenn es einfacher ist, einen Fehlerzustand auf einer höheren Teststufe aufzudecken als auf einer niedrigeren).

## 1.3 Spezifische Systeme

### *1.3.1 Multisysteme*

Ein Multisystem (System von Systemen) ist ein System zusammenarbeitender Komponenten (mit Hardware, individuellen Softwareapplikationen und Kommunikation), die miteinander einen gemeinsamen Zweck erfüllen; es gibt keine eindeutige Managementstruktur für Multisysteme. Zu Merkmalen und Risiken eines Multisystems gehören:

- Das schrittweise Zusammenfügen unabhängiger Einzelsysteme, damit nicht ein ganzes System völlig neu erstellt werden muss. Dazu lassen sich beispielsweise COTS Systeme mit geringem zusätzlichem Entwicklungsaufwand integrieren.
- Technische und organisatorische Komplexität (beispielsweise zwischen verschiedenen Betroffenen) stellen ein Risiko für effektives Management dar. Wenn die Teilsysteme nach unterschiedlichen Entwicklungsmodellen erstellt werden, können Kommunikationsprobleme zwischen den verschiedenen Teams entstehen (Entwicklung, Testen, Herstellung, Produktionslinie, Nutzer usw.). Das übergeordnete Management eines Multisystems muss mit der hohen technischen Komplexität umgehen können, die mit der Integration der verschiedenen Teilsysteme verbunden ist. Es muss verschiedene organisatorische Themen wie Outsourcing und Off-Shoring bewältigen können.
- Vertraulichkeit und Schutz von spezifischem Know-how, Schnittstellen zwischen verschiedenen Organisationen (beispielsweise im staatlichen und privaten Bereich) oder Regulierungen (beispielsweise ein Monopolverbot) können dazu führen, dass ein komplexes System als ein Multisystem betrachtet werden muss.
- Multisysteme sind in der Regel weniger zuverlässig als Individualsysteme, weil sich jede Beschränkung eines (Teil-)Systems automatisch auf das gesamte Multisystem auswirkt.
- Die einzelnen Komponenten eines Multisystems müssen ein hohes Niveau an technischer und funktioneller Interoperabilität liefern, dadurch wird der Integrationstest kritisch und wichtig und erfordert präzise spezifizierte und vereinbarte Schnittstellen.

#### 1.3.1.1 Management und Testen von Multisystemen

Die höhere Komplexität von Projektmanagement und Konfigurationsmanagement ist Multisystemen gemeinsam. Zu komplexen Systemen und Multisystemen gehören meist eine stärkere Einflussnahme der Qualitätssicherung und definierte Prozesse. Oft sind mit Multisystemen formelle Entwicklungs-Lebenszyklen, Meilensteine und Reviews verbunden.

#### 1.3.1.2 Merkmale des Lebenszyklus von Multisystemen

Zu jeder Teststufe eines Multisystems gehören neben den in Abschnitt 1.2 Testen im Softwarelebenszyklus beschriebenen Merkmalen noch die folgenden:

- mehrstufiges Integrations- und Versionsmanagement,
- lange Projektdauer,
- formale Weitergabe von Informationen zwischen den Projektmitgliedern,
- nicht-gleichzeitige Entwicklung der Komponenten und die Notwendigkeit, Regressionstests auf Multisystem-Ebene durchzuführen,
- Wartungstests, weil Einzelkomponenten wegen Veralterung oder Erweiterung ersetzt werden.

Teststufen müssen bei Multisystemen sowohl mit ihrem eigenen Detaillierungsgrad als auch auf einer höheren Integrationsstufe berücksichtigt werden. So lässt sich der Systemtest für ein Subsystem als Komponententest für die übergeordnete Komponente betrachten.

Normalerweise wird jede Einzelkomponente eines Multisystems auf jeder Teststufe getestet, bevor sie in das Multisystem integriert wird, wobei weitere Tests erforderlich sind.

Spezielle Managementaspekte von Multisystemen enthält Abschnitt 3.11.2.

## *1.3.2 Sicherheitskritische Systeme*

Sicherheitskritische Systeme sind Systeme, bei denen Betriebsausfälle oder Funktionsbeeinträchtigungen (beispielsweise als Folge von unsachgemäßer oder unachtsamer Bedienung) katastrophale oder kritische Konsequenzen haben. Der Lieferant eines sicherheitskritischen Systems kann für den Schaden oder Schadensersatz haftbar gemacht werden; daher werden Testaktivitäten eingesetzt, um die Haftung zu reduzieren. Die Testaktivitäten liefern den Beweis dafür, dass das System adäquat getestet worden ist, um katastrophale oder kritische Folgen möglichst zu vermeiden.

Beispiele für sicherheitskritische Systeme sind Flugzeug-Steuerungssysteme, automatische Handelssysteme, Systeme zur Überwachung von Atomkraftwerken, medizinische Systeme usw.

Folgende Aspekte sollten für sicherheitskritische Systeme umgesetzt werden:

- Rückverfolgbarkeit zu regulatorischen Anforderungen und Methoden zur Konformität (dem Nachweis, dass die Auflagen erfüllt sind),
- striktes Vorgehen bei Entwicklung und Test,
- Sicherheitsanalyse,
- redundante Architektur und ihre Qualifizierung,
- Schwerpunkt auf Qualität,
- hohes Niveau der Dokumentation (Tiefe und Umfang der Dokumentation),
- hohe Auditierbarkeit.

Spezielle Testmanagementaspekte im Zusammenhang mit sicherheitskritischen Systemen enthält Abschnitt 3.11.3.

#### 1.3.2.1 Konformität/Einhalten der Vorschriften

Sicherheitskritische Systeme sind oft Gegenstand staatlicher, internationaler oder branchenspezifischer Auflagen oder Standards (siehe Abschnitt 8.2). Sie können sich auf Entwicklungsprozess und Organisationsstruktur beziehen oder auf das zu erstellende Produkt.

Um die Konformität der Organisationsstruktur und des Entwicklungsprozesses nachzuweisen, können Organisationsdiagramme und Audits ausreichen.

Um die Konformität mit den spezifischen Regulierungen für das zu erstellende System (Produkt) nachzuweisen, muss gezeigt werden, dass jede Anforderung aus diesen Regulierungen adäquat

erfüllt ist. Der Weg von der Anforderung zu ihrer Umsetzung muss sich vollständig verfolgen lassen, um die Konformität zu beweisen. Das beeinflusst während des gesamten Entwicklungsprozesses Management, Entwicklungslebenszyklus, Testaktivitäten und Qualifizierung oder Zertifizierung (durch eine dafür zugelassene Stelle).

#### 1.3.2.2 Sicherheitskritische Systeme und Komplexität

Viele komplexe Systeme und Multisysteme haben sicherheitskritische Komponenten. Der Sicherheitsaspekt ist nicht immer auf System- oder Teilsystemebene ersichtlich, sondern auf der höheren Ebene, auf der komplexe Systeme implementiert werden (beispielsweise Avioniksysteme für Luftfahrzeuge oder Flugsicherungs-Systeme).

Ein Router beispielsweise ist an sich kein sicherheitskritisches System, aber er kann dazu werden, wenn lebenswichtige Daten übertragen werden, wie im Fall von Telemedizin.

Risikomanagement, das die Eintrittswahrscheinlichkeit und/oder die Auswirkungen von Risiken reduziert, ist unverzichtbar für sicherheitskritische Entwicklungen und den Testkontext (siehe Kapitel 3 Testmanagement). Häufig werden außerdem FMEA (siehe Abschnitt 3.10) und Software Common Cause Failure Analysis (Analyse der gemeinsamen Ursachen von Ausfällen) in diesem Zusammenhang eingesetzt.

## 1.4 Metriken und Messung

Eine Vielzahl von Metriken und Messungen sollte während des Softwarelebenszyklus Anwendung finden (beispielsweise Planung, Überdeckung, Arbeitsbelastung usw.). In jedem Fall muss eine Referenzkonfiguration (engl. Baseline) festgelegt und dann der Fortschritt in Bezug auf diese Referenzkonfiguration verfolgt werden.

Erfassbare Größen sind

1. Zeitplan, Überdeckungsgrad und ihre zeitliche Entwicklung
2. Anforderungen, ihre Entwicklung und ihre Auswirkungen auf Zeitplan, Ressourcen und Aufgaben
3. Arbeitspensum und Nutzung der Ressourcen sowie ihre zeitliche Entwicklung
4. Meilensteine und ihr Umfang sowie deren zeitliche Entwicklung
5. Tatsächliche und bis zur Erfüllung der Aufgaben geplante Kosten
6. Risiken und Maßnahmen zur Risikominderung sowie deren zeitliche Entwicklung
7. Gefundene Fehlerwirkungen, behobene Fehlerwirkungen und Korrekturdauer

Mit Metriken können Tester ihrem Management in gleichbleibender Art und Weise berichten, und der Fortschritt im zeitlichen Verlauf lässt sich kohärent verfolgen.

Drei Bereiche sind zu berücksichtigen:

- Metriken definieren: Eine begrenzte Menge von sinnvollen Metriken sollte definiert werden. Sind sie definiert, müssen sich alle Betroffenen auf ihre Interpretation einigen, um zukünftige Diskussionen zu vermeiden, wenn die Metrikwerte vorliegen. Metriken können passend zu den Zielen eines Prozesses oder einer Aufgabe festgelegt werden, für Komponenten oder Systeme, für Einzelpersonen oder Teams. Oft werden tendenziell zu viele Metriken definiert, anstatt die aussagekräftigsten festzulegen.
- Metriken verfolgen: Berichte über und Zusammenstellungen von Metriken sollten möglichst weit automatisiert werden, um den zeitlichen Aufwand für die Generierung der Metrikwerte niedrig zu halten. Abweichungen bei den Daten im zeitlichen Ablauf könnten auch andere Informationen widerspiegeln als die, über deren Interpretation in der Definitionsphase Einigkeit erzielt wurde.

- Metriken berichten: Ziel ist es, die Information für das Management unmittelbar verständlich darzustellen. Präsentationen können eine Momentaufnahme der Metrik zu einem bestimmten Zeitpunkt wiedergeben oder die zeitliche Entwicklung der Metrik(en), sodass sich Tendenzen einschätzen lassen.

## 1.5 Ethische Leitlinien

Durch ihre Tätigkeit im Bereich Softwaretesten erhalten Personen oft Zugang zu vertraulichen und rechtlich privilegierten Informationen. Damit diese Informationen nicht missbräuchlich verwendet werden, sind ethische Leitlinien nötig. In Anlehnung an den Ethik-Kodex von ACM und IEEE stellt das ISTQB® die folgenden ethischen Leitlinien auf.

- Öffentlichkeit
  Das Verhalten zertifizierter Softwaretester soll dem öffentlichen Interesse nicht widersprechen.

- Kunde und Arbeitgeber
  Das Verhalten zertifizierter Softwaretester soll den Interessen ihrer Kunden und Arbeitgeber entsprechen und dabei dem öffentlichen Interesse nicht widersprechen.

- Produkt
  Zertifizierte Softwaretester sollen sicherstellen, dass die Arbeitsergebnisse, die sie liefern (für die von ihnen getesteten Produkte und Systeme), höchste fachliche Anforderungen erfüllen.

- Urteilsvermögen
  Zertifizierte Softwaretester sollen bei ihrer professionellen Beurteilung aufrichtig und unabhängig sein.

- Management
  Zertifizierte Software-Testmanager und Testleiter sollen eine ethische Haltung beim Management des Softwaretestens haben und fördern.

- Berufsbild
  Zertifizierte Softwaretester sollen Integrität und Ansehen ihrer Profession fördern und dabei dem öffentlichen Interesse nicht widersprechen.

- Kollegen
  Zertifizierte Softwaretester sollen sich ihren Kolleginnen und Kollegen gegenüber fair und hilfsbereit verhalten und die Kooperation mit Softwareentwicklern fördern.

- Persönlich
  Zertifizierte Softwaretester sollen sich ihr Leben lang fort- und weiterbilden und eine ethische Haltung in ihrer Berufsausübung vertreten.

# 2. Testprozesse

*Begriffe:*

Abschluss der Testaktivitäten, BS 7925/2, IEEE 829, Testbedingungen, Testbericht, Testdurchführung, Testendekriterien, Testentwurf, Testfall, Testplanung, Testprotokoll, Testrealisierung, Testskript, Teststeuerung, Testszenario, Testvorgehen.

## 2.1 Einführung

Im ISTQB® Foundation-Level-Lehrplan sind folgende Aktivitäten eines grundlegenden Testprozesses beschrieben:

- Testplanung und -steuerung
- Testanalyse und Testentwurf
- Testrealisierung und Testdurchführung
- Testauswertung und Bericht
- Abschluss der Testaktivitäten

Diese Aktivitäten können sequenziell oder teilweise auch parallel ablaufen. So können sich beispielsweise Testanalyse und Testentwurf mit Testrealisierung und Testdurchführung zeitlich überlappen, während die anderen Aktivitäten sequenziell durchgeführt werden.

Da das Testmanagement in einem grundlegenden Zusammenhang mit dem Testprozess steht, müssen Testmanager in der Lage sein, den gesamten Inhalt dieses Kapitels im konkreten Testmanagement eines spezifischen Projekts umzusetzen. Das im ISTQB® Foundation Level erworbene Wissen dagegen ist weitgehend ausreichend für Testanalysts und Technische Testanalysts, mit Ausnahme der oben angeführten Aufgaben in der Testentwicklung. Das dafür nötige Wissen wird in diesem Kapitel allgemein abgehandelt und in den Kapiteln 4 Testverfahren und 5 Test der Softwareeigenschaften vertieft.

## 2.2 Testprozessmodelle

Prozessmodelle sind Annäherungen und Abstraktionen. Sie können nicht den gesamten Testprozess in all seiner Komplexität abdecken, mit all den Nuancen und Aktivitäten, die in realen Projekten oder Vorhaben vorkommen. Ein Testprozessmodell sollte nicht als starres Korsett aufgefasst werden, sondern als Leitfaden, mit dessen Hilfe Testprozesse sich besser verstehen und organisieren lassen.

Dieser Lehrplan verwendet den im ISTQB® Foundation-Level-Lehrplan beschriebenen Testprozess als Beispiel (siehe oben), es gibt aber auch andere wichtige Testprozessmodelle. Drei Beispiele folgen. Bei den drei Modellen handelt es sich um Testprozessmodelle und Modelle zur Verbesserung des Testprozesses. Alle drei Modelle (Practical Software Testing enthält das Test Maturity Model) sind im Hinblick auf die Reifestufen definiert, die sie unterstützen. Weitere Informationen zu den drei Testprozessmodellen und TPI® enthält Abschnitt 8.3 Testverbesserungs-Prozess.

- Practical Software Testing – Test Maturity Model [Burnstein03]
- Critical Testing Processes [Black03]
- Systematic Test and Evaluation Process (STEP)

## 2.3 Testplanung und -steuerung

Dieses Kapitel behandelt die Prozesse bei der Testplanung und -steuerung.

Die Testplanung erfolgt größtenteils in der Anfangsphase des Testens; dazu gehört es, alle Testaktivitäten und Ressourcen zu identifizieren und festzulegen, die zur Erfüllung der in der Teststrategie festgelegten Aufgaben und Testziele notwendig sind.

Das risikoorientierte Testen (siehe Kapitel 3 Testmanagement) liefert für den Testplanungs-Prozess Informationen über geeignete Maßnahmen zur Reduzierung der identifizierten Risiken. Wird beispielsweise bei einem bestimmten Projekt festgestellt, dass die schwerwiegendsten Fehler normalerweise in der Designspezifikation zu finden sind, dann könnte der Testplanungs-Prozess zusätzliche statische Tests (Reviews) der Designspezifikation festlegen, bevor sie in Code umgesetzt wird. Risikoorientiertes Testen liefert dem Testplanungs-Prozess außerdem Informationen über die jeweiligen Prioritäten der verschiedenen Testaktivitäten.

Zwischen Testbasis, Testbedingungen, Testfällen und Testvorgehen kann ein komplexes Beziehungsgeflecht bestehen, das zu vielen wechselseitigen Beziehungen zwischen diesen Arbeitsergebnissen führt. Tester müssen diese Wechselbeziehungen verstehen, um eine effektive Testplanung und Teststeuerung durchführen zu können.

Die Teststeuerung ist eine fortlaufende Aktivität, bei der der tatsächliche Testfortschritt mit dem Plan verglichen und der aktuelle Status berichtet werden, einschließlich möglicher Abweichungen vom Plan. Sie steuert den Testprozess, damit Aufgabenumfang, Teststrategie und Testziele erfüllt werden; auch die Anpassung der Testplanungs-Dokumente nach Bedarf gehört dazu.

Die Teststeuerung muss auf die Informationen durch das Testen und auf veränderte Randbedingungen eines Projekts oder Vorhabens adäquat reagieren. So müssen Risikoanalyse und Zeitplan revidiert werden, wenn beispielsweise beim dynamischen Testen gehäufte Fehler in Bereichen entdeckt werden, wo man sie für unwahrscheinlich gehalten hatte, oder wenn die verfügbare Zeit für die Tests wegen einer Verzögerung beim Testbeginn gekürzt wurde. Das kann dazu führen, dass Tests neu priorisiert und verbleibende Ressourcen neu verteilt werden müssen.

Weitere Informationen zum Inhalt der Testplanungs-Dokumente enthält Kapitel 3 Testmanagement.

Mögliche Metriken zur Überwachung der Testplanung und –steuerung sind

- Risiko- und Testüberdeckung
- Fehlerfindung und -information
- Verhältnis geplanter zu tatsächlich aufgewendeten Stunden für die Entwicklung der Testmittel und die Durchführung der Testfälle

## 2.4 Testanalyse und Testentwurf

Während der Testplanung werden auch die Testziele identifiziert. Sie dienen in der Testanalyse- und Testentwurfsphase dazu:

- Testbedingungen zu identifizieren,
- Testfälle zu entwerfen, mit denen die identifizierten Testbedingungen behandelt werden.

Die bei der Risikoanalyse und Testplanung festgelegten Priorisierungskriterien sollten während des gesamten Testprozesses angewendet werden, von Testanalyse und -entwurf bis hin zu Testrealisierung und -durchführung.

### 2.4.1 Testbedingungen identifizieren

Testbedingungen werden identifiziert durch die Analyse von Testbasis und Testzielen. Sie legen fest, was durch die Anwendung von in der Teststrategie und/oder im Testkonzept identifizierter Testverfahren zu testen ist.

Auf der Basis von funktionalen und nicht-funktionalen Merkmalen der Testobjekte lässt sich entscheiden, wie Granularität und Strukturierung der Testbedingungen festzulegen sind. Folgende Aspekte sind relevant:

1. Granularität der Testbasis: Allgemeine Anforderungen können zunächst zu abstrakten Testkriterien führen, wie „Nachweisen, dass Bildschirm X funktioniert". Daraus lässt sich dann ein konkretes Testkriterium ableiten, beispielsweise „Nachweisen, dass Bildschirm X eine Kontonummer zurückweist, die ein Zeichen zu wenig hat".
2. Behandelte Produktrisiken: Für ein als hoch eingestuftes Risiko könnten beispielsweise sehr detaillierte Testbedingungen als Ziel festgelegt werden.
3. Anforderungen für die Berichterstattung an das Management und die Rückverfolgbarkeit der Informationen.
4. Entscheidung, dass nur mit Testbedingungen gearbeitet wird und keine Testfälle daraus entwickelt werden (z.B. um mit Testbedingungen den Fokus für explorative Tests festzulegen)

### 2.4.2 Testfälle entwerfen

Testfälle werden in einer schrittweisen Ausarbeitung und Verfeinerung der identifizierten Testbedingungen entworfen. Dabei nutzen die Tester Testverfahren (siehe Kapitel 4), die in der Teststrategie festgelegt wurden. Testfälle sollten wiederholbar, nachprüfbar und auf die Anforderungen zurückzuführen sein.

Zum Entwurf von Testfällen gehört die Identifizierung von:

- Vorbedingungen, wie projektbezogene oder lokale Testumgebungen (Testvorrichtungen) und deren geplante Bereitstellung
- Anforderungen an die Testdaten
- erwarteten Ergebnissen aus dem Testfall und dessen Nachbedingungen

Oft stellt die Definition der erwarteten Testergebnisse eine besondere Herausforderung dar. Es gilt, ein oder mehrere Testorakel zu finden, die Sollergebnisse für den Testfall liefern können. Bei der Suche nach Sollergebnissen werden nicht nur Bildschirmausgaben untersucht, sondern auch Daten und Nachbedingungen nach dem Testfall in der Testumgebung.

Wenn die Testbasis klar definiert ist, ist das theoretisch einfach. In der Praxis ist die Testbasis jedoch oft vage oder widersprüchlich, deckt Schlüsselbereiche nicht ab, ist unvollständig oder gar nicht vorhanden. In solchen Fällen müssen Tester entweder selbst fachliche Expertise haben oder sie zumindest abrufen können. Auch wenn die Testbasis gut spezifiziert vorliegt, können komplexe Interaktionen zwischen verschiedenen Stimuli und ausgelösten Reaktionen die Definition der erwarteten Testergebnisse erschweren. Tester brauchen deshalb ein Testorakel. Es hat nur sehr begrenzten Wert einen Test durchzuführen, wenn sich die Richtigkeit der Testergebnisse nicht überprüfen lässt, denn daraus können sich ungültige Abweichungsberichte und unbegründetes Vertrauen in das System ergeben.

Die oben beschriebenen Aktivitäten lassen sich auf allen Teststufen anwenden, nur die Testbasis ändert sich. So werden für Anwender-Abnahmetests in erster Linie die Anforderungsspezifikation, Anwendungsfälle und definierte Geschäftsprozesse als Basis verwendet, während Komponententests meist auf einer detaillierten Entwurfsspezifikation basieren.

Bei der Entwicklung der Testbedingungen und Testfälle entstehen in der Regel verschiedene Arbeitsergebnisse aus der begleitenden Dokumentation. IEEE Standard 829 enthält Vorgaben für diese Dokumentation und erläutert die wichtigsten Dokumente zu Testanalyse und -entwurf, Testentwurfs- und Testfallspezifikation sowie Testrealisierung. In der Praxis gibt es jedoch große Unterschiede bei Umfang und Detaillierungsgrad der Dokumentation von Arbeitsergebnissen. Dies wird beispielsweise beeinflusst durch:

- Projektrisiken (was muss dokumentiert werden und was nicht)
- den Mehrwert, den die Dokumentation für das Projekt schafft
- Normen und Standards, die eingehalten werden müssen
- das Lebenszyklusmodell (bei agilen Methoden wird der Umfang der Dokumentation beispielsweise weitgehend durch enge und häufige Kommunikation im Projektteam minimiert)
- die Anforderung an Rückverfolgbarkeit von der Testbasis über die Testanalyse bis hin zu Testentwurf, Testfällen und Testdurchführung (inklusive der Testergebnisse)

Je nach Testumfang können sich Testanalyse- und Testentwurfsphase auch mit den Merkmalen des Testobjekts befassen. ISO 9126 bietet dafür eine nützliche Referenzgrundlage. Beim Testen von Hardware-/Softwaresystemen können weitere Merkmale relevant sein.

Der Testanalyse- und Testentwurfs-Prozess lässt sich durch eine Verknüpfung mit Reviews und statischer Analyse qualitativ verbessern. Die Durchführung des Testanalyse- und –entwurfsprozesses basierend auf der Anforderungsspezifikation ist beispielsweise eine ausgezeichnete Vorbereitung auf die Review-Sitzung für die Anforderungsspezifikation. Auch die Arbeitsergebnisse von Tests, Risikoanalysen und Testplänen sollten Reviews und statischen Analysen unterzogen werden.

In der Testentwurfsphase lassen sich Anforderungsdetails an die Testinfrastruktur definieren; in der Praxis werden sie aber erst zu Beginn der Testrealisierung endgültig festgelegt. Hinweis: Zur Testinfrastruktur gehört mehr als nur Testobjekte und Testmittel (Beispiele sind Räumlichkeiten, Ausrüstungen, Personal, Software, Werkzeuge, Peripheriegeräte, Kommunikationseinrichtungen, Zugriffsberechtigungen, und alles was sonst zum Durchführen des Test nötig ist).

Metriken für die Überwachung von Testanalyse und -entwurf sind

- prozentualer Anteil der Anforderungen, die von Testbedingungen abgedeckt werden
- prozentualer Anteil der Testbedingungen, die von Testfällen abgedeckt werden
- Anzahl der Fehlerzustände, die während der Testanalyse- und Testentwurfsphase aufgedeckt wurden

## 2.5 Testrealisierung und Testdurchführung

### *2.5.1 Testrealisierung*

Bei der Testrealisierung werden die Testfälle in Testszenarien (Testdrehbuch/Testablaufspezifikation) umgesetzt, wobei Testdaten und Testumgebungen endgültig festzulegen sind. Daraus entsteht der Testausführungsplan. Wenn er vorliegt, kann die Ausführung der Testfälle beginnen. Dazu gehört auch die Überprüfung anhand von expliziten und impliziten Eingangsbedingungen für die betroffene Teststufe.

Die Testszenarien sollten priorisiert sein, damit die in der Teststrategie festgelegten Testziele möglichst effizient erreicht werden. Das kann heißen, die wichtigsten Testszenarien zuerst auszuführen.

Der in der Testrealisierung notwendige Detaillierungsgrad und die damit verbundene Komplexität der Aufgaben können durch den Detaillierungsgrad der anderen Arbeitsergebnisse (Testfälle und Testbedingungen) beeinflusst werden. In manchen Fällen gelten gesetzliche Bestimmungen und die

Tests müssen den Nachweis erbringen, dass die gültigen Normen und Standards tatsächlich erfüllt sind, wie die für den Flugzeugbau von der United States Federal Aviation Administration definierte Norm DO-178B/ED 12B.

Wie in Abschnitt 2.4 angeführt, sind Testdaten für das Testen nötig, und diese Datenmengen können ziemlich umfangreich sein. Während der Testrealisierungsphase erzeugen die Tester Eingabe- und Umgebungsdaten, die in Datenbanken oder anderen Repositories abgelegt werden. Sie erstellen Skripte und andere Datengeneratoren für die Daten, die dann beim Durchführen des Tests als eingehende Last an das System gesendet werden.

In der Testrealisierungsphase sollten die Tester die Reihenfolge der durchzuführenden manuellen und automatisierten Tests endgültig festlegen. Bei einer Automatisierung der Tests gehört zur Testrealisierungsphase auch das Erstellen von Testrahmen und Testskripten. Dabei sind Einschränkungen genau zu beachten, die möglicherweise eine bestimmte Reihenfolge der Tests vorgeben. Abhängigkeiten von bestimmten Testumgebungen oder Testdaten müssen bekannt sein und überprüft werden.

Die Testrealisierung befasst sich außerdem mit der Testumgebung oder den –umgebungen. In dieser Phase und noch vor der Testdurchführung sollten sie vollständig vorhanden und verifiziert sein. Eine zweckdienliche Testumgebung ist unverzichtbar: Sie sollte so beschaffen sein, dass sich vorhandene Fehlerzustände unter definierten Rahmenbedingungen aufdecken lassen; sie sollte normal operieren, solange keine Fehlerwirkungen auftreten, und schließlich sollte sie bei Bedarf in den höheren Teststufen beispielsweise die Produktions- oder Anwendungsumgebung angemessen abbilden.

In der Testrealisierungsphase müssen die Tester bzw. muss der Testmanager sicherstellen, dass die für Erzeugung und Wartung der Testumgebung zuständigen Personen bekannt und verfügbar sind, und dass die Testmittel und Werkzeuge zur Testunterstützung sowie die zugehörigen Prozesse einsatzbereit sind. Das schließt ein: Konfigurationsmanagement, Fehler- und Abweichungsmanagement, Testprotokollierung und Testmanagement. Außerdem müssen die Verfahren verifiziert werden, mit denen die Daten für Testendekriterien und für Berichte über die Testergebnisse gesammelt werden.

Für die Testrealisierung empfiehlt sich ein ausgewogener Ansatz. Risikoorientierte, analytische Teststrategien werden beispielsweise oft mit dynamischen Teststrategien kombiniert. In diesem Fall wird ein Teil des Testdurchführungsaufwands für das Testen ohne Skripte verwendet.

Testen ohne Testskripte sollte aber nicht ad hoc oder ziellos sein, da der Zeitaufwand schwer einzuschätzen ist, falls keine zeitliche Eingrenzung erfolgt (siehe SBTM, Abschnitt 3.11.1). Im Laufe der Zeit haben Tester verschiedene erfahrungsbasierte Verfahren entwickelt, wie Fehlerangriffe (siehe Abschnitt 4.4 und [Whittaker03]), intuitive Testfallermittlung (Fehlerraten) [Myers79] und exploratives Testen. Dabei kommen Testanalyse, Testentwurf und Testrealisierung immer noch vor, allerdings vorwiegend beim Durchführen des Tests und nicht vorab. Wenn solche dynamischen Teststrategien verfolgt werden, dann beeinflussen die Testergebnisse jedes einzelnen Tests Analyse, Entwurf und Realisierung der nachfolgenden Tests. Diese Teststrategien sind „leicht" und oft sehr effektiv beim Auffinden von Fehlern, sie benötigen aber erfahrene Tester. Auch ist ihre Dauer schwer abzuschätzen, sie liefern oft nicht die notwendigen Informationen zur Testüberdeckung, und ohne ein spezifisches Werkzeug für Regressionstests können sie schwierig zu wiederholen sein.

### *2.5.2 Testdurchführung*

Die Testdurchführung beginnt, sobald das Testobjekt zur Verfügung steht und die Eingangsbedingungen für die Testdurchführung erfüllt sind. Die Tests sollten gemäß den Testszenarien (Testablaufspezifikation) durchgeführt werden, obwohl den Testern ein gewisser Spielraum eingeräumt werden kann, damit sie zusätzliche interessante Testszenarien und

Testverhalten verfolgen können, die sich erst während des Testens ergeben. Werden bei einer Abweichung vom festgelegten Testvorgehen Fehlerwirkungen aufgedeckt, müssen die Änderungen der Testvorgehen so dokumentiert werden, dass sich die Fehlerwirkungen reproduzieren lassen. Automatisierte Tests laufen ohne Abweichung von den definierten Vorgaben ab.

Kernstück der Testdurchführung ist der Vergleich zwischen tatsächlichen und erwarteten Testergebnissen. Diese Aufgabe verlangt allergrößte Aufmerksamkeit und Sorgfalt, denn alle Arbeit bei Entwurf und Realisierung des Tests kann umsonst gewesen sein, wenn Fehlerwirkungen übersehen werden (falsch positives Ergebnis), oder wenn korrektes Verhalten irrtümlicherweise als inkorrekt gedeutet wird (falsch negatives Ergebnis). Wenn erwartete und tatsächliche Testergebnisse nicht übereinstimmen, dann liegt eine Abweichung vor. Abweichungen müssen sorgfältig überprüft werden, um die Ursachen festzustellen (und ob es sich dabei um einen Fehlerzustand im Testobjekt handelt), und um die Daten zum Beheben der Abweichung zu sammeln. Weitere Informationen zum Abweichungsmanagement enthält Kapitel 7.

Wird eine Fehlerwirkung entdeckt, sollte zunächst die Testspezifikation einer genauen Bewertung unterzogen werden, um deren Richtigkeit sicherzustellen. Testspezifikationen können aus verschiedenen Gründen inkorrekt sein: bei Problemen mit den Testdaten, bei Fehlern im Testdokument, oder wenn der Test falsch ausgeführt wurde. Stellt sich heraus, dass die Testspezifikation inkorrekt war, muss sie korrigiert und der Test wiederholt werden. Änderungen an Testbasis und Testobjekt können dazu führen, dass eine Testspezifikation nicht mehr stimmt, obwohl sie schon mehrmals erfolgreich abgearbeitet wurde. Es sollte den Testern deshalb immer bewusst sein, dass beobachtete Testergebnisse auch auf einem inkorrekten Test beruhen könnten.

Während der Testdurchführung müssen die Testergebnisse angemessen protokolliert werden. Wenn ein Test durchgeführt wurde, die Ergebnisse aber nicht aufgezeichnet wurden, muss er eventuell wiederholt werden, um das korrekte Ergebnis festzustellen. Das ist ineffizient und führt zu Verzögerungen. (Hinweis: Adäquate Protokollierung kann aber die Vorbehalte bezüglich Überdeckungsgrad und Wiederholbarkeit bei dynamischen Teststrategien entkräften.) Da sich Testobjekt, Testmittel und Testumgebungen weiterentwickeln können, muss aus der Protokollierung hervorgehen, welche Version getestet wurde.

Die Testprotokollierung liefert eine chronologische Aufzeichnung aller relevanten Details der Testdurchführung.

Die Testergebnisse sind sowohl bei einzelnen Tests als auch bei Ereignissen zu protokollieren. Jeder Test sollte eindeutig gekennzeichnet und sein Status im Verlauf der Testdurchführung protokolliert werden. Ereignisse, die sich auf das Durchführen des Tests auswirken, sind zu protokollieren. Die Testprotokollierung sollte ausreichende Informationen aufzeichnen, um die Testüberdeckung zu messen und Gründe für Verzögerungen oder Unterbrechungen im Testbetrieb zu dokumentieren. Darüber hinaus müssen Informationen protokolliert werden, die für Teststeuerung, Testfortschrittsberichte, Messung der Testendekriterien und für Verbesserungen des Testprozesses von Nutzen sind.

Je nach Teststufe und -strategie ist die Protokollierung unterschiedlich. Bei automatisierten Komponententests zeichnen beispielsweise die automatisierten Tests den größten Teil der Protokolldaten selbst auf. Bei manuellen Abnahmetests kann der Testmanager das Testprotokoll erstellen. In manchen Fällen, wie bei der Testrealisierung, beeinflussen Vorschriften oder Audit-Anforderungen die Testprotokollierung.

IEEE Standard 829 „Standard for Software Testing Documentation“ beschreibt die Informationen, die in einem Testprotokoll festzuhalten sind

- Testprotokoll-ID (eindeutige Kennung des Testprotokolls)
- Beschreibung

- Aktivitäts- und Ereigniseinträge

Auch der britische Standard BS 7925-2 enthält eine Beschreibung der zu protokollierenden Informationen.

In manchen Fällen sind Nutzer oder Kunden am Durchführen des Tests beteiligt, was dazu dienen kann, ihr Vertrauen in das System zu stärken. Voraussetzung ist dann aber, dass die Tests im Auffinden von Fehlern weitgehend erfolglos bleiben. Eine derartige Annahme trifft in den frühen Teststufen selten zu, kann während der Abnahmetests aber durchaus gelten.

Mögliche Metriken für die Überwachung der Testrealisierung und Testdurchführung sind

- prozentualer Anteil der konfigurierten Testumgebungen
- prozentualer Anteil der geladenen Testdatensätze
- prozentualer Anteil der durchgeführten Testbedingungen und Testfälle
- prozentualer Anteil der automatisierten Testfälle

## 2.6 Testauswertung und Bericht

Weitere Informationen zu den Themen Dokumentation und Berichte für die Testfortschrittsüberwachung und Teststeuerung enthält Abschnitt 3.6. Für den Testprozess geht es bei der Überwachung des Testfortschritts vor allem darum, Informationen gemäß den Anforderungen an die Berichte zu sammeln. Dazu gehört auch, den Testfortschritt mit Bezug auf die Testendekriterien zu messen.

Zu den Metriken für die Überwachung von Testfortschritt und –abschluss gehört, dass die in der Testplanungsphase festgelegten Testendekriterien abgebildet sind, dazu kann eine oder mehrere der folgenden Metriken gehören:

- Vergleich der geplanten mit den tatsächlich durchgeführten Testbedingungen, Testfällen oder Testspezifikationen, jeweils mit der Angabe, ob sie bestanden wurden oder nicht
- Gesamtzahl der aufgedeckten Fehlerzustände, jeweils mit Schweregrad und Priorität, für die korrigierten und die noch offenen Fehler
- Anzahl der Änderungen (Änderungsanträge), die erzeugt, akzeptiert (implementiert) und getestet wurden
- geplante Kosten gegenüber tatsächlichen Kosten
- geplanter Zeitaufwand gegenüber tatsächlich aufgewendeter Zeit
- identifizierte Risiken, eingeteilt in durch die Testaktivitäten geminderte Risiken und verbleibende
- prozentualer Anteil der Testzeit, die durch blockierende Ereignisse verloren ging
- Testobjekte, die einem Fehlernachtest unterzogen wurden
- geplante Gesamttestzeit gegenüber tatsächlich aufgewendeter Testzeit

Für einen Testbericht spezifiziert IEEE Standard 829 folgende Abschnitte:

- Kennung/ID des Testberichts
- Zusammenfassung
- Abweichungen
- umfassende Bewertung
- Zusammenfassung der Testergebnisse
- Testauswertung
- Zusammenfassung der Testaktivitäten
- Genehmigungen/Freigaben

Die Testberichte können jeweils nach Abschluss einer Teststufe erstellt werden, aber auch zum Abschluss der gesamten Testarbeiten.

## 2.7 Abschluss der Testaktivitäten

Nachdem die Testdurchführung als abgeschlossen erklärt wurde, sollten die wichtigsten Ergebnisse festgehalten und entweder an die zuständige Person weitergeleitet oder archiviert werden. Man bezeichnet das als den Abschluss der Testaktivitäten. Testabschlussaktivitäten können in folgende vier Hauptgruppen eingeteilt werden:

1. Sicherstellen, dass alle Testaufgaben tatsächlich abgeschlossen sind: So sollten alle geplanten Tests tatsächlich durchgeführt oder aber gezielt übersprungen worden sein. Alle bekannten Fehlerzustände sollten entweder behoben und nachgetestet oder auf einen zukünftigen Release verschoben worden sein, oder aber als permanente Einschränkung akzeptiert sein.
2. Die wertvollen Arbeitsergebnisse den Personen oder Stellen liefern, die sie benötigen: So sollten beispielsweise diejenigen, die das System benutzen oder seine Benutzung betreuen werden (beispielsweise Support), erfahren, welche bekannten Fehler verschoben oder akzeptiert wurden. Die für die Wartungstests zuständigen Personen sollten Tests und Testumgebungen erhalten. Ein anderes Arbeitsergebnis könnte ein Satz manueller oder automatisierter Regressionstests sein.
3. Besprechungen über die Erfahrungen aus dem Testen (Bewertungssitzung,“lessons learned“) halten oder an ihnen teilnehmen: In diesen Besprechungen können wichtige Erkenntnisse sowohl aus dem eigentlichen Testprozess als auch aus dem gesamten Softwarelebenszyklus dokumentiert und daraus Maßnahmen abgeleitet werden, damit sie sich nicht wiederholen. Falls sich Probleme nicht lösen lassen, können sie zumindest in zukünftigen Projektplänen berücksichtigt werden. Beispiele:
    a. Weil es unerwartete Fehleranhäufungen erst relativ spät aufdecken konnte, kommt das Team vielleicht zu der Erkenntnis, dass an den Besprechungen zur Qualitätsrisikoanalyse bei zukünftigen Projekten ein breiterer Querschnitt an Nutzern teilnehmen sollte.
    b. Möglicherweise waren die Testschätzungen sehr unzutreffend, sodass sich hieraus Lehren für zukünftige Schätzaktivitäten ziehen lassen, vor allem aus den Ursachen. So kann es an ineffektivem Testen gelegen haben, oder die Schätzung war von vornherein zu niedrig angesetzt.
    c. Die Tester können Tendenzen feststellen und Ursache und Wirkung bei aufgedeckten Fehlern analysieren. Dazu verfolgen sie, warum und wann die Fehler auftraten, und untersuchen, ob sich Tendenzen abzeichnen. So könnten späte Änderungsanträge die Qualität von Analyse und Entwicklung beeinflusst haben. Auch nach ungeeigneten Praktiken können die Tester suchen, und wie viel Zeit diese gekostet haben, wie das Auslassen einer Teststufe, die die Fehler früher und kosteneffektiver aufgedeckt hätte. Einige Fehlertendenzen lassen sich auch mit Rahmenbedingungen in Zusammenhang bringen, wie dem Einsatz neuer Technologien, Personalwechsel oder fehlender fachliche Qualifikation.
    d. Mögliche Verbesserungen für den Testprozess identifizieren.
4. Ergebnisse, Protokolle, Berichte und andere Dokumente und Arbeitsergebnisse im Konfigurationsmanagement-System archivieren, mit Anbindung an das System. So sollten beispielsweise Testkonzept und Projektplan im Planungsarchiv archiviert werden und eindeutig auf das System und die Version verweisen, für die sie eingesetzt wurden.

All diese Aufgaben sind wichtig, auch wenn sie oft vergessen werden, und sie sollten explizit ein Teil des Testkonzepts sein.

Oft werden eine oder mehrere dieser Aufgaben ausgelassen, meist weil das Testteam zu früh aufgelöst wird, weil Zeit oder Ressourcen für nachfolgende Projekte benötigt werden, oder weil das Team überarbeitet und erschöpft ist. Bei Vertragsprojekten, beispielsweise kundenindividuellen Entwicklungsaufträgen, sollten die notwendigen Aufgaben im Vertrag spezifiziert sein.

Mögliche Metriken für den Abschluss der Testaktivitäten sind

- prozentualer Anteil der bei der Testdurchführung ausgeführten Testfälle (Überdeckung)
- prozentualer Anteil der Testfälle, die in das Repository für wiederverwendbare Testfälle aufgenommen wurden
- Anteil der automatisierten gegenüber den zu automatisierenden Testfällen
- prozentualer Anteil der Testfälle, die als Regressionstests gelten
- prozentualer Anteil ungelöster Fehlerberichte, die geschlossen wurden (beispielsweise verschoben, keine weiteren Maßnahmen, Änderungsanforderung usw.)
- prozentualer Anteil der identifizierten und archivierten Arbeitsergebnisse.

# 3. Testmanagement

*Begriffe:*

FMEA, Mastertestkonzept, Produktrisiko, Projektrisiko, Risikobeherrschung bzw. -steuerung, Risikoanalyse, Risikoidentifizierung, Risikomanagement, risikoorientierter Test, Risikostufe, Risikotyp, session-based Testmanagement, Stufentestkonzept, Testfortschrittsbericht, Testmanagement, Testpunktanalyse (TPA), Testrichtlinie, Testschätzung, Teststeuerung, Teststrategie, Teststufe, Testüberwachung, zeitliche Testplanung, Wideband Delphi.

## 3.1 Einführung

Dieses Kapitel deckt die Wissensgebiete ab, die speziell für Testmanager relevant sind.

## 3.2 Testmanagement-Dokumentation

Dokumentation entsteht oft als Teil des Testmanagements. Während Bezeichnungen und Umfang der Testmanagement-Dokumente oft variieren, so sind doch die folgenden typischen Testmanagement-Dokumente in Unternehmen und Projekten üblich:

- Testrichtlinie: Sie beschreibt die Unternehmensphilosophie für das Testen (möglicherweise auch für die Qualitätssicherung).
- Teststrategie: Sie beschreibt die Testmethoden beim Risikomanagement von Produkt- und Projektrisiken, die Aufteilung des Tests in Schritte, die generelle Teststrategie, Teststufen oder Testphasen sowie übergeordnete testbezogene Aktivitäten im Unternehmen.
- Mastertestkonzept (Projekttestkonzept, Testvorgehensweise): Es beschreibt die Anwendung der Teststrategie für ein bestimmtes Projekt, einschließlich der jeweiligen Teststufen und deren Verhältnis zueinander.
- Stufentestkonzept (Phasentestplan): Es beschreibt die in den jeweiligen Teststufen durchzuführenden Aktivitäten, einschließlich etwaiger Erweiterungen des Mastertestkonzepts für die spezifisch dokumentierte Teststufe.

In einigen Unternehmen und Projekten werden die oben aufgeführten Dokumente in einem einzigen Dokument zusammengefasst, oder sie sind in anderen Dokumenten enthalten, oder der Inhalt wird einfach als intuitive, ungeschriebene oder traditionelle Testmethodik zu Grunde gelegt. Größere, formal strukturierte Unternehmen und Projekte gestalten die Dokumente meist als Arbeitsergebnisse in schriftlicher Form, während in kleineren Unternehmen und Projekten meist weniger dokumentiert wird. Der Lehrplan beschreibt jedes der oben angeführten Dokumente, in der Praxis gibt aber der konkrete Unternehmens- und Projektzusammenhang vor, wie das jeweilige Dokument richtig anzuwenden ist.

### *3.2.1 Testrichtlinie*

Die Testrichtlinie beschreibt die Unternehmensphilosophie für das Testen (möglicherweise auch für die Qualitätssicherung). Verschiedentlich wird die Testrichtlinie auch als Testpolitik bezeichnet. Sie liegt entweder in schriftlicher Form vor oder als eine Managementanweisung und beschreibt die allgemeinen Ziele, die das Unternehmen durch das Testen erreichen möchte. Die Richtlinie kann

durch die Abteilungen IT, Forschung und Entwicklung oder Produktentwicklung erstellt werden und sollte die unternehmerischen Werte und Ziele hinsichtlich des Testens darstellen.

Die Testrichtlinie kann entweder eine Ergänzung zur allgemeinen Qualitätsrichtlinie des Unternehmens sein oder ein Teil davon. Die Qualitätsrichtlinie legt fest, welche Werte und Ziele das Management in Bezug auf die Qualitätsansprüche des Unternehmens anstrebt.

Wenn eine Testrichtlinie vorliegt, kann sie ein kurzes abstraktes Dokument sein, das

- das Testen definiert, im Sinne einer vertrauensbildenden Maßnahme, dass das System wie beabsichtigt funktioniert, oder dass Fehler aufgedeckt werden.
- einen fundamentalen Testprozess festschreibt, der beispielsweise bestehen kann aus: Testplanung und -steuerung, Testanalyse und -entwurf, Testrealisierung und Testdurchführung, Testauswertung der Testendekriterien und Berichterstattung sowie Abschluss der Testaktivitäten.
- beschreibt, wie Wirksamkeit und Effizienz des Testens zu bewerten sind, beispielsweise die Fehlerfindungsrate und die Kosten von im Test gefundenen Fehlern im Gegensatz zu den Kosten, die Fehlerzustände nach der Freigabe verursachen.
- gewünschte Qualitätsziele definiert, wie Zuverlässigkeit (beispielsweise gemessen an der Ausfallrate) oder Benutzbarkeit.
- Aktivitäten für die Testprozessverbesserung vorgibt, beispielsweise Einsatz des Test Maturity Modells (TMM) oder des Test Process Improvement Modells (TPI™), oder die Umsetzung von Erkenntnissen aus abgeschlossenen Projekten.

Die Testrichtlinie kann sich sowohl auf Testaktivitäten für Neuentwicklungen als auch auf die Wartung beziehen. Sie kann außerdem ein verbindlicher unternehmensweiter Standard für Testterminologie sein.

### *3.2.2 Teststrategie*

Die Teststrategie beschreibt die Testmethoden des Unternehmens. Beschrieben werden auch Produkt- und Projekt-Risikomanagement, die Aufteilung des Testens in Teststufen oder Testphasen sowie die übergeordneten testbezogenen Aktivitäten. Der in der Teststrategie beschriebene Testprozess und die Testaktivitäten sollten der Testrichtlinie entsprechen. Die Teststrategie sollte die allgemeinen Testerfordernisse für das Unternehmen oder für ein oder mehrere Projekte liefern.

Wie im Foundation-Level-Lehrplan beschrieben, lassen sich generelle Teststrategien und -vorgehensweisen) nach dem Beginn der Testentwurfsphase klassifizieren:

- Eine vorbeugende Strategie entwirft den Test frühzeitig, um Fehler zu verhindern.
- Eine reagierende Strategie entwirft den Test, nachdem die Software oder das System erstellt wurde.

Typische Teststrategien und Testvorgehensweisen sind u.a.:

- analytische Strategien, wie das risikoorientierte Testen
- modellorientierte Strategien, wie das am Anwendungsprofil orientierte Testen
- methodische Strategien, beispielsweise auf Qualitätsmerkmalen basierend
- prozesskonforme oder standardkonforme Strategien, beispielsweise auf dem Standard IEEE 829 basierend
- dynamische und heuristische Strategien, wie die Nutzung fehlerbasierter Angriffe
- beratende Strategien, wie das anwendergesteuerte Testen
- Regressionstest-Strategien, beispielsweise weitgehende Automatisierung

Verschiedene Strategien lassen sich kombinieren. Die ausgewählte Strategie sollte sich an den Erfordernissen und Mitteln des Unternehmens orientieren, und Unternehmen können die Strategien auf ihre eigenen speziellen Abläufe und Projekte zuschneiden.

Oft erläutert eine Teststrategie die Projekt- und Produktrisiken, und wie im Testprozess mit diesen Risiken umzugehen ist. Der Zusammenhang zwischen Test und Risiken sollte ausdrücklich erklärt werden, wie auch die Möglichkeiten für Risikominderung und -management.

Die Teststrategie kann die auszuführenden Teststufen beschreiben. Sie sollte dann eine grobe Leitlinie für die Eingangs- und Ausgangsbedingungen jeder Teststufe sowie für die Beziehungen zwischen den Teststufen (beispielsweise Aufteilung Testüberdeckungsziele) vorgeben.

Die Teststrategie kann außerdem beschreiben

- Vorgehensweise für die Integration
- Testspezifikationsverfahren
- Unabhängigkeit des Testens (kann je nach Teststufe variieren)
- Verpflichtend oder freiwillig einzuhaltende Normen/Standards
- Testumgebungen
- Testautomatisierung
- Wiederverwendbarkeit von Software und Arbeitsergebnissen des Testens
- Fehlernachtest und Regressionstests
- Teststeuerung und -berichte
- Testmaße und -metriken
- Abweichungsmanagement
- Konfigurationsmanagement der Testmittel

Es sollten sowohl kurz- als auch langfristige Teststrategien definiert werden, entweder in einem oder mehreren Dokumenten. Unterschiedliche Teststrategien eignen sich für unterschiedliche Unternehmen und unterschiedliche Projekte. Wenn es beispielsweise um sicherheitsrelevante oder sicherheitskritische Anwendungen geht, dann sind intensivere Strategien geeigneter.

### *3.2.3 Mastertestkonzept*

Das Mastertestkonzept beschreibt die Anwendung der generellen Teststrategie für ein spezifisches Projekt. Dazu gehören die auszuführenden Teststufen und die Beziehungen zwischen ihnen. Das Mastertestkonzept sollte mit der Testrichtlinie und der Teststrategie konsistent sein. Falls es in bestimmten Bereichen davon abweicht, sollten die Abweichungen und Ausnahmen erklärt werden. Das Mastertestkonzept sollte den Projektplan oder das Betriebshandbuch ergänzen und den Testaufwand beschreiben, der Teil eines größeren Projekts oder einer größeren Operation ist.

Inhalt und Aufbau des Mastertestkonzepts können je nach Unternehmen, den im Unternehmen verwendeten Dokumentationsstandards und der im Projekt gelebten Formalität variieren. Typische Themen für ein Mastertestkonzept sind

- Objekte, die getestet oder nicht getestet werden sollen
- Qualitätsmerkmale, die getestet oder nicht getestet werden sollen
- Testplan und Budget für das Testen (in Anlehnung an das Projekt- oder Betriebsbudget)
- Testdurchführungszyklen und deren Beziehung zum Release-Plan für die Software
- wirtschaftliche Begründung für das Testen und Geschäftswert des Testens
- Beziehungen und Arbeitsergebnisse der Testabteilung zu anderen Personen oder Abteilungen
- definierte Abgrenzungen zwischen Testobjekten in den jeweiligen Teststufen

- Spezifikation der Eingangsbedingungen, Unterbrechungs- und Wiederaufnahmekriterien sowie Ausgangsbedingungen für jede Teststufe und der Beziehungen zwischen den Teststufen
- Testprojektrisiken

In kleineren Projekten oder Unternehmen, in denen nur eine Teststufe formal getestet wird, werden Mastertestkonzept und Testkonzept für diese Teststufe oft in einem Dokument zusammengefasst. Ist der Systemtest die einzige formale Teststufe, mit einem informellen, von den Entwicklern durchgeführten Komponenten- und Integrationstest und einem informellen Abnahmetest, der vom Kunden als Teil eines Beta-Tests durchgeführt wird, so können all diese Elemente im Systemtestkonzept enthalten sein.

Meist hängt das Testen von anderen Aktivitäten im Projekt ab. Wenn diese Aktivitäten nicht ausreichend dokumentiert sind, vor allem was ihren Einfluss auf und ihre Beziehung zum Testen betrifft, dann kann dies im Mastertestkonzept (oder im entsprechenden Stufentestkonzept) abgedeckt werden. Ist beispielsweise der Konfigurationsmanagement-Prozess nicht dokumentiert, dann sollte im Mastertestkonzept festgehalten werden, wie Testobjekte an das Testteam geliefert werden.

### *3.2.4 Stufentestkonzept*

Das Stufentestkonzept beschreibt die in jeder Teststufe auszuführenden Aktivitäten. Wenn nötig, kann es das Mastertestkonzept für die dokumentierte Teststufe erweitern und beispielsweise Details zu Terminplan, Aufgaben und Meilensteinen spezifizieren, die im Mastertestkonzept nicht dokumentiert sind. Wenn unterschiedliche Standards und Dokumentvorlagen für die Spezifikation der Tests in verschiedenen Teststufen gelten, werden sie im Stufentestkonzept dokumentiert.

In weniger formalen Projekten oder Unternehmen entsteht oft ein Stufentestkonzept mit nur einer Stufe als einziges Testmanagementdokument. Es kann dann einige der in den Abschnitten 3.2.1, 3.2.2 und 3.2.3 erwähnten Informationen enthalten.

## 3.3 Dokumentvorlagen für Testkonzepte

Wie in Abschnitt 3.2 erwähnt, variieren Inhalt und Aufbau des Mastertestkonzepts je nach Unternehmen, Dokumentationsstandards und der Formalität des Projekts. Viele Unternehmen erstellen eigene Dokumentvorlagen oder passen vorhandene an, um so für die Testkonzeptdokumentation Einheitlichkeit und Lesbarkeit projekt– und prozessübergreifend sicherzustellen. Dokumentvorlagen für die Testkonzeptdokumentation stehen zur Verfügung.

IEEE Standard 829 „Standard for Software Testing Documentation“ enthält Vorlagen für die Testdokumentation und eine Anleitung zu ihrer Anwendung sowie eine Anleitung für die Erstellung von Testkonzepten. Die Norm befasst sich auch mit dem verwandten Thema der Testobjektübergabe (Freigabe von Testobjekten für den Test).

## 3.4 Testaufwandsschätzung

Aufwandsschätzungen als Managementaktivität führen zu einem ungefähren Kosten- und Terminplan für die Aktivitäten in einem bestimmten Unternehmen oder Projekt. Die besten Aufwandsschätzungen:

- repräsentieren die kollektive Erfahrung von erfahrenen Praktikern und werden von allen Betroffenen getragen
- liefern detaillierte Aufstellungen der Kosten, Ressourcen, Aufgaben und beteiligten Mitarbeiter
- zeigen die wahrscheinlichen Kosten, Aufwand und Dauer für jede geschätzte Aktivität.

Die Schätzung von Aufgaben in der Software- und Systementwicklung ist bekanntermaßen schwierig, sowohl technisch als auch politisch, obwohl es im Projektmanagement Best Practice-Verfahren für die Schätzung gibt. Die Testschätzung ist die Anwendung dieser Verfahren auf die Testaktivitäten in einem Projekt oder Unternehmen.

Die Testschätzung sollte alle Aktivitäten im Testprozess einschließen, von Testplanung und Teststeuerung über Testanalyse und Testentwurf, Testrealisierung und -durchführung, abschließende Testauswertung und Berichte bis zum Abschluss der Testaktivitäten. Geschätzte Kosten, Aufwand und vor allem Dauer der Testdurchführung interessieren das Management oft besonders, weil die Testdurchführung typischerweise auf dem kritischen Pfad des Projekts liegt. Schätzungen der Testdurchführung sind aber schwierig und tendenziell unzuverlässig, wenn die Gesamtqualität der Software schlecht ist oder nicht bekannt. Es ist gängige Praxis, die Anzahl von erforderlichen Testfällen zu schätzen. Die getroffenen Annahmen bei der Aufwandschätzung sollten immer als Teil der Aufwandsschätzung dokumentiert werden.

Die Testschätzung sollte alle Faktoren berücksichtigen, die Kosten, Aufwand und Dauer der Testaktivitäten beeinflussen können. Dazu gehören:

- erforderliches Qualitätsniveau des Systems
- Größe des zu testenden Systems
- historische Daten aus früheren Testprojekten (auch Benchmark-Daten)
- Prozessfaktoren, wie die Entwicklung der Teststrategie; Wartungslebenszyklus und Prozessreife; Richtigkeit der Projektschätzung
- Materialfaktoren, wie Testautomatisierung und Testwerkzeuge, Testumgebung, Testdaten, Entwicklungsumgebung(en); Projektdokumentation (beispielsweise Anforderungen, Entwürfe, usw.) und wiederverwendbare Arbeitsergebnisse
- Personalfaktoren, wie Manager und Technische Leiter; Engagement und Erwartungen der Geschäftsleitung und des oberen Managements; fachliches Können, Erfahrung und Motivation im Projektteam, Stabilität des Projektteams, Beziehungen der Projektmitglieder untereinander; Hilfe bei der Nutzung der Test- und Debuggingumgebung; Verfügbarkeit qualifizierter Auftragnehmer und Berater; Fachwissen.

Weitere Faktoren können die Testaufwandsschätzung beeinflussen: Komplexität des Testprozesses; Technik, Organisation; Anzahl der Betroffenen beim Testen; viele räumlich getrennte Teilteams, außergewöhnlicher Aufwand für den Aufbau der Testmannschaft, Training und Einarbeitung; Anpassung oder Entwicklung von neuen Testwerkzeugen, Verfahren, kundenspezifische Hardware, Anzahl der Testmittel; Anforderungen für eine außergewöhnlich detaillierte Testspezifikation, besonders bei Anwendung eines nicht vertrauten Dokumentationsstandards; komplexe Terminierung der Komponentenbeschaffung, besonders für Integrationstest und Testentwicklung; und veränderliche Testdaten (beispielsweise Daten mit Löschfristen).

Die Testaufwandsschätzung lässt sich Bottom-Up oder Top-Down durchführen, dabei können die folgenden Testaufwandsschätzverfahren eingesetzt werden:

- Intuition und Raten
- Erfahrungen aus früheren Projekten
- Aufteilen in Aufgabenblöcke und Erstellung eine Projektstrukturplans (WBS, Work Breakdown Structures)
- gemeinsame Schätzungen im Team (beispielsweise Wideband Delphi)
- Drei-Punkt-Schätzung
- Testpunktanalyse (TPA) [Pol02]
- Unternehmensstandards und Normen

- prozentualer Anteil am Gesamtprojekt oder an bestimmten Mitarbeitergruppen (beispielsweise das Verhältnis von Testern zu Entwicklern)
- Modelle, basierend auf gemessenen Werten, zum Schätzen der Anzahl von Fehlerwirkungen, Testzyklen, Testfällen; des durchschnittlichen Aufwands je Test und der Anzahl von Regressionstestzyklen
- Durchschnittswerte der Branche und Vorhersagemodelle, wie Testpunkte (test points), Funktionspunkte (function points), Anzahl der Codezeilen, geschätzter Aufwand der Entwicklerinnen und Entwickler oder andere Projektparameter

Wenn die Testaufwandsschätzung erstellt ist, muss sie meist mit einer Begründung (siehe Abschnitt 3.7) dem Management vorgelegt werden. Oft folgen Verhandlungen, die nicht selten zu einer Überarbeitung der vorgelegten Zahlen führen. Im Idealfall ist die endgültige Version der Testaufwandsschätzung der beste mögliche Kompromiss zwischen Unternehmenszielen und Projektzielen hinsichtlich Qualität, Zeitplan, Budgetierung und erwarteter Funktionalität.

## 3.5 Zeitliche Testplanung

Wenn Aktivitäten vorab geplant werden, lassen sich meist schon im Vorfeld zum einen Risiken in Zusammenhang mit diesen Aktivitäten entdecken und beherrschen, und zum andern diese Aktivitäten mit den Beteiligten sorgfältig und frühzeitig koordinieren, was sich in einem qualitativ hochwertigen Plan niederschlägt. Das gilt auch für die Testplanung. Testplanung auf Basis der Testschätzung bietet noch mehr Vorteile:

- Entdeckung und Management von Projektrisiken und Problemen auch außerhalb des Testens
- Entdeckung und Management von Produktrisiken (Qualitätsrisiken) und Problemen vor Beginn der Testdurchführung
- Erkennen von Problemen im Projektplan oder bei anderen Arbeitsergebnissen des Projekts
- Chancen auf Bewilligung einer Aufstockung des Testteams, des Budgets, des Aufwands und/oder der Dauer, um damit eine höhere Qualität zu erreichen
- Identifizierung kritischer Komponenten, um die frühere Lieferung dieser Komponenten in den Test zu beschleunigen

Bei der Testplanung sollte das Testteam eng mit den Entwicklern zusammenarbeiten, denn das Testen hängt stark vom Zeitplan für Entwicklung und Lieferung ab.

Bis alle zur Erstellung des Testkonzepts notwendigen Informationen vorliegen, kann aber wertvolle Zeit verstrichen sein und der mögliche Nutzen verloren gehen. Deshalb sollten Entwürfe von Testkonzepten möglichst früh erstellt und vorgelegt werden. Wenn dann neue Informationen verfügbar sind, kann sie der Verfasser des Testkonzepts (in der Regel ein Testmanager) einarbeiten. Dieser iterative Ansatz für Erstellung, Freigabe und Review des Testkonzepts fördert Konsens, Kommunikation und Diskussion beim Testen.

## 3.6 Testfortschritt überwachen und steuern

Die fünf wesentlichen Aspekte für die Überwachung des Testfortschritts sind

- Produktrisiken
- Fehlerzustände
- Tests
- Überdeckungsgrad
- Vertrauen in die Softwarequalität

Während des Betriebs oder eines Projekts werden Produktrisiken, Fehlerzustände, Testfälle und Überdeckungsgrad auf spezielle Art und Weise gemessen und berichtet (Testfortschrittsbericht). Die Messungen sollten mit den im Testkonzept definierten Testendekriterien verknüpft sein. Vertrauen in der Softwarequalität, auch wenn es durch Umfragen messbar ist, wird normalerweise subjektiv berichtet.

Mögliche Metriken für Produktrisiken sind

- Anzahl der Restrisiken einschließlich Art des Risikos und Risikostufe
- Anzahl der geminderten Risiken einschließlich Art des Risikos und Risikostufe

Mögliche Metriken für Fehler sind

- Gesamtzahl der gemeldeten (identifizierten) gegenüber der Gesamtzahl behobener (abgeschlossener) Fehlerzustände
- durchschnittliche Zeit zwischen dem Auftreten von Fehlerwirkungen
- Analyse der Anzahl Fehlerzustände, bezogen auf bestimmte Testobjekte oder Komponenten; Ursachen, Quellen; Testversionen; Phasen, in denen sie eingeführt, festgestellt oder entfernt wurden, und in einigen Fällen Eigentümer der Fehlermeldung
- Trends bei der Dauer zwischen Eingehen der Fehlermeldung und Behebung

Mögliche Metriken für Tests sind

- Gesamtzahl geplanter, spezifizierter (entwickelter), durchgeführter, bestandener/nicht bestandener, blockierter und ausgelassener Tests
- Status der Regressions- und Fehlernachtests
- Anzahl für das Testen geplanter gegenüber tatsächlich geleisteten Stunden pro Tag

Mögliche Metriken für die Testüberdeckung sind

- Anforderungsüberdeckung und Überdeckung der entwickelten Komponenten
- Risikoüberdeckung
- Testumgebungs-/Konfigurationsüberdeckung

Die Messungen können in Textform, tabellarisch oder graphisch berichtet werden, und lassen sich für verschiedene Zwecke nutzen, beispielsweise:

- Analysen, um anhand der Testergebnisse herauszufinden, was mit dem Produkt, im Projekt oder Prozess geschieht
- Berichte, um die Testergebnisse an Projektmitglieder und Betroffene zu kommunizieren
- Teststeuerung, um den Verlauf eines Tests oder auch des Projekts als Ganzes zu ändern und um die Ergebnisse der Kurskorrektur zu überwachen

Wie die Messwerte der Tests geeignet gesammelt, analysiert und berichtet werden, hängt vom spezifischen Informationsbedarf ab, sowie den Zielen und den Fähigkeiten der Adressaten, die diese Messungen nutzen.

Wenn der Steuerungsaufwand eines Projekts anhand der Testergebnisse gemessen oder beeinflusst werden soll, bestehen folgende Möglichkeiten:

- Qualitätsrisikoanalyse, Testprioritäten und/oder Testkonzepte überarbeiten
- weitere Ressourcen bereitstellen oder den Testaufwand der vorhandenen Ressourcen erhöhen
- Auslieferungstermin verschieben
- Testendekriterien abschwächen oder verschärfen

Die Umsetzung erfordert normalerweise Konsens unter den Projektmitarbeitern und Betroffenen im Unternehmen sowie die Zustimmung von Projektmanagern oder Geschäftsführung.

Wie der Testbericht strukturiert ist, hängt weitgehend von den Adressaten ab, beispielsweise Projektmanagement oder Geschäftsführung. Projektmanagerin oder Projektmanager interessieren sich wahrscheinlich für ausführliche Informationen zu Fehlerzuständen, während das Hauptinteresse der Geschäftsführung bei Informationen über den Status der Produktrisiken liegen dürfte.

IEEE Standard 829 „Standard for Software Test Documentation“ liefert eine Dokumentvorlage für einen zusammenfassenden Testbericht.

Bei Abweichungen vom Testkonzept sollte die Teststeuerung auf Basis des Testfortschrittsberichts eingreifen, um Abweichungen zu minimieren. Folgende Steuerungsmaßnahmen sind möglich:

- Testfallpriorisierung überdenken
- zusätzliche Ressourcen beschaffen
- Release-Termin verschieben
- Projektumfang (Funktionalität) ändern
- Testendekriterien überdenken (nur mit Zustimmung der Betroffenen).

## 3.7 Geschäftswert des Testens

Zwar halten die meisten Unternehmen das Testen für in gewisser Weise wertvoll, aber nur wenige Manager, Testmanager eingeschlossen, können diesen Wert quantifizieren, beschreiben oder in Worte fassen. Viele Testmanager, Testleiter und Tester konzentrieren sich vor allem auf die Ausführungsdetails des Testens (Aspekte, die die konkrete Aufgabe oder Teststufe betreffen), lassen aber die übergeordneten strategischen Gesichtspunkte außer Acht, für die sich andere Projektbeteiligte interessieren, vor allem Manager.

Das Testen bringt dem Unternehmen, dem Projekt und/oder dem Betrieb Nutzen, quantitativ wie qualitativ:

- Der quantitative Nutzen des Testens liegt darin, dass es Fehler vor einer Freigabe vermeidet oder behebt, Fehler vor der Freigabe bekannt macht, Risiken durch die Tests vermindert und Informationen über den Projekt-, Prozess- und Produktstatus liefert.
- Der qualitative Nutzen des Testens liegt in gesteigerter Reputation für Qualität, reibungsloseren und besser berechenbaren Releases, neuem und/oder gesteigertem Vertrauen in die Software, Schutz vor Haftungsansprüchen sowie im Reduzieren des Risikos, ganze Aufträge oder sogar Menschenleben zu verlieren.

Testmanager und Testleiter sollten verstehen, wo der relevante geschäftliche Nutzen für ihr Unternehmen, ihr Projekt und/oder den Betriebsablauf liegt, und sie sollten anderen diesen geschäftlichen Nutzen des Testens vermitteln können.

Eine bewährte Methode für die Messung von quantitativem Nutzen und Effizienz des Testens ist in dem Begriff Qualitätskosten zusammengefasst (manchmal auch als Kosten schlechter Qualität bezeichnet). Die Qualitätskosten fassen Projekt- oder Betriebskosten in drei Kategorien zusammen:

1. Kosten für die Vorbeugung
2. Kosten für die Aufdeckung
3. Kosten für die interne Fehlerwirkungen
4. Kosten für die externe Fehlerwirkungen

Die Gesamtkosten für die Vorbeugung und für die Aufdeckung sind meist weniger als die Kosten der Fehlerwirkungen; das Testen ist deshalb außerordentlich wertvoll. Testmanager und Testleiter haben überzeugende wirtschaftliche Argumente für das Testen, wenn sie die Kosten in diesen Kategorien aufzeigen können.

## 3.8 Verteiltes Testen, Outsourcing und Insourcing

Oft besteht das Testteam, das den gesamten Testaufwand leistet, nicht nur aus Mitarbeitern des Projektteams und arbeitet nicht am selben Ort wie das Projektteam. Beim Testen an mehreren Standorten spricht man von verteiltem Testen. Wenn Personen an einem oder mehreren Standorten testen, die nicht Mitarbeiter des Projektteams sind und nicht an seinem Standort arbeiten, spricht man von Outsourcing. Wenn Personen testen, die nicht Mitarbeiter des Projektteams sind, aber am selben Standort arbeiten, dann spricht man von Insourcing.

Alle drei Arten der Testorganisation brauchen klare Kommunikationswege und gut definierte Erwartungen an Ziel, Aufgaben und Arbeitsergebnisse. Das Projektteam kann sich dabei kaum auf informelle Kommunikationswege verlassen, wie Gespräche unter Kollegen im Flur, oder auf außerbetriebliche soziale Kontakte. Unterschiedliche Standorte, Zeitzonen, kulturelle und sprachliche Verschiedenheit machen die Zusammenarbeit kritisch.

Alle drei Arten der Testorganisation müssen auch ihre Methodiken angleichen. Wenn zwei Testteams verschiedene Methoden anwenden, oder wenn das Testteam eine andere Methode als die Entwicklungsabteilung oder die Projektleitung benutzt, führt das besonders bei der Testdurchführung zu gravierenden Problemen.

Beim Testen an verschiedenen Standorten muss die Aufteilung explizit und intelligent festgelegt sein. Auch die kompetenteste und hoch qualifizierte Gruppe kann ohne solche Vorgaben die Testarbeit nicht erfolgreich durchführen. Die Testarbeit wird Lücken (mit steigendem Restrisiko für die Qualität bei Auslieferung) und Überlappungen haben (was die Effizienz verringert).

Letztlich ist für alle drei Arten der Testorganisation entscheidend, dass das ganze Projektteam das Vertrauen entwickelt und behält, dass jedes Testteam seine Rollen richtig ausführt, auch angesichts von organisatorischen, kulturellen, sprachlichen und geographischen Grenzen. Ein Mangel an Vertrauen kann dazu führen, dass durchgeführte Aktivitäten überdurchschnittlich kontrolliert werden und kann die Entwicklung von effektiven Testmannschaften bremsen. Dabei entstehen Ineffizienzen und mögliche Schwierigkeiten bei der Einhaltung von Plänen.

## 3.9 Risikoorientiertes Testen

### *3.9.1 Einführung in das risikoorientierte Testen*

Risiko ist die Möglichkeit, dass es zu einem unerwünschten Ergebnis kommt. Ein Risiko besteht grundsätzlich immer, wenn Probleme auftauchen, die die Produktqualität oder den Projekterfolg mindern könnten, wie sie von Kunden, Anwendern, Beteiligten oder Betroffenen wahrgenommen werden.

Wenn sich das potenzielle Problem in erster Linie auf die Produktqualität auswirkt, dann bezeichnet man derartige Probleme als Produktrisiken (oder Qualitätsrisiken). Ein Beispiel hierfür sind Fehler aufgrund mangelnder Zuverlässigkeit, die einen Systemabsturz während des normalen Betriebs verursachen könnten. Wenn sich das potenzielle Problem dagegen vor allem auf den Projekterfolg auswirkt, dann bezeichnet man derartige Probleme als Projektrisiken (oder Planungsrisiken). Ein Beispiel hierfür wäre eine Personalknappheit, die den Abschluss des Projekts verzögert.

Nicht alle Risiken geben gleich viel Anlass zu Besorgnis. Unterschiedliche Faktoren beeinflussen die Risikostufe:

- die Wahrscheinlichkeit, dass das Problem auftreten wird
- die Auswirkungen, die das Problem haben wird, falls es auftritt.

Beim risikoorientierten Testen begegnet man dem Risiko auf drei verschiedene Arten:

- Der Aufwand muss der Risikostufe der identifizierten Produkt- oder Qualitätsrisiken angemessen sein. Es sind Umfang des Testaufwands, Wahl der Testverfahren, Ablauf der Testaktivitäten und Beheben der Fehlerzustände zu beachten.
- Planung und Management der Testaktivitäten müssen geeignet sein, jedes wesentliche identifizierte Projekt- oder Planungsrisiko zu beherrschen und auch auf unerwartete Ereignisse angemessen zu reagieren.
- Testergebnisse und Projektstatus in den Berichten müssen auf die Restrisiken verweisen; wenn beispielsweise Tests noch nicht durchgeführt oder ausgelassen wurden, oder wenn Fehler noch nicht behoben oder noch nicht nachgetestet wurden.

Tester sollten dem Risiko mit diesen drei Reaktionen nicht nur zu Beginn und Ende des Projekts, sondern während des gesamten Softwarelebenszyklus begegnen. Im Verlauf des Projekts sollten sie vor allem

- dadurch die Risiken reduzieren, dass sie die wichtigsten Fehler (bei Produktrisiken) entdecken, und dass sie geeignete Maßnahmen zur Risikobeherrschung und Vorkehrung gegen Risiken so umsetzen, wie in der Teststrategie und im Mastertestkonzept vorgesehen; und
- Risiken so bewerten, dass sie Wahrscheinlichkeit oder Auswirkungen der bereits identifizierten und analysierten Risiken so erhöhen oder vermindern, wie es Informationen und Erkenntnisse aus dem Projektverlauf nahe legen.

In beiden Fällen beeinflussen die Maßnahmen, wie das Testen auf das Risiko reagiert.

Risikoorientiertes Testen hat vieles gemeinsam mit einer Versicherung. Man kauft dann eine Versicherung, wenn man sich Sorgen über ein potentielles Risiko macht, und man ignoriert Risiken, über die man sich keine Sorgen macht. Quantitative Analysen, wie sie im Versicherungsgeschäft üblich sind, können anwendbar sein, beim risikoorientierten Testen verlassen die Tester sich aber meist auf qualitative Analysen.

Für fachgerechtes risikoorientiertes Testen sollten die Tester typische Produkt- und Projektrisiken identifizieren, analysieren und beherrschen können, sei es im Zusammenhang mit der Betriebssicherheit, geschäftlichen und wirtschaftlichen Aspekten, System- und Datensicherheit sowie mit technischen und geschäftspolitischen Faktoren.

## *3.9.2 Risikomanagement*

Das Risikomanagement besteht aus drei Hauptaktivitäten:

1. Risikoidentifikation
2. Risikoanalyse – und bewertung
3. Risikobeherrschung (auch als Risikosteuerung bezeichnet)

An sich folgen diese Aktivitäten aufeinander. Weil aber ein kontinuierliche Risikomanagement notwendig ist, wie in den vorigen und nachfolgenden Abschnitten erwähnt, sollten alle drei Arten von Risikomanagement iterativ fast während des ganzen Projekts eingesetzt werden.

Risikomanagement ist am effektivsten, wenn alle Betroffenen im Projekt daran beteiligt werden. Manchmal werden Betroffene auch vertreten. Bei der Entwicklung von Standard-Software für den Massenmarkt kann es beispielsweise eine kleine Gruppe möglicher Kunden sein, die dabei hilft, die Fehler auszumachen, die ihre Nutzung der Software am meisten beeinträchtigen würden. Diese Gruppe möglicher Kunden steht dann für den gesamten möglichen Kundenkreis.

Wegen ihres speziellen Fachwissens sollten die Test Analystes aktiv an Risikoidentifikation und Risikoanalyse beteiligt werden.

### 3.9.2.1 Risikoidentifikation

Tester können sowohl die Produktrisiken als auch die Projektrisiken durch ein oder mehrere der folgenden Verfahren identifizieren:

- Experten-Interviews
- unabhängige Bewertungen
- Verwendung von Risiko-Vorlagen
- Erfahrungen aus dem Testen (beispielsweise Besprechungen beim Projektabschluss)
- Risiko-Workshops (beispielsweise mit Fehler-Möglichkeits- und Einfluss-Analyse (FMEA), siehe Abschnitt 3.10)
- Brainstorming
- Checklisten
- Erfahrungen aus der Vergangenheit

Je breiter die Basis der Betroffenen im Risikoidentifikations-Prozess ist, desto höher die Wahrscheinlichkeit, dass dabei die größtmögliche Menge an wichtigen Risiken aufgedeckt wird.

Bei manchen Vorgehensweisen zur Risikoidentifikation endet der Prozess schon mit Identifizierung des eigentlichen Risikos.

Andere Verfahren, beispielsweise die Fehler-Möglichkeits- und Einfluss-Analyse (FMEA), gehen weiter. Hier müssen für jede potenzielle Fehlerwirkung die Auswirkungen auf das restliche System (einschließlich übergeordneter Systeme bei Multisystemen) und auf mögliche Anwender des Systems identifiziert werden.

Wieder andere Verfahren, beispielsweise die Gefährlichkeitsanalyse, erfordern eine Annahme über die Risikoquelle.

Weitere Informationen zu Fehler-Möglichkeits- und Einfluss-Analyse (FMEA) sowie Software-Fehlermöglichkeits-, Einfluss- und Kritikalitäts-Analyse (FMECA) enthalten Abschnitt 3.10 sowie [Stamatis95], [Black02], [Craig02], [Gerrard02].

### 3.9.2.2 Risikoanalyse und -bewertung

Während es bei der Risikoidentifikation darum geht, möglichst viele vorhandene Risiken zu identifizieren, befasst sich die Risikoanalyse mit der Untersuchung der identifizierten Risiken. Sie kategorisiert das Risiko und legt Eintrittswahrscheinlichkeit und -wirkungen fest.

Risiken zu kategorisieren bedeutet eine Einteilung in Risikotypen. ISO Standard 9126 für Qualitätsmerkmale behandelt typische Qualitätsrisiken, die eine Klassifizierung von Risiken unterstützen. Manche Organisationen arbeiten mit eigenen Qualitätsmerkmalen. Hinweis: Bei einer Risikoidentifikation auf Basis von Checklisten wird oft beim Identifizieren das Risiko einem bestimmten Risikotyp zugeordnet.

Für die Festlegung der Risikostufe werden für jedes einzelne Risiko die Eintrittswahrscheinlichkeit und die Auswirkung (Schadenshöhe, damit verbundener Schaden) eingeschätzt. Eintrittswahrscheinlichkeit (Wahrscheinlichkeit des Auftretens, Auftrittswahrscheinlichkeit) bezeichnet oft die Wahrscheinlichkeit, dass das potenzielle Problem im getesteten System vorhanden ist. Es geht also um ein technisches Risiko.

Folgende Faktoren beeinflussen das technische Risiko:

- Komplexität der Technologie und des Teams
- Ausbildung und Erfahrung der Testbeteiligten (z.B. Mitarbeiter von Fachbereichen, Systementwickler, Programmierer
- Konflikte im Team
- vertragliche Probleme mit Zulieferern

- räumlich verteilte Entwicklungsorganisation
- Altsysteme versus neue Herangehensweisen
- Werkzeuge und Technologie
- schlechte organisatorische oder technische Leitung
- Zeitdruck, knappe Ressourcen, Druck durch das Management
- Fehlen eines bisherigen Qualitätsmanagements
- häufige Änderungen
- hohe bisherige Fehlerraten
- Schnittstellen-/Integrationsproblematik

Die Auswirkung bezeichnet oft das Ausmaß der Wirkung auf Anwender, Kunden oder andere Betroffene. Es geht also um ein Geschäftsrisiko.

Folgende Faktoren beeinflussen das geschäftliche Risiko:

- Häufigkeit der Nutzung bei einer betroffenen Funktion
- Schaden für die Reputation
- entgangene Geschäfte
- mögliche finanzielle, ökologische oder soziale Verluste oder Haftungsansprüche
- zivil- oder strafrechtliche Maßnahmen
- Aberkennung der Lizenz
- fehlende vernünftige Lösungsalternativen
- das negative Erscheinungsbild, wenn Mängel bekannt werden

Die Risikostufe lässt sich entweder quantitativ oder qualitativ betrachten. Wenn Risikowahrscheinlichkeit und –auswirkungen quantitativ bestimmt werden können, ergibt ein einfaches Multiplizieren der beiden die Kosten einer Gefährdung, also den erwarteten Verlust bei einem bestimmten Risiko.

In der Regel lässt die Risikostufe sich aber nur qualitativ bestimmen. Dabei kann die Wahrscheinlichkeit zwar mit sehr hoch, hoch, mittel, gering oder sehr gering angegeben werden; aber es lässt sich nicht genau sagen, ob sie bei 90%, 75%, 50%, 25%, oder 10% liegt. Trotzdem ist diese Vorgehensweise nicht weniger wertvoll als die quantitative Methode. Wenn das quantitative Vorgehen nicht fachgerecht eingesetzt wird; könnte es vielmehr die Betroffenen darüber täuschen, inwieweit Risiko sich tatsächlich verstehen und managen lässt. Informelle Vorgehensweisen, wie in [vanVeenendaal02], [Craig02] und [Black07b] beschrieben, sind oft qualitativ und weniger rigoros.

Wenn die Risikoanalyse nicht auf umfangreichen und statistisch korrekten Risikodaten beruht, wie im Versicherungsbereich, dann wird sie, ob quantitativ oder qualitativ, immer auf der jeweiligen Wahrnehmung von Risikowahrscheinlichkeit und -auswirkungen basieren. Persönliche Wahrnehmung und Auffassungen über die Faktoren werden die Einstufung des Risikos beeinflussen. Projektmanager, Programmierer, Anwender, Fachanalytiker, Systemarchitekten und Tester haben unterschiedliche Wahrnehmungen und deshalb vielleicht auch ganz andere Ansichten über die Risikostufe des jeweiligen Risikos. Die Risikoanalyse sollte deshalb Wege vorsehen, wie Konsens zu erreichen ist, oder – im ungünstigsten Fall – die gültige Risikostufe anzuordnen ist. Nur dann können die Risikostufen eine Richtschnur für die Aktivitäten zur Risikobeherrschung bieten.

#### 3.9.2.3 Risikobeherrschung[2]

Nachdem ein Risiko identifiziert und analysiert wurde, gibt es vier Möglichkeiten damit umzugehen:

[2] Gebräuchliche Synnonyme in diesem Kontext: Risikovermeidung, Risikosteuerung und –bewältigung, Risikoüberwachung

1. das Risiko durch vorbeugende Maßnahmen beherrschen, um Risikowahrscheinlichkeit und/oder –auswirkungen zu minimieren
2. Notfallpläne, um die möglichen Auswirkungen bei einem Auftreten des Risikos zu reduzieren
3. das Risiko an eine andere Partei auslagern
4. das Risiko akzeptieren und nicht weiter beachten

Welche dieser Optionen tatsächlich gewählt wird, hängt sowohl von den Vorteilen und Chancen jeder Option, als auch von ihren Kosten und möglichen Folgerisiken ab.

**Strategien zur Risikobeherrschung**

Grundlage des Mastertestkonzepts und anderer Testpläne bei den meisten risikoorientierten Teststrategien sind Risikoidentifizierung, Risikoanalyse und das Festlegen von Aktivitäten zur Risikobeherrschung. Die Risikostufe für das jeweilige Risiko entscheidet über den Umfang des Testaufwands, der zur Risikobeherrschung verwendet wird. An der Sicherheit orientierte Standards, wie FAA DO-178B/ED 12B oder IEC 61508, schreiben Testverfahren und Grad der Überdeckung je nach Risikostufe vor.

**Beherrschung von Projektrisiken**

Wenn Projektrisiken identifiziert wurden, müssen die Projektmanager eventuell darüber informiert werden und handeln. Nicht alle Risiken lassen sich innerhalb der Testorganisation durch entsprechende Maßnahmen reduzieren. Es gibt aber auch Projektrisiken, die die Testmanager erfolgreich steuern können:

- einsatzbereite Testumgebung und Werkzeuge
- Verfügbarkeit und Qualifikation des Testpersonals
- schlechte Testbasis
- zu hohe Änderungsrate an der Testbasis
- fehlende Standards, Regeln und vorgeschriebene Verfahren für das Testen

Zum Vorgehen bei der Risikobeherrschung gehören eine frühe Vorbereitung der Testmittel, Vorabtests der Testausstattung, Vorabtests früherer Produktversionen, strengere Testeingangsbedingungen, Anforderungen an die Testbarkeit, Teilnehmen an Reviews früherer Projektergebnisse, Teilnehmen am Problem- und Änderungsmanagement, Überwachung des Projektfortschritts und der Qualität.

**Beherrschung von Produktrisiken**

Das Testen ist eine Art der Risikobeherrschung für Produkt- und Qualitätsrisiken. Wenn die Tester Fehler finden, reduzieren sie das Risiko, weil sie das Bewusstsein für vorhandene Fehler schärfen und die Möglichkeit schaffen, sie vor Release des Systems zu beheben. Wenn die Tester keine Fehler finden, reduzieren sie beim Testen das Risiko, denn sie stellen sicher, dass das System unter bestimmten (den getesteten) Bedingungen richtig funktioniert.

Produkt- oder Qualitätsrisiken lassen sich auch durch Aktivitäten steuern, die nicht im eigentlichen Sinne Testaktivitäten sind. Wird beispielsweise festgestellt, dass die Anforderungen nicht gut spezifiziert sind, dann sind gründliche Reviews sicher ein geeignetes Mittel zur Risikobeherrschung. Sie sind weitaus effizienter als das Entwerfen und Priorisieren von Tests, die erst dann durchgeführt werden, wenn eine schlecht spezifizierte Software entwickelt und in Code umgesetzt wurde.

Die Risikostufe wird auch für die Priorisierung der Tests verwendet. Manchmal werden alle hohen Risikostufen vor den niedrigeren Risikostufen getestet, und die Tests erfolgen streng nach der Reihenfolge der Risikos (Depth-first, d.h. Testen in die Tiefe). In anderen Fällen machen die Tester Stichproben von identifizierten Risiken aller Risikostufen, gewichten die Risiken und wählen Tests aus, wobei sie dafür sorgen, dass jedes Risiko mindestens einmal abgedeckt wird (Breadth-first, d.h. Testen in die Breite).

Weitere Aspekte der Risikobeherrschung sind

- die Auswahl eines geeigneten Testentwurfsverfahrens
- Reviews und Inspektionen
- Review des Testentwurfs
- Unabhängigkeitsgrad der Testorganisation
- die erfahrenste Person wird eingesetzt
- wie Fehlernachtests durchgeführt werden
- Regressionstests

In [Gerrard02] wird das Konzept der Testeffektivität als (prozentualer) Indikator dafür eingeführt, wie effektiv das Testen für die Risikobeherrschung eingeschätzt wird. Demnach würde man das Testen nicht als eine Maßnahme zur Risikobeherrschung einsetzen, wenn es nur wenig effektiv ist.

Unabhängig davon, ob beim risikoorientierten Testen in die Tiefe oder in die Breite getestet wird, ist es möglich, dass die Zeit für das Testen abläuft, ohne dass alle Tests durchgeführt wurden. Beim risikoorientierten Testen können die Tester in solchen Fällen das Management über das verbleibende Restrisiko informieren, und das Management kann entscheiden, ob dieses Restrisiko an die Anwender, Kunden, Helpdesks oder technische Unterstützung und/oder an das Bedienpersonal weitergegeben wird.

**Anpassung des Testens an weitere Testzyklen**

Wenn noch Zeit für weitere Tests zur Verfügung steht, dann sollten zusätzliche Testzyklen auf Basis einer neuen Risikoanalyse erfolgen. Wichtige Faktoren dafür sind mögliche neue oder deutlich veränderte Produktrisiken, instabile oder fehleranfällige Bereiche des Systems, die im Laufe des Testens identifiziert wurden, Risiken als Folge behobener Fehler, eine Konzentration auf Fehler, die sich beim Testen als typische Fehler herausgestellt haben, sowie nicht ausreichend getestete Bereiche des Systems (Bereiche mit niedriger Testüberdeckung). Die Planung neuer oder zusätzlicher Testzyklen sollte auf einer Analyse derartiger Risiken basieren. Es ist außerdem sehr empfehlenswert, zu jedem Meilenstein eine aktualisierte Risikoeinschätzung zu erstellen.

### *3.9.3 Risikomanagement im Softwarelebenszyklus*

Im Idealfall begleitet das Risikomanagement den gesamten Softwarelebenszyklus. Gibt es eine Testrichtlinie und/oder eine Teststrategie in einer Organisation, dann sollten sie den grundlegenden Prozess beschreiben, nach dem Produkt- und Projektrisiken im Testen behandelt werden. Sie sollten zeigen, dass Risikomanagement integraler Bestandteil aller Teststufen ist, und wie es sie beeinflusst.

Die Aktivitäten zur Risikoidentifikation und –analyse können während des Anfangsstadiums eines Projekts beginnen, unabhängig vom Lebenszyklusmodell der Softwareentwicklung. Maßnahmen zur Risikobeherrschung lassen sich als Teil des gesamten Testplanungs-Prozesses planen und umsetzen. Im Mastertestkonzept und/oder in den Teststufenplänen können sowohl Projekt- als auch Produktrisiken berücksichtigt werden. Art des Risikos und Risikostufe beeinflussen dann die Teststufen für das Risiko, den Umfang des zur Risikobeherrschung verwendeten Testaufwands, Test- und andere –verfahren zur Risikobeherrschung, und die Kriterien, nach denen beurteilt wird, ob die Maßnahmen zur Risikobeherrschung ausreichend waren.

Nachdem die Planungsphase abgeschlossen ist, sollte das Risikomanagement einschließlich Risikoidentifikation, -analyse und –reduzierung ein fortlaufender Prozess während des gesamten Projekts sein. Es schließt die Identifikation neuer Risiken, erneutes Bewerten der Risikostufen für bestehende Risiken und die Bewertung ein, ob Risikobeherrschungsmaßnahmen effektiv waren. Ein Beispiel: Wurden während der Anforderungsanalyse Risikoidentifikation und –analyse auf Basis der Anforderungsspezifikation durchgeführt, dann sollten die Risiken nach Vorliegen des Systementwurfs neu bewertet werden. Ein weiteres Beispiel: Falls beim Testen weit mehr Fehler in einer Komponente

gefunden wurden als erwartet, dann kann man annehmen, dass die Fehlerwahrscheinlichkeit in diesem Bereich höher ist als angenommen und Wahrscheinlichkeit und Risikostufe nach oben korrigieren. Für diese Komponente könnte das zu einem erhöhten Testaufwand führen.

Nachdem die Risikoidentifikation und –analyse abgeschlossen sind und Maßnahmen zur Risikobeherrschung greifen, ist messbar, wie weit die Risiken reduziert wurden. Dazu werden die Testfälle und aufgedeckten Fehlerzustände auf die dazugehörigen Risiken zurückverfolgt. Die Tester können während der Tests und beim Aufdecken von Fehlern das Restrisiko feststellen. Damit unterstützt das risikoorientierte Testen bei der Festlegung des richtigen Zeitpunkts für die Freigabe. [Black03] enthält ein Beispiel für die Berichterstattung von Testergebnissen basierend auf der Risikoüberdeckung.

Aus den Testberichten sollten kontrollierte und noch offene Risiken, sowie erzielter und noch ausstehender Nutzen hervorgehen.

## 3.10 FMEA (Fehler-Möglichkeits- und Einfluss-Analyse)

Die Fehlermöglichkeits- und Einfluss-Analyse (FMEA, Failure Mode and Effects Analysis) sowie die Variante Fehlermöglichkeits-, Einfluss- und Kritikalitäts-Analyse (FMECA, Failure Mode, Effects and Criticality Analysis), die eine Kritikalitätsanalyse einschließt, sind iterative Aktivitäten zur Analyse von Auswirkung und Kritikalität der Fehlerzustände im System. Wenn Software damit analysiert wird, heißen sie auch SFMEA und SFMECA, wobei das S für Software steht. Die Informationen in den folgenden Abschnitten gelten auch für die drei anderen Methoden, es wird aber immer die Abkürzung FMEA benutzt.

Tester sollen zum Erstellen von FMEA-Dokumenten beitragen können. Dazu müssen sie Zweck und Anwendung dieser Dokumente verstehen und ihr Fachwissen zur Bestimmung von Risikofaktoren einbringen.

### *3.10.1 Anwendungsbereiche*

Die FMEA sollte angewendet werden,

- wenn die Kritikalität der Software oder des Systems analysiert werden muss, um das Fehlerrisiko zu reduzieren (beispielsweise bei sicherheitskritischen Systemen wie Flugzeug-Steuerungssystemen).
- wenn obligatorische oder gesetzliche Anforderungen gelten (siehe Abschnitt 1.3.2 Sicherheitskritische Systeme).
- um Fehler in einem möglichst frühen Stadium zu beseitigen.
- um für sicherheitskritische Systeme spezielle Erwägungen beim Testen, Einschränkungen beim Betrieb und designrelevante Entscheidungen zu definieren.

### *3.10.2 FMEA durchführen*

Eine FMEA sollte eingeplant werden, sobald vorläufige Informationen auf übergeordneter Ebene verfügbar sind, und, sobald Details verfügbar werden, auf niedrigere Ebenen erweitert werden. Die FMEA kann auf jeder System- oder Softwareebene angewendet werden, je nach verfügbaren Informationen und Programmanforderungen.

Dabei werden iterativ für jede kritische Funktion, Modul oder Komponente

- eine Funktion ausgewählt und ihre Fehlermöglichkeiten festgestellt, beispielsweise, wie sie fehlschlagen könnte

- mögliche Ursachen für die Fehlerwirkung definiert und die Folgen dargestellt; und Mechanismen zur Reduktion oder Beherrschung der Fehlerwirkungen entworfen.

### 3.10.3 Kosten und Nutzen

FMEA bietet folgende Vorteile:

- Erwartete Systemausfälle durch Fehlerwirkungen der Softwareausfälle oder Anwendungsfehler lassen sich aufdecken.
- Wird sie systematisch eingesetzt, kann sie zur Analyse auf Ebene des Gesamtsystems beitragen.
- Ergebnisse können zu Entwurfsentscheidungen und/oder –begründungen beitragen.
- Ergebnisse können verwendet werden, um bestimmte (kritische) Bereiche der Software besonders intensiv zu testen.

Folgende Aspekte sind bei der Anwendung der FMEA zu berücksichtigen:

- Mögliche Folgefehler werden selten beachtet.
- Es kann viel Zeit kosten, die FMEA-Tabellen zu erstellen.
- Es kann schwierig sein, unabhängige Funktionen zu definieren.
- Es kann schwierig sein, die Fehlerfortpflanzung zu identifizieren.

## 3.11 Besonderheiten beim Testmanagement

### 3.11.1 Testmanagement beim explorativen Testen

Session-based Testmanagement (SBTM) ist ein Managementkonzept für exploratives Testen. Eine Session (Testsitzung) ist die grundlegende Testeinheit; die Tester konzentrieren sich dabei mit einem bestimmten Testziel (Test-Charta) ohne Unterbrechung auf ein spezifisches Testobjekt. Am Ende jede einzelnen Testeinheit erstellen sie einen Bericht über die in der Testsitzung durchgeführten Aktivitäten (Session Sheet). SBTM arbeitet mit einer strukturierten Testdokumentation und erzeugt Protokolle, die die Verifizierungsdokumentation ergänzen.

Eine Testeinheit lässt sich in drei Stufen unterteilen:

- Testeinheit aufbauen: Einrichten der Testumgebung und besseres Kennenlernen des Produkts
- Test entwerfen und durchführen: Untersuchen des Testobjekts und Problemsuche
- Fehleranalyse und Bericht: Sie beginnen, sobald ein Tester auf etwas stößt, das eine Fehlerwirkung sein könnte.

Das SBTM Session Sheet besteht aus:

- Test-Charta
- Name des oder der Tester
- Datum und Uhrzeit des Beginns
- Aufteilen der durchgeführten Aufgaben in Testeinheiten oder Testsitzungen
- Testdatendateien
- Testnotizen
- Probleme
- Fehlerwirkungen

Am Ende jeder Testsitzung hält der Testmanager eine Abschlussbesprechung mit dem Testteam. Dabei findet ein Review der Session Sheets statt, die Chartas werden verbessert, das Testteam teilt seine Eindrücke mit und der Manager schätzt und plant weitere Testsitzungen.

Die Tagesordnung für eine solche Abschlussbesprechung lässt sich mit dem englischen Begriff PROOF abkürzen:

- **P**ast: Was passierte in der abgelaufenen Testsitzung?
- **R**esults: Welche Ergebnisse wurden erzielt?
- **O**utlook: Vorschau; was ist noch zu tun?
- **O**bstacles: Welche Hürden für gutes Testen gab es?
- **F**eelings: Was denken oder fühlen die Tester in diesem Zusammenhang?

### 3.11.2 *Testmanagement bei Multisystemen*

Zum Testmanagement von Multisystemen gehören folgende Aspekte:

- Testmanagement ist komplexer, denn möglicherweise werden die einzelnen Systeme, aus denen das Multisystem besteht, an unterschiedlichen Standorten getestet, von unterschiedlichen Organisationen und mit unterschiedlichen Lebenszyklusmodellen. Deshalb sieht das Mastertestkonzept für ein Multisystem in der Regel ein formelles Lebenszyklusmodell vor und legt besonderen Wert auf Managementaspekte, wie Meilensteine oder Quality-Gates. Oft gibt es einen formalen Qualitätssicherungs-Prozess, der in einem eigenen Qualitätsplan definiert ist.
- Unterstützende Prozesse, wie Konfigurationsmanagement, Änderungs- und Release-Management müssen formal definiert und die Schnittstellen zum Testmanagement vereinbart werden. Diese Prozesse sind unverzichtbar, damit die Softwarelieferungen kontrolliert erfolgen, Änderungen systematisch eingebracht werden und die Referenzkonfiguration der zu testenden Anwendungen richtig definiert ist.
- Es kann eine große technische und organisatorische Herausforderung sein, repräsentative Testumgebungen einschließlich der Testdaten zu erstellen und verwalten.
- Möglicherweise müssen Simulatoren für die Teststrategie beim Integrationstest entwickelt werden. Für frühe Teststufen kann das relativ einfach und kostengünstig sein; Simulatoren für ganze Systeme und die übergeordneten Integrationstests bei Multisystemen können aber komplex und teuer werden. Deshalb finden Planung, Kostenschätzung und Entwicklung von Simulatoren oft in einem separaten Projekt statt.
- Abhängigkeiten zwischen den Teilsystemen erzeugen zusätzliche Restriktionen für System- und Abnahmetests; Systemintegrationstests und die zugehörigen Testbasisdokumente (beispielsweise Schnittstellenspezifikationen) gewinnen an Bedeutung.

### 3.11.3 *Testmanagement bei sicherheitskritischen Systemen*

Aspekte beim Testmanagement von sicherheitskritischen Systemen:

- Normalerweise gelten die (industrie-)spezifischen Standards der jeweiligen Branche (beispielsweise Transport, Medizintechnik, Militär). Sie können den Entwicklungsprozess und den organisatorischen Aufbau betreffen, oder das Produkt, das entwickelt wird. Weitere Informationen enthält Kapitel 6.
- Haftung ist bei sicherheitskritischen Systemen oft ein wichtiges Thema. Deshalb können für die Konformität (den Nachweis, dass die Auflagen erfüllt sind) formale Aspekte nötig sein, wie Rückverfolgbarkeit der Anforderungen, Erreichen eines bestimmten Testüberdeckungsgrads, Erfüllen von Abnahmekriterien und vorgeschriebene Testdokumentation.

- Konformität bei Organisationsstruktur und Entwicklungsprozess lassen sich möglicherweise durch Organisationsdiagramme und Audits nachweisen.
- Es wird ein definierter Entwicklungslebenszyklus zu Grunde gelegt, je nach vorgeschriebenem Standard. Die Lebenszyklen sind in der Regel sequenziell.
- Falls das System von einer Organisation als kritisch eingestuft wurde, müssen die folgenden nicht-funktionalen Eigenschaften in der Teststrategie und/oder im Testkonzept berücksichtigt werden:
  - Zuverlässigkeit (Reliability)
  - Verfügbarkeit (Availability)
  - Wartbarkeit (Maintainability)
  - Sicherheit des Betriebs/gegenüber Angriffen (Safety/security)

  Aufgrund dieser Eigenschaften bezeichnet man solche Systeme auch als RAMS-Systeme (Abkürzung aus den Anfangsbuchstaben der englischen Begriffe).

## 3.11.4 *Sonstige Besonderheiten beim Testmanagement*

Wenn nicht-funktionale Tests nicht eingeplant wurden, kann das ein erhebliches Risiko für den Erfolg einer Anwendung bedeuten. Andererseits sind viele Arten nicht-funktionaler Tests mit erheblichen Kosten verbunden, deshalb sind Kosten und Risiko gegeneinander abzuwägen.

Es gibt viele unterschiedliche nicht-funktionale Tests. Nicht alle sind unbedingt für eine bestimmte Anwendung geeignet.

Folgende Faktoren können die Planung und Durchführung nicht-funktionaler Tests beeinflussen:

- Anforderungen der Betroffenen
- benötigte Werkzeuge
- benötigte Hardware
- organisatorische Faktoren
- Kommunikation
- Sicherheit der Daten

### 3.11.4.1 Anforderungen der Betroffenen

Nicht-funktionale Anforderungen sind oft unbekannt oder kaum spezifiziert. Die Tester müssen deshalb in der Planungsphase die Erwartungshaltungen der Betroffenen sondieren und daraufhin bewerten, welche Risiken sie für das Projekt bedeuten.

Beim Ermitteln der Anforderungen sollten verschiedene Perspektiven berücksichtigt werden. Sie müssen die Anforderungen beispielsweise von Kunden, Anwendern, Bedien- und Wartungspersonal einholen, sonst übersehen sie möglicherweise Anforderungen.

Folgende Aspekte sind wesentlich für die Verbesserung der Testbarkeit nicht-funktionaler Anforderungen:

- Der Aufwand für die Spezifikation testbarer Anforderungen ist fast immer kosteneffektiv. Wichtig sind dabei eine einfache Ausdrucksweise und konsistente und richtige Terminologie, die mit dem Projekt-Wörterbuch/Glossar übereinstimmt. Bestimmte Formulierungen sind einzuhalten: muss (ist Pflicht), sollte (ist wünschenswert) und soll (kann als Synonym für muss verwendet werden, am besten vermeidet man es aber).
- Leserinnen und Leser der Anforderungen haben unterschiedliche Hintergründe.
- Anforderungen sind klar, deutlich und unmissverständlich zu formulieren, es sollte ein Standardformat für die Anforderungen verwendet werden.
- Wann immer möglich, sollten die Tester Anforderungen quantitativ spezifizieren. Dabei ist zu entscheiden, welche Metrik für ein bestimmtes Merkmal am besten geeignet ist

(beispielsweise Millisekunden für die Messung von Leistung), und innerhalb welches Wertebereichs die Ergebnisse akzeptiert oder nicht akzeptiert werden. Für bestimmte nicht-funktionale Eigenschaften ist das nicht einfach (beispielsweise Benutzbarkeit).

#### 3.11.4.2 Benötigte Werkzeuge

Kommerzielle Werkzeuge oder Simulatoren sind besonders relevant für das Messen von Leistung und Effizienz sowie für einige Sicherheitstests. Die Testplanung sollte die geschätzten Kosten und Vorlaufzeiten für die benötigten Werkzeuge berücksichtigen. Wenn Spezialwerkzeuge zum Einsatz kommen, sind damit verbundene Lernkurven oder die Kosten für den Einsatz externer Spezialisten zu berücksichtigen.

Einen komplexen Simulator zu entwickeln, kann zu einem eigenen Entwicklungsprojekt werden, es sollte dann auch entsprechend geplant werden. Wenn Simulatoren für sicherheitskritische Anwendungen eingesetzt werden sollen, sind auch die entsprechenden Abnahmetests und eine etwaige Zertifizierung des Simulators durch eine unabhängige Instanz zu planen.

#### 3.11.4.3 Benötigte Hardware

Für viele nicht-funktionale Tests wird eine produktionsähnliche Testumgebung benötigt, um realistische Messungen zu ermöglichen. Je nach Größe und Komplexität des zu testenden Systems kann das Planung und Finanzierung der Tests maßgeblich beeinflussen. Kosten für das Durchführen nicht-funktionaler Tests können so hoch sein, dass nur begrenzt Zeit dafür zur Verfügung steht.

Will man beispielsweise die Anforderungen an die Skalierbarkeit einer stark frequentierten Internetseite verifizieren, kann dafür eine simulierte Nutzung durch Hunderttausende virtuelle Nutzer nötig sein. Das würde die Hardware- und Werkzeugkosten erheblich beeinflussen. Da derartige Kosten in der Regel durch Mietlizenzen für die benötigten Lasttestwerkzeuge minimiert werden (Aufstockung von Lizenzen), ist die verfügbare Zeit für solche Tests begrenzt.

Benutzbarkeitstests setzen möglicherweise eigens eingerichtete Versuchslabors oder zahlreiche Fragebögen voraus. Normalerweise werden diese Tests nur einmal im Entwicklungslebenszyklus eines Systems durchgeführt.

Andere nicht-funktionale Tests (beispielsweise Sicherheitstests, Performanztests) müssen in einer produktionsähnlichen Testumgebung durchgeführt werden. Da die Kosten für solche Testumgebungen hoch sein können, ist die Nutzung der echten Produktionsumgebung oft die einzig praktikable Alternative. Die Durchführung der Tests muss dann allerdings sorgfältig geplant werden, sie können wahrscheinlich nur zu bestimmten Zeiten (beispielsweise nachts) stattfinden.

Wenn Effizienztests durchgeführt werden sollen (beispielsweise Performanz oder Last), sind Hardwarekapazitäten und Kommunikationsbandbreiten einzuplanen. Der Bedarf orientiert sich in erster Linie an der Zahl der simulierten virtuellen Nutzer und dem voraussichtlichen Volumen ihres Netzwerkverkehrs. Wird er nicht einbezogen, sind die Messungen nicht repräsentativ.

#### 3.11.4.4 Organisatorische Aspekte

Bei nicht-funktionalen Tests ist oft auch das Verhalten mehrerer Komponenten eines Gesamtsystems zu messen (beispielsweise Server, Datenbanken, Netzwerke). Wenn diese Komponenten auf unterschiedliche Standorte und Organisationen verteilt sind, kann das erheblichen Aufwand für die Planung und Koordinierung der Tests bedeuten. So können bestimmte Softwarekomponenten nur zu bestimmten Uhrzeiten oder nur an bestimmten Tagen im Jahr für den Systemtest zur Verfügung stehen. Organisationen stellen möglicherweise nur eine begrenzte Anzahl von Tagen zur Verfügung, an denen sie das Testen unterstützen. Es kann zu empfindlichen Störungen der geplanten Tests führen, wenn nicht im Vorfeld geklärt worden ist, dass die Systemkomponenten und das Personal der anderen Organisationen für die Testzwecke auf Abruf bereitstehen.

#### 3.11.4.5 Kommunikationsaspekte

Ob Spezifikation und Durchführung bestimmter nicht-funktionaler Tests (vor allem Effizienztests) möglich sind, kann davon abhängen, ob sich bestimmte Kommunikationsprotokolle für die Testzwecke ändern lassen. In der Planungsphase sollte das sichergestellt werden, beispielsweise mit Werkzeugen, die Kompatibilität erzeugen.

#### 3.11.4.6 Sicherheit der Daten

Spezifische Sicherheitsmaßnahmen für ein System sollten bereits in der Testplanungsphase einkalkuliert werden, damit alle vorgesehenen Testaktivitäten tatsächlich möglich sind. Werden beispielsweise die Daten verschlüsselt, kann es schwierig sein, Testdaten zu erzeugen und die Testergebnisse zu verifizieren.

Richtlinien und gesetzliche Bestimmungen zum Datenschutz schließen möglicherweise aus, dass die Tester virtuelle Nutzer auf Basis der Produktionsdaten generieren. Es kann eine schwierige Aufgabe sein, die Testdaten zu anonymisieren, und muss als Teil der Testdurchführung geplant werden.

# 4. Testverfahren

*Begriffe:*

anforderungsbasierter Test, Anweisungsüberdeckungstest, anwendungsfallbasierter Test, Äquivalenzklassenbildung, (einfacher) Bedingungsüberdeckungstest, BS 7925-2, Datenflussanalyse, dynamische Analyse, EE-Pfad, Entscheidungstabellentest, Entscheidungsüberdeckungstest, erfahrungsbasiertes Testverfahren, exploratives Testen, fehlerbasiertes Testverfahren, intuitive Testfallermittlung, fehlerhafter Zeiger („wilder" Zeiger), Fehlertaxonomie, Grenzwertanalyse, Klassifikationsbaumverfahren , Kontrollflussanalyse, LCSAJ, Mehrfachbedingungsüberdeckungstest, Pfadüberdeckungstest, Fehlerangriffe, Speicherengpass, spezifikationsorientiertes Testverfahren, statische Analyse, strukturorientiertes szenarienbasierter Test, Testsitzung, Testverfahren, Test-Charta, Ursache-Wirkungs-Graph, zustandsbasierter Test, Zweigtest

## 4.1 Einführung

Kategorien für die in diesem Kapitel behandelten Testentwurfsverfahren sind

- Spezifikationsorientierte Testverfahren (auch Black-Box-Verfahren)
- Strukturorientierte Testverfahren (auch White-Box-Verfahren)
- Fehlerbasierte Testverfahren
- Erfahrungsbasierte Testverfahren

Die Verfahren ergänzen sich wechselseitig und können unabhängig von der Teststufe je nach Testaktivität eingesetzt werden. Obwohl Test Analysts und Technical Test Analysts jedes der Verfahren anwenden können, unterscheidet dieser Lehrplan nach der üblichen Arbeitsteilung:

- Spezifikationsorientierte Testverfahren → Test Analysts und Technical Test Analysts
- Strukturorientierte Testverfahren → Technical Test Analysts
- Erfahrungsbasierte Testverfahren → Test Analysts und Technical Test Analysts
- Fehlerbasierte Testverfahren → Test Analysts und Technical Test Analysts

Neben diesen Verfahren beschreibt das Kapitel weitere Testverfahren, wie Fehlerangriffe, statische Analyse und dynamische Analyse. Sie werden in der Regel von Technical Test Analysts angewendet.

Hinweis: Spezifikationsorientierte Verfahren können entweder für das fachliche Testen (siehe Abschnitt 5.2) oder das technische Testen (siehe Abschnitt 5.3) eingesetzt werden.

## 4.2 Spezifikationsorientierte Testverfahren

Bei spezifikationsorientierte Testverfahren werden die Testbedingungen und Testfälle aus der Testbasisdokumentation der Komponente oder des Systems abgeleitet und ausgewählt. Die interne Struktur von Komponente oder System bleibt unberücksichtigt.

Gemeinsame Merkmale der spezifikationsorientiertem Testverfahren sind

- Die Spezifikation des zu lösenden Problems, der Software oder ihrer Komponenten beruht auf formellen oder informellen Modellen.
- Aus diesen Modellen können die Testfälle systematisch abgeleitet werden.

Einige Verfahren liefern Kriterien für den Überdeckungsgrad als Maß für die Testentwurfsaufgabe. Wenn der Überdeckungsgrad vollständig ist, heißt das nicht, dass die Menge von Tests vollständig ist.

Es bedeutet vielmehr, dass das Modell für das vorliegende Testverfahren keine weiteren nützlichen Tests bietet.

Spezifikationsorientierte Tests basieren oft auf Anforderungen. Da die Anforderungsspezifikation das Systemverhalten vorgeben sollte, vor allem hinsichtlich seiner Funktionalität, lassen sich aus den spezifizierten Anforderungen Tests ableiten, die das Verhalten des Systems prüfen. Für die oben in diesem Abschnitt erwähnten Testverfahren lässt sich das Modell für das Systemverhalten aus dem Text und den Grafiken der Anforderungsspezifikation ableiten. Die Tests werden auf dieser Basis durchgeführt wie oben beschrieben.

Der Advanced Level Lehrplan umfasst die folgenden spezifikationsorientierte Testentwurfsverfahren:

| Name | Verfahren | Kriterien für die Überdeckung |
|---|---|---|
| Äquivalenzklassentest | Beschreibung siehe ISTQB® Foundation-Level-Lehrplan 2007, Abschnitt 4.3.1. | Überdeckte Klassen/Gesamtzahl der Äquivalenzklassen |
| Grenzwertanalyse | Beschreibung siehe ISTQB® Foundation-Level-Lehrplan 2007, Abschnitt 4.3.2.<br>Hinweis: Die Grenzwertanalyse kann mit zwei oder mit drei Werten durchgeführt werden. Die Entscheidung darüber wird sich am Risiko orientieren. | Überdeckte Grenzwerte/Gesamtzahl der Grenzwerte |
| Entscheidungstabellentest und Ursache-Wirkungs-Graph-Analyse | Beschreibung des Entscheidungstabellentests siehe ISTQB® Foundation-Level-Lehrplan 2007, Abschnitt 4.3.3.<br>Beim Entscheidungstabellentest wird jede mögliche Kombination von Bedingungen, Beziehungen und Beschränkungen getestet. Zusätzlich kann auch ein Graph in logischer Notation, der Ursache-Wirkungs-Graph, verwendet werden.<br>Hinweis: Neben dem Testen aller möglichen Kombinationen von Bedingungen sind auch eingeschränkte Tabellen möglich. Die Entscheidung darüber wird sich wahrscheinlich am Risiko orientieren [Copeland03]. | Überdeckte Kombinationen von Bedingungen/Gesamtzahl der Kombinationen von Bedingungen |
| Zustandsbasiertes Testen | Beschreibung siehe ISTQB® Foundation-Level-Lehrplan 2007, Abschnitt 4.3.4. | Für einfache Zustandsübergänge (Transitionen) ist das Überdeckungsmaß der prozentuale Anteil aller gültigen, während des Tests ausgeführten Transitionen. Es wird als 0-Switch-Überdeckung bezeichnet. Für n Transitionen ist das Überdeckungsmaß |

| Name | Verfahren | Kriterien für die Überdeckung |
|---|---|---|
| | | der prozentuale Anteil aller gültigen Folgen von n Transitionen, die während des Tests ausgeführt werden. Es wird als (n-1)-Switch-Überdeckung bezeichnet. |

Klassifikationsbaumverfahren, Testen mit orthogonalen Arrays und paarweises Testen

| | | |
|---|---|---|
| | Es werden zu untersuchende Faktoren oder Variablen und die Optionen oder Werte identifiziert, die sie annehmen können. Die Optionen oder Werte werden dann einzeln, paarweise oder in Kombinationen von drei oder sogar mehr identifiziert. [Copeland03]<br>Das Klassifikationsbaumverfahren setzt eine graphische Notation ein, um die Testbedingungen (-klassen) und ihre Kombinationen für die Testfälle abzubilden. [Grochtmann94] | Kriterien hängen vom verwendeten Verfahren ab, das Überdeckungsmaß des Verfahrens Paarweises Testen unterscheidet sich beispielsweise von dem des Klassifikationsbaumverfahrens. |

Anwendungsfallbasiertes Testen (szenarienbasiertes Testen)

| | |
|---|---|
| Beschreibung siehe ISTQB® Foundation-Level-Lehrplan 2007, Abschnitt 4.3.5. | Es sind keine formalen Kriterien anwendbar. |

Manchmal werden für die Erstellung von Tests auch Verfahren kombiniert. So können beispielsweise die in einer Entscheidungstabelle identifizierten Bedingungen einer Äquivalenzklassenbildung unterzogen werden, um unterschiedliche Möglichkeiten herauszufinden, wie eine Bedingung erfüllt werden kann. Die Tests decken dann nicht nur alle Kombinationen von Bedingungen ab, sondern es werden zusätzliche Tests für die Äquivalenzklassen der Bedingungen erstellt, sodass auch diese überdeckt werden.

## 4.3 Strukturorientierte Testverfahren

Strukturorientierte Testentwurfsverfahren heißen auch White-Box- oder codebasierte Testverfahren. Dabei verwenden die Tester Code, Daten, Architektur oder Systemfluss als Grundlage des Testentwurfs und leiten die Tests systematisch aus der Struktur ab. Welche Elemente der Struktur berücksichtigt werden, hängt vom Verfahren ab. Das Verfahren liefert auch Kriterien für den Überdeckungsgrad, der entscheidet, wann die Ableitung von Tests abgeschlossen werden kann. Wenn der Überdeckungsgrad vollständig ist, heißt das nicht, dass die Menge von Tests vollständig ist. Es bedeutet vielmehr, dass die untersuchte Struktur für das vorliegende Testverfahren keine weiteren nützlichen Tests bietet. Der Überdeckungsgrad muss gemessen und mit dem Ziel verglichen werden, das für Projekt oder Unternehmen definiert wurde.

Gemeinsame Merkmale der strukturorientierte Testverfahren sind

- Die Testfälle werden aus Informationen darüber abgeleitet, wie die Software aufgebaut ist, beispielsweise dem Code und der Struktur.
- Der Überdeckungsgrad der Software lässt sich für die bestehenden Testfälle messen. Zusätzliche Testfälle lassen sich systematisch ableiten, um die Überdeckung zu erhöhen.

Der Advanced Level Lehrplan behandelt folgende strukturorientierte Testentwurfsverfahren:

| Name | Verfahren | Kriterien für die Überdeckung |
|---|---|---|
| Anweisungstest | Es werden alle ausführbaren (nicht kommentierten und nicht leeren) Anweisungen identifiziert. | Anzahl der ausgeführten Anweisungen/ Gesamtzahl der Anweisungen |
| Entscheidungsüberdeckungstest | Es werden alle Entscheidungsanweisungen identifiziert, wie if/else, switch/case, for und while. | Anzahl der ausgeführten Entscheidungen/Gesamtzahl der Entscheidungen |
| Zweigüberdeckungstest | Es werden alle Verzweigungen identifiziert, wie if/else, switch/case, for und while-Anweisungen. Hinweis: Entscheidungstests und Zweigtests sind bei einem Überdeckungsgrad von 100% gleich, können sich bei niedrigeren Überdeckungsgraden aber unterscheiden. | Anzahl der ausgeführten Zweige/Gesamtzahl der Zweige |
| Einfacher Bedingungstest | Es werden alle Wahr/Falsch-Bedingungen in Verzweigungsanweisungen identifiziert (beispielsweise if/else, switch/case, for und while). | Anzahl der ausgeführten booleschen Bedingungen/Gesamtzahl der booleschen Bedingungen |
| Mehrfachbedingungsüberdeckungstest | Es werden alle möglichen Kombinationen von Wahr/Falsch-Bedingungen identifiziert. | Anzahl der ausgeführten Kombinationen von booleschen Bedingungen/ Gesamtzahl der Kombinationen von booleschen Bedingungen |
| Definierter Bedingungstest | Es werden alle möglichen Kombinationen von Wahr/Falsch-Bedingungen identifiziert, die sich auf die Verzweigungsentscheidung (Zweige) auswirken können. | Anzahl der booleschen Operanden, die unabhängig voneinander die Entscheidung beeinflussen/Gesamtzahl der booleschen Operanden |
| LCSAJ (Schleifentest) | Es werden die möglichen Bedingungen identifiziert, die das Durchlaufen von Schleifen steuern. LCSAJ (Linear Code Sequence and Jump) [Beizer95] wird verwendet, um einen bestimmten Abschnitt im Code zu testen (eine lineare Folge ausführbarer Anweisungen). Der Abschnitt beginnt an einem bestimmten Kontrollfluss und endet mit einem Sprung zu einem anderen Kontrollfluss oder zum Ende des Programms. Diese Code-Fragmente werden auch als EE-Pfade bezeichnet | Anzahl der ausgeführten LCSAJ-Tests/Gesamtzahl der LCSAJ-Tests |

| Name | Verfahren | Kriterien für die Überdeckung |
|---|---|---|
| | (Entscheidung zu Entscheidung). Das Verfahren wird verwendet, um spezifische Testfälle mit den zugehörigen Testdaten zu definieren, die den gewählten Kontrollfluss ausführen. Für den Entwurf dieser Tests wird ein Modell des Quellcodes benötigt, das die Sprünge im Kontrollfluss definiert. LCSAJ kann als Basis für die Codeüberdeckungsmessung dienen. | |
| Pfadüberdeckungstest | | |
| | Es werden alle eindeutigen Folgen von Anweisungen (Pfade) identifiziert. | Anzahl der ausgeführten Pfade/Gesamtzahl der Pfade |

Mittels strukturorientierter Verfahren ermittelte Überdeckungskriterien messen oft die Vollständigkeit oder Unvollständigkeit einer Menge von Tests, die mit spezifikationsorientierten und/oder erfahrungsbasierten Testverfahren entworfen wurden. Dazu werden Testwerkzeuge eingesetzt, sogenannte Codeüberdeckungswerkzeuge, um die von den Tests erzielte strukturelle Überdeckung zu erfassen. Wenn dabei wichtige Lücken in der strukturellen Überdeckung aufgedeckt werden (oder die durch geltende Standards vorgeschriebene Überdeckung nicht erreicht wird), dann schließt man diese Lücken durch zusätzliche Tests nach strukturorientierten oder anderen Testverfahren.

**Analyse der Überdeckung**

Ob eine bestimmte Codeüberdeckung erzielt wurde oder nicht, lässt sich durch dynamisches Testen nach strukturorientierten oder anderen Testverfahren feststellen. Bei kritischen Systemen liefert es den Nachweis, dass eine spezifische Codeüberdeckung erreicht wurde (siehe Abschnitt 1.3.2). Die Ergebnisse können zeigen, dass einige Codeabschnitte nicht ausgeführt wurden, und zu einer Definition zusätzlicher Testfälle führen, damit auch diese Abschnitte ausgeführt werden.

# 4.4 Fehlerbasierte und erfahrungsbasierte Testverfahren

## 4.4.1 *Fehlerbasierte Testverfahren*

Bei fehlerbasierten Testentwurfsverfahren dient die Kategorie des gesuchten Fehlers als Basis für den Testentwurf, wobei die Tests systematisch davon abgeleitet werden, was über den Fehler bekannt ist.

Das Verfahren liefert auch Kriterien für den Überdeckungsgrad, der entscheidet, wann die Ableitung von Tests abgeschlossen werden kann. In der Praxis sind die Kriterien für Überdeckungsgrade bei fehlerbasierten Testentwurfsverfahren weniger systematisch als bei verhaltensbasierten und strukturorientierten Verfahren. Die allgemeinen Regeln zur Überdeckung sind zwar vorgegeben, es liegt aber im eigenen Ermessen, wo genau die Grenze einer sinnvollen Überdeckung in Testentwurf und -durchführung liegt. Wenn der Überdeckungsgrad vollständig ist, heißt das nicht, dass die Menge von Tests vollständig ist. Es bedeutet vielmehr, dass die berücksichtigten Fehler für dieses Testverfahren keine weiteren sinnvollen Tests erlauben.

Der Advanced Level Lehrplan behandelt das folgende fehlerbasierte Testentwurfsverfahren:

| Name | Verfahren | Kriterien für die Überdeckung |
|---|---|---|
| Fehlertaxonomien (Kategorien und Listen möglicher Fehler) | Der Tester, der eine Fehlertaxonomie verwendet, wählt ein potenzielles Problem zur Analyse aus der Liste. Fehlertaxonomien können Grundursache, Fehlerzustände und Fehlerwirkung angeben, sie listen die häufigsten Fehler in der getesteten Software auf. Die Liste wird für den Entwurf von Testfällen verwendet. | Identifizierte Datenfehler und Fehlerarten. |

## 4.4.2 *Erfahrungsbasierte Testverfahren*

Es gibt weitere Testentwurfsverfahren, bei denen die Fehlerhistorie eine Rolle spielt, die aber nicht zwangsläufig systematische Überdeckungsgrade liefern. Sie fallen unter die Kategorie der erfahrungsbasierten Testentwurfsverfahren.

Bei erfahrungsbasierten Tests nutzt man die fachliche Kompetenz und Intuition der Tester sowie deren Erfahrungen mit ähnlichen Anwendungen oder Technologien. Die Tests sind durchaus effektiv beim Auffinden von Fehlern, sie sind aber weniger geeignet als andere Verfahren, wenn es darum geht, bestimmte Überdeckungsgrade zu erzielen oder wiederverwendbare Testszenarien zu produzieren.

Beim Einsatz dynamischer oder heuristischer Ansätze neigen Tester dazu, erfahrungsbasierte Tests zu verwenden. Das Testen ist dann eher eine Reaktion auf Ereignisse als eine geplante Vorgehensweise. Außerdem finden Testdurchführung und Testauswertung gleichzeitig statt. Einige strukturierte Vorgehensweisen bei erfahrungsbasierten Tests sind nicht durchweg dynamisch; die Tests werden also nicht vollkommen gleichzeitig entworfen und durchgeführt.

Hinweis: Erfahrungsbasierte Verfahren liefern keine formalen Kriterien für Überdeckungsgrade, auch wenn in der nachfolgenden Tabelle einige Aspekte der Überdeckung erscheinen.

Der Advanced Level Lehrplan behandelt die folgenden erfahrungsbasierten Testentwurfsverfahren:

| Name | Verfahren | Kriterien für die Überdeckung |
|---|---|---|
| Intuitive Testfallermittlung | Tester nutzen ihre Erfahrung, um Fehler zu erraten, die vielleicht gemacht wurden, und bestimmen die Methoden für deren Aufdeckung. Das Erraten von Fehlen ist auch bei der Risikoanalyse nützlich, um potenzielle Fehlerwirkungen zu identifizieren. [Myers97] | Anzahl der identifizierte Datenfehler und Fehlerkategorien, wenn eine Taxonomie eingesetzt wurde. Ohne Taxonomie ist die Überdeckung begrenzt durch die Erfahrenheit des Testers und die verfügbare Zeit. |
| Checklistenbasiert | Erfahrene Tester benutzen eine abstrakte Checkliste mit Punkten, die beachtet und geprüft werden müssen oder nicht vergessen werden dürfen. Es kann auch ein Satz von Vorschriften und Kriterien sein, gegen die das Produkt geprüft und verifiziert werden muss. Diese Checklisten werden aus einer Reihe von Standards, Erfahrungen und anderen relevanten Überlegungen zusammengestellt. Beispiel für einen checklistenbasierten Test ist eine Checkliste mit Standards für Benutzerschnittstellen, die als | Alle Punkte der Checkliste wurden abgedeckt. |

| Name | Verfahren | Kriterien für die Überdeckung |
|---|---|---|
| | Grundlage der Tests für die Anwendung dient. | |
| Explorativ | Die Tester lernen gleichzeitig das Produkt und dessen Fehler kennen, planen vorgesehene Testaufgaben, entwerfen die Tests und führen sie durch, und berichten über die Testergebnisse. Gute explorative Tests sind geplant, interaktiv und kreativ. Die Tester passen die Testziele während der Testdurchführung dynamisch an und dokumentieren dabei nur sparsam. [Veenendaal02] | Es lassen sich Test-Chartas erstellen, die Aufgaben, Ziele und Arbeitsergebnisse spezifizieren. Für explorative Testsitzungen lässt sich spezifizieren, was erreicht werden soll, worauf man sich konzentrieren sollte, was innerhalb und außerhalb des Leistungsumfangs liegt und welche Ressourcen vorzusehen sind. Außerdem können Heuristiken über Fehler und Qualität einfließen. |
| Fehlerangriffe | Die Tester bewerten die Systemqualität, indem sie gezielt versuchen, bestimmte Fehlerwirkungen auszulösen. Bei [Whittaker03] ist das Prinzip von Fehlerangriffen, dass sie auf den Interaktionen der getesteten Software mit ihrer Betriebsumgebung basieren. Dazu gehören: Benutzerschnittstelle, Betriebssystem mit Kernel, Systemschnittstellen (APIs) und Dateisystem. Die Interaktionen basieren auf einem genau definierten Datenaustausch, falsche Einstellungen bei einer oder mehreren Interaktionen können eine Fehlerwirkung verursachen. | Kriterien sind die verschiedenen Schnittstellen der getesteten Anwendung. Die wichtigsten sind Benutzerschnittstelle, Betriebssystem, Kernel des Betriebssystems, APIs und Dateisystem. |

Fehler- und erfahrungsbasierte Testentwurfsverfahren nutzen die Kenntnis von Fehlern und andere Erfahrungen, um die Fehleraufdeckung zu erhöhen. Sie reichen von Schnelltests, bei denen Tester überhaupt keine formal geplanten Aktivitäten haben, über geplante Testsitzungen bis hin zu Tests mit Testskripten (skriptbasiertes Testen). Sie sind fast immer nützlich; unter folgenden Bedingungen sind sie aber besonders wertvoll:

- Es sind keine Spezifikationen vorhanden oder verfügbar.
- Das zu testende System ist schlecht dokumentiert.
- Für Entwurf und Erstellung von Testszenarien ist nicht genug Zeit.
- Die Tester sind in diesem Fachbereich und/oder in der Technologie besonders erfahren.
- Es wird eine Erhöhung der Vielfalt gegenüber dem reinen Testen auf Basis von Testskripten gesucht.
- Betriebsausfälle sind zu analysieren.

Fehler- und erfahrungsbasierte Testentwurfsverfahren sind außerdem nützlich, wenn sie mit verhaltensbasierten und strukturorientierten Verfahren kombiniert werden, weil sie die Lücken in der Testüberdeckung schließen, die aus den systematischen Schwächen dieser Verfahren resultieren.

## 4.5 Statische Analyse

Bei der statischen Analyse wird getestet, ohne dass die zu testende Software tatsächlich ausgeführt wird; man betrachtet beispielsweise den Code oder die Systemarchitektur.

## 4.5.1 Statische Analyse des Programmcodes

Als statische Analyse bezeichnet man jede Art von Testen, die durchgeführt werden kann, ohne den Programmcode auszuführen. Es stehen die folgenden Analyseverfahren zur Verfügung.

### 4.5.1.1 Kontrollflussanalyse

Die Kontrollflussanalyse liefert Informationen über logische Entscheidungsknoten im Softwaresystem und die Komplexität ihrer Struktur. Die Kontrollflussanalyse ist im ISTQB® Foundation-Level-Lehrplan und in [Beizer95] beschrieben.

### 4.5.1.2 Datenflussanalyse

Die Datenflussanalyse ist ein strukturiertes Testverfahren, bei dem Datenflusspfade zwischen dem Setzen einer Variablen und ihrer späteren Verwendung analysiert werden. Diese Pfade bezeichnet man als Define-Use-Paare (Definieren-Verwenden) oder Set-Use-Paare (Setzen-Verwenden). Bei dieser Methode werden Testmengen erstellt, die nach Möglichkeit eine 100%ige Überdeckung all dieser Paare erreichen.

Obwohl das Verfahren als Datenflussanalyse bezeichnet wird, prüft es auch den Kontrollfluss der Software, weil es Setzen und Verwendung jeder Variablen prüft und dabei dem Kontrollfluss der Software folgt. [Beizer95].

### 4.5.1.3 Konformität mit Programmierkonventionen

Die statische Analyse kann auch die Einhaltung von Programmierkonventionen im vorhandenen Code bewerten. Programmierkonventionen geben sowohl Aspekte der Systemarchitektur als auch die erlaubte oder nicht erlaubte Verwendung von Programmierstrukturen vor.

Die Einhaltung von Programmierkonventionen beim Codieren verbessert Wartbarkeit und Testbarkeit der Software. Statische Analysen können auch bestimmte Sprachanforderungen prüfen.

### 4.5.1.4 Erzeugung von Codemetriken

Bei einer statischen Analyse können Codemetriken ermittelt werden, die bei Einhaltung festgelegter Grenzen zu einer höheren Wartbarkeit und Zuverlässigkeit des Programmcodes beitragen. Beispiele für solche Metriken sind

- zyklomatische Zahl
- Größe
- Häufigkeit von Kommentaren
- Anzahl der Verschachtelungen
- Anzahl von Funktionsaufrufen

## 4.5.2 Statische Analyse der Softwarearchitektur

### 4.5.2.1 Statische Analyse einer Webseite

Auch die Architektur einer Webseite lässt sich mit Hilfe von statischen Analysewerkzeugen bewerten. Hier geht es darum zu prüfen, ob die Baumstruktur der Webseite ausgeglichen ist oder ob eine Unausgewogenheit vorliegt, die folgende Konsequenzen haben könnte:

- schwierigere Testaufgaben
- höherer Arbeitsaufwand für die Wartung
- schwierige Navigation für den Nutzer

Spezifische Testwerkzeuge, die mit *web spider engines* ausgerüstet sind, können über die statische Analyse auch Informationen über die Größe der Seiten liefern, Dauer des Herunterladens sowie über

das Vorhandensein/Fehlen einer Seite (http error 404). Das sind nützliche Informationen für Entwickler, Webmaster und Tester.

#### 4.5.2.2 Aufrufgraphen

Aufrufgraphen können unterschiedlichen Zwecken dienen:

- Tests entwerfen, die ein bestimmtes Modul aufrufen
- Herausfinden, wo das Modul überall aufgerufen wird
- Vorgehensweisen für die Integration vorschlagen (paarweise und benachbarte Integration [Jorgensen02])
- Struktur des Gesamtcodes und seiner Architektur bewerten

Zusätzlich kann die dynamische Analyse (siehe Abschnitt 4.6) Informationen darüber liefern, wie oft ein Modul aufgerufen wird. Die Tester können die Informationen aus den Aufrufgraphen der statischen Analyse mit denen aus der dynamischen Analyse zusammenführen und sich dann auf die Module konzentrieren, die oft und/oder wiederholt aufgerufen werden. Wenn es außerdem komplexe Module sind (siehe Abschnitt 1.3.2.2), sollten sie umfangreich und im Detail getestet werden.

## 4.6 Dynamische Analyse

Die dynamische Analyse untersucht die laufende Anwendung. Dazu ist meist eine Instrumentierung notwendig, die manuell oder automatisch in eine temporäre Kopie des Programmquellcode eingefügt wird.

### *4.6.1 Übersicht*

Fehlerwirkungen, die nicht ohne Weiteres reproduzierbar sind, können erhebliche Konsequenzen für den Testaufwand haben und können die Softwareproduktionsfreigabe verhindern. Ursachen solcher Fehler können Speicherengpässe sein, inkorrekter Einsatz von Zeigern und andere Korruptionen (beispielsweise des System-Stacks) [Kaner02]. Diese Art von Fehlern kann zu einer allmählichen Verschlechterung der Systemleistung oder sogar zu Systemabstürzen führen; die Teststrategie muss deshalb die mit diesen Fehlern verbundenen Risiken berücksichtigen. Falls nötig, müssen die Risiken durch eine dynamische Analyse reduziert werden (normalerweise mit Hilfe von Testwerkzeugen).

Die dynamische Analyse wird durchgeführt, während die Software läuft, und soll

- Fehler verhindern, indem fehlerhafte „wilde“ Zeiger und Speicherengpässe aufgedeckt werden.
- Systemausfälle analysieren, die nicht leicht reproduzierbar sind.
- Netzwerkverhalten bewerten.
- die Systemleistung verbessern, indem Informationen über das Laufzeitverhalten des Systems erfasst werden.

Die dynamische Analyse kann in jeder Teststufe durchgeführt werden, sie ist aber besonders nützlich für Komponenten- und Integrationstests. Zum Spezifizieren der Testziele und vor allem für die Analyse der Testergebnisse sind technische Fähigkeiten und Systemkenntnisse erforderlich.

### *4.6.2 Speicherengpässe aufdecken*

Ein Speicherengpass tritt auf, wenn der Speicher (RAM), der für ein Programm verfügbar ist, zwar allokiert, wegen Programmierfehlern aber nicht wieder freigegeben wird. Das Programm kann dann auf diesen Teil des Speichers nicht mehr zugreifen und der nutzbare Speicherplatz kann ausgehen.

Speicherengpässe verursachen Probleme, die sich erst allmählich entwickeln und oft nicht erkennbar sind, so beispielsweise, wenn die Software erst kürzlich installiert oder das System neu gestartet wurde, was beim Testen oft geschieht. Die negativen Auswirkungen von Speicherengpässen werden deshalb oft erst bemerkt, wenn das Programm in Produktion gegangen ist.

Symptom eines Speicherengpasses ist die kontinuierlichen Verlangsamung der Antwortzeiten, die letztendlich zu einem Systemausfall führen kann. Man kann solchen Ausfällen zwar mit einem Neustart (Rebooten) des Systems begegnen, das ist aber oft unpraktisch oder sogar unmöglich.

Bereiche, in denen Speicherengpässe vorkommen, lassen sich mit Werkzeugen identifizieren, so dass die Speicherengpässe korrigiert werden können. Mit einem einfachen Speichermonitor lässt sich herausfinden, ob der verfügbare Speicherplatz sich allmählich verringert, dann müsste noch eine Nachanalyse durchgeführt werden.

Es müssen noch andere Arten von Engpässen betrachtet werden, beispielsweise bei Dateibezeichnern, Zugriffsgenehmigungen (Semaphore) und Verbindungspools für Ressourcen.

### 4.6.3 Fehlerhafte Zeiger aufdecken

Fehlerhafte („wilde“) Zeiger in einem Programm sind unbrauchbare Zeiger. Sie entstehen, wenn es die Objekte oder die Funktionen nicht mehr gibt, auf die sie zeigen, oder wenn sie nicht auf den vorgesehenen Speicher verweisen, sondern beispielsweise auf einen Speicherbereich außerhalb des zugewiesenen Arrays. Wenn ein Programm fehlerhafte Zeiger enthält, kann das verschiedene Folgen haben:

1. Unter Umständen funktioniert das Programm wie erwartet, wenn der Zeiger auf quasi freie Speicherbereiche zeigt, die vom Programm zurzeit nicht genutzt werden.
2. Das Programm kann abstürzen, wenn der Zeiger auf einen kritischen Teil des Speichers für die Programmumgebung zeigt (beispielsweise des Betriebssystems).
3. Das Programm funktioniert nicht korrekt, weil es auf benötigte Objekte nicht zugreifen kann. Es kann dann zwar weiterhin funktionieren, gibt aber Fehlermeldungen aus.
4. Durch fehlerhafte Zeiger können Daten zerstört oder unbrauchbar werden, sodass dann inkorrekte Werte verwendet werden.

Hinweis: Jede Änderung an der Speichernutzung durch das Programm (beispielsweise ein neuer Build nach einer Softwareänderung) kann eine dieser vier Folgen haben. Dies ist vor allem dann kritisch, wenn das Programm zunächst wie erwartet funktioniert, obwohl es fehlerhafte Zeiger enthält, jedoch nach einer Softwareänderung abstürzt (vielleicht sogar im Produktionseinsatz). Solche Ausfälle sind Symptome eines Fehlerzustands (des fehlerhaften Zeigers) und nicht der Fehler selbst (siehe auch [Kaner02], Lesson 74).

Werkzeuge können fehlerhafte Zeiger identifizieren, die ein Programm verwendet, unabhängig von den Folgen der Zeiger für die Programmausführung.

### 4.6.4 Systemleistung analysieren

Mit einer dynamischen Analyse lässt sich auch die Programmleistung analysieren. Werkzeuge können Leistungsengpässe identifizieren und ein breites Spektrum an Performanzmetriken erzeugen. Damit können die Entwickler das System in einer sogenannten Performanzprofilierung (performance profiling) optimieren.

# 5. Test der Softwareeigenschaften

*Begriffe*

Anwendungsprofil, Benutzbarkeitstest, betrieblicher Abnahmetest, Effizienztest, heuristische Evaluation, Interoperabilitätstest, Portabilitätstest, Sicherheitstest, SUMI (Software Usability Measurement Inventory), Test auf Richtigkeit, Wartbarkeitstest, Wiederherstellbarkeitstest, Zugänglichkeitstest, Zuverlässigkeitstest, Zuverlässigkeits-Wachstumsmodell

## 5.1 Einführung

Während das vorige Kapitel die verschiedenen Testverfahren für Tester beschreibt, behandelt dieses Kapitel wie Tester die Verfahren anwenden, um die wichtigsten Qualitätsmerkmale für Softwareanwendungen oder Systeme zu bewerten.

Die Qualitätsmerkmale, die von Test Analysts und von Technical Test Analysts zu bewerten sind, werden in separaten Abschnitten dieses Lehrplans behandelt. Hintergrund ist die Beschreibung der Qualitätsmerkmale im ISO Standard 9126.

Die verschiedenen Qualitätsmerkmale zu verstehen, ist grundlegendes Lernziel aller drei Module. Je nachdem, welches der Qualitätsmerkmale behandelt wird, entsteht entweder im Modul für Test Analysts oder im Modul für Technical Test Analysts ein tieferes Verständnis. Die jeweiligen Adressaten können so typische Risiken erkennen, geeignete Teststrategien entwickeln und Testfälle spezifizieren.

## 5.2 Qualitätsmerkmale bei fachlichen Tests

Funktionale Tests sind in erster Linie darauf fokussiert, was das Produkt leistet (weniger auf das Wie). Sie basieren im Allgemeinen auf einer Spezifikation oder einem Anforderungsdokument, spezifischer Expertise in einem Fachgebiet oder einem unterstellten Bedarf. Funktionale Tests unterscheiden sich je nach der Teststufe oder -phase, in der sie durchgeführt werden. So wird beispielsweise bei einem Integrationstest die Funktionalität der Schnittstellenmodule für eine definierte Funktion getestet. Systemtests prüfen die Funktionalität der gesamten Anwendung. Bei Multisystemen werden mit funktionalen Tests vor allem die gesamten integrierten Systeme End to End getestet.

Funktionale Tests setzen viele unterschiedliche Testverfahren ein (siehe Kapitel 4). Dabei kann entweder ein dedizierter Tester die funktionalen Tests durchführen, ein Fachexperte oder ein Entwickler (wenn es um Komponenten geht).

Folgenden Qualitätsmerkmale werden untersucht:

- Richtigkeit
- Angemessenheit
- Interoperabilität
- Funktionale Sicherheit
- Benutzbarkeit
- Zugänglichkeit

### 5.2.1 Tests auf Richtigkeit

Funktionale Tests auf Richtigkeit prüfen die Einhaltung von spezifizierten oder impliziten Anforderungen an eine Anwendung; sie können auch die Richtigkeit der Berechnungen prüfen. Tests auf Richtigkeit nutzen viele der Testverfahren aus Kapitel 4.

### 5.2.2 Tests auf Angemessenheit

Tests auf Angemessenheit bewerten und validieren, ob sich eine Menge von Funktionen für die vorgesehenen spezifizieren Aufgaben eignen. Die Tests können auf Anwendungsfällen (use cases) oder auf Vorgehen basieren.

### 5.2.3 Interoperabilitätstests

Interoperabilitätstests prüfen, ob eine Anwendung in allen vorgesehenen Umgebungen (Hardware, Software, Middleware, Betriebssystem usw.) korrekt funktionieren. Für die Spezifikation der Interoperabilitätstests sind Kombinationen der vorgesehenen Zielumgebungen zu identifizieren, zu konfigurieren und dem Testteam zur Verfügung zu stellen. Diese Umgebungen werden dann mit ausgewählten funktionalen Testfällen getestet, welche die verschiedenen Komponenten der Testumgebung prüfen.

Interoperabilität betrifft das Zusammenwirken verschiedener Softwaresysteme. Software mit guten Interoperabilitätseigenschaften lässt sich leicht mit verschiedenen anderen System integrieren, ohne dass größere Änderungen nötig sind. Messen lässt sich die Interoperabilität durch die Anzahl notwendiger Änderungen und den damit verbundenen Aufwand.

Beim Testen der Softwareinteroperabilität können beispielsweise die folgenden Designmerkmale im Fokus stehen:

- die Verwendung von industrie-üblichen Kommunikationsstandards, beispielsweise XML;
- die Fähigkeit der Software, die Kommunikationsanforderungen anderer Systeme automatisch zu erkennen, mit denen sie zusammenwirkt, und sie entsprechend anzusteuern.

Interoperabilitätstests sind besonders wichtig für

- Unternehmen, die kommerzielle Standardsoftware und –werkzeuge entwickeln,
- die Entwicklung von Multisystemen.

Die Tests finden vorwiegend während der Systemintegrationstests statt.

### 5.2.4 Funktionale Sicherheitstests

Funktionale Sicherheitstests (Penetrationstests) fokussieren die Fähigkeit der Software, unberechtigten Zugriff auf Daten oder Funktionen zu verhindern, unabhängig davon, ob der Zugriff unbeabsichtigt oder vorsätzlich erfolgt. Die Tests untersuchen Rechte der Nutzer, Zugriff und Berechtigungsklassen. Diese Informationen sollten in den Spezifikationen des Systems enthalten sein. Sicherheitstests betreffen auch einige Aspekte, die eher für Technical Test Analysts relevant sind, sie behandelt Abschnitt 5.3.

### 5.2.5 Benutzbarkeitstests

Die Ursachen für mögliche Schwierigkeiten von Benutzern bei ihrer Arbeit mit dem zukünftigen System müssen erkennbar sein. Nutzer können zu sehr unterschiedlichen Personengruppen gehören, angefangen von IT-Experten bis hin zu Kindern oder Menschen mit besonderen Bedürfnissen.

Einige nationale Verbände (beispielsweise das British Royal National Institute for the Blind) empfehlen, dass Webseiten für behinderte, blinde, sehbehinderte oder taube Nutzer und für

Menschen mit Bewegungs- oder Wahrnehmungseinschränkungen auch zugänglich sein sollten. Die Prüfung, ob eine Webseite für diese Personengruppen nutzbar ist, verbessert die Benutzbarkeit auch für alle anderen.

Benutzbarkeitstests prüfen die Eignung der Software für ihre Nutzer. Sie messen dabei anhand der folgenden Faktoren, ob festgelegte Nutzer in bestimmten Umgebungen oder Anwendungskontexten spezifizierte Ziele erreichen:

- Effektivität: Eignung der Anwendung für die Nutzer, spezifizierte Ziele in einem spezifizierten Anwendungskontext richtig und vollständig zu erreichen
- Effizienz: Eignung der Anwendung für die Nutzer, der Effektivität angemessene Ressourcen in einem spezifizierten Anwendungskontext einzusetzen
- Zufriedenheit: Eignung der Anwendung, die Nutzer in einem spezifizierten Anwendungskontext zufriedenzustellen

Messbare Merkmale sind

- Verständlichkeit: Softwaremerkmale, die beeinflussen, welchen Aufwand die Nutzer leisten müssen, um das logische Konzept und dessen Anwendbarkeit zu erkennen
- Erlernbarkeit: Softwaremerkmale, die beeinflussen, welchen Aufwand die Nutzer leisten müssen, um die Anwendung zu erlernen
- Operabilität: Softwaremerkmale, die beeinflussen, welchen Aufwand die Nutzer leisten müssen, um Aufgaben effektiv und effizient auszuführen
- Attraktivität: Eigenschaft der Software, dass sie dem Nutzer gefällt

Die Bewertung der Benutzbarkeit hat zwei Ziele:

- Benutzbarkeitsfehler zu beseitigen (auch formative Bewertung)
- zu testen, ob die Benutzbarkeitsanforderungen erfüllt sind (auch summative Bewertung)

Die Tester sollten Wissen oder Expertise in folgenden Wissensbereichen haben:

- Soziologie
- Psychologie
- Einhaltung nationaler Standards oder Vorschriften (beispielsweise das Allgemeine Gleichbehandlungsgesetz ((AGG)) vom 17. August 2006)
- Ergonomie

Das implementierte System sollte unter realitätsnahen Bedingungen validiert werden. Möglicherweise muss dazu ein Benutzbarkeitslabor eingerichtet werden mit Videokameras, nachgebauten Büros, Review-Gruppen, Nutzern usw., damit das Entwicklungsteam die Wirkung des tatsächlichen Systems auf echte Personen beobachten kann.

Viele Benutzbarkeitstests werden als Teil anderer Tests durchgeführt, beispielsweise beim funktionalen Systemtest. Eine Ergonomierichtlinie ist hilfreich beim konsistenten Herangehen an Aufdeckung und Berichterstattung von Benutzbarkeitsfehlern in allen Softwarelebenszyklusphasen.

#### 5.2.5.1 Benutzbarkeitstests spezifizieren

Die wichtigsten Verfahren für Benutzbarkeitstests sind

- Reviews (z.B. Inspektionen) oder Bewertungen
- Verifizierung und Validierung der tatsächlichen Systemimplementierung
- Gutachten und Befragungen

**Reviews**

Weil Reviews (z.B. Inspektionen) von Spezifikation und Entwürfen unter Gesichtspunkten der Benutzbarkeit die Nutzerbeteiligung erhöhen, können sie Probleme früh entdecken und kosteneffektiv sein.

Mit einer heuristischen Evaluation (der systematischen Inspektion des Entwurfs einer Benutzerschnittstelle auf ihre Benutzbarkeit) lassen sich Probleme der Benutzbarkeit im Entwurf aufdecken, sodass sie im iterativen Entwurfsprozess bearbeitet werden können. Dabei prüft ein kleines Prüferteam die Schnittstelle und ihre Konformität mit anerkannten Grundsätzen der Benutzbarkeit (den heuristischen Grundsätzen der Ergonomie).

**Validierung der tatsächlichen Systemimplementierung**

Zur Validierung der tatsächlichen Systemimplementierung können Benutzbarkeitstestszenarien aus Tests entwickelt werden, die für den funktionalen Systemtest spezifiziert wurden. In diesen Testszenarien werden statt der funktionalen Ergebnisse die spezifischen Benutzbarkeitsmerkmale gemessen, beispielsweise Lerngeschwindigkeit oder Operabilität.

Es lassen sich spezifische Benutzbarkeitstestszenarien für das Testen von Syntax und Semantik entwickeln:

- Syntax: Aufbau oder Grammatik der Schnittstelle (was beispielsweise in ein Eingabefeld eingegeben werden kann)
- Semantik: Bedeutung und Zweck (beispielsweise sinnvolle und aussagekräftige Systemmeldungen und –ausgaben an den Nutzer)

Mögliche Verfahren für die Entwicklung solcher Testszenarien sind

- Black-Box-Verfahren, wie beispielsweise im BS 7925-2 (British Standard) beschrieben
- Anwendungsfälle, definiert entweder in Textform oder UML (Unified Modeling Language)

Testszenarien für Benutzbarkeitstests enthalten Anleitungen für die Nutzer, Zeiträume vor und nach den Tests für Interviews, in denen die Anleitung vermittelt oder die Rückmeldungen gesammelt werden, sowie das vereinbarte Vorgehen im Verlauf der Testeinheiten. Das Vorgehen legt fest, wie die Tests ausgeführt werden, ihren zeitlichen Ablauf, Protokollierung und Aufzeichnung der Testsitzung sowie die Methoden für Begutachtung und Befragung.

**Gutachten und Befragungen**

Mit Gutachten und Befragungen lassen sich Beobachtungen über das Verhalten der Anwender bei der Systemnutzung in einem Testlabor sammeln. Standardisierte und öffentlich zugängliche Gutachten, wie etwa SUMI (Software Usability Measurement Inventory) und WAMMI (Website Analysis and MeasureMent Inventory) ermöglichen ein Benchmarking gegen eine Datenbank mit früheren Benutzbarkeitsmessungen. SUMI liefert konkrete Benutzbarkeitsmetriken, diese lassen sich als Testende- oder Abnahmekriterien verwenden.

## *5.2.6 Zugänglichkeitstests*

Es ist wichtig, die Zugänglichkeit der Software für Menschen mit besonderen Anforderungen oder eingeschränkten Nutzungsmöglichkeiten zu prüfen. Dazu gehören auch behinderte Personen. Die relevanten Standards sollten eingehalten werden, wie etwa Richtlinien über Barrierefreiheit (Web Content Accessibility Guidelines) oder Anti-Diskriminierungsgesetze (Disability Discrimination Acts, Großbritannien, Australien) und Section 508 (USA).

## 5.3 Qualitätsmerkmale bei technischen Tests

Die für Technical Test Analysts relevanten Qualitätsmerkmale sind in erster Linie darauf fokussiert, wie das Produkt funktioniert (weniger auf das funktionale Was). Die Tests können in jeder Teststufe stattfinden, besonders relevant sind sie für:

Komponententest (besonders für Echtzeitsysteme und eingebettete Systeme)

- Performanz-Benchmarking
- Ressourcennutzung

Systemtests und betrieblicher Abnahmetest (Produktionsabnahmetest)

- Je nach Risiken und verfügbaren Ressourcen gehören dazu die nachfolgend genannten Qualitätsmerkmale und Teilmerkmale.
- Technisch ausgerichtete Tests in dieser Teststufe beziehen sich auf das Testen eines spezifischen Systems, d.h. einer Kombination aus Hardware und Software, einschließlich Server, Clients, Datenbanken, Netzwerken und anderen Ressourcen.

Diese Tests werden oft auch noch durchgeführt, nachdem die Software in Produktion gegangen ist. Dann ist oft ein separates Team oder eine Organisation dafür zuständig. Metriken für die Qualitätsmerkmale aus den Tests vor Produktionseinführung können als Grundlage für die Service Level-Vereinbarungen zwischen Lieferant und Betreiber des Softwaresystems dienen.

Folgenden Qualitätsmerkmale werden getestet:

- technische Sicherheit
- Zuverlässigkeit
- Effizienz
- Wartbarkeit
- Portabilität

### *5.3.1 Technische Sicherheitstests*

Sicherheitstests unterscheiden sich von anderen Formen fachlicher oder technischer Tests in zwei wichtigen Aspekten:

1. Standardverfahren zur Auswahl der Testeingangsdaten können wichtige Sicherheitsthemen übersehen.
2. Die Symptome von Sicherheitsfehlern unterscheiden sich grundlegend von Symptomen bei anderen Testverfahren.

Sicherheits-Schwachstellen gibt es an Stellen, wo die Software nicht nur funktioniert wie im Entwurf vorgesehen, sondern wo sie zusätzlich weitere, nicht beabsichtigte Aktionen ausführt. Solche Nebeneffekte sind eine der größten Bedrohungen für die Softwaresicherheit gegenüber Angriffen. Beispiel: Eine Abspielsoftware spielt Audio korrekt ab, schreibt dabei aber Dateien in einen nicht verschlüsselten Zwischenspeicher. Softwarepiraten könnten diesen Nebeneffekt ausnutzen.

Wichtigste Aufgabe der Softwaresicherheit ist es, eine nicht beabsichtigte Nutzung von Informationen durch nicht autorisierte Personen zu verhindern. Sicherheitstests versuchen, die Sicherheitsrichtlinie eines Systems außer Kraft zu setzen, indem sie seine Schwachstellen für Bedrohungen durch verschiedene Maßnahmen ausnutzen:

- Nicht autorisiertes Kopieren von Anwendungen oder Daten (beispielsweise Softwarepiraterie)
- Nicht autorisierter Zugriff (Aktionen auszuführen, für die der Nutzer keine Zugriffsberechtigung hat)

- Überlauf des Eingabebereichs (Pufferüberlauf), der beispielsweise durch Eingabe extrem langer Zeichenketten über ein Eingabefeld der Benutzerschnittstelle ausgelöst werden kann. Nach diesem Überlauf lässt sich möglicherweise bösartiger Code ausführen.
- Denial of Service-Angriff, sodass die Anwendung nicht mehr genutzt werden kann (beispielsweise indem ein Webserver von Übermengen von Störanfragen überschwemmt wird)
- Abhören von Datenübertragungen in Netzwerken, um an vertrauliche Informationen zu gelangen (beispielsweise über Kreditkarten-Transaktionen)
- Verschlüsselungscodes knacken, die vertrauliche Daten schützen
- Logische Fallen (logic bombs, in den USA manchmal Easter Eggs genannt), die in bösartiger Absicht in den Code eingeschleust und nur unter bestimmten Bedingungen aktiviert werden (beispielsweise an einem bestimmten Datum). Werden diese logischen Fallen aktiviert, so lösen sie Schadaktionen aus, wie das Löschen von Dateien oder Formatieren von Festplatten.

Sicherheitsbelange lassen sich folgendermaßen einteilen:

- Die Benutzerschnittstelle betreffend
  - nicht autorisierter Zugriff
  - bösartige Eingaben
- Das Dateisystem betreffend
  - Zugriff auf vertrauliche Daten in Dateien oder Repositories
- Das Betriebssystem betreffend
  - unverschlüsseltes Ablegen von sicherheitskritischen Informationen, wie Passwörtern. Wird das System durch bösartige Eingaben zum Absturz gebracht, können die Informationen zugänglich werden.
- Externe Software betreffend
  - Interaktionen zwischen externen Komponenten, die das System nutzt. Sie können auf Netzwerkebene stattfinden (wenn beispielsweise inkorrekte Datenpakete oder Meldungen übertragen werden) oder auf Ebene der Softwarekomponenten (wenn beispielsweise eine Softwarekomponente ausfällt, die vom System benötigt wird).

Hinweis: Verbesserungen bei der Sicherheit eines Systems können die Systemleistung beeinträchtigen. Es ist also ratsam, nach den Sicherheitsverbesserungen die Performanztests zu wiederholen.

#### 5.3.1.1 Sicherheitstestspezifikation

Mögliche Verfahren beim Entwickeln der Sicherheitstests sind

- Informationen abrufen
  Mit verbreiteten Werkzeugen wird ein Profil oder Netzwerkplan des Systems erstellt. Mögliche Informationen sind: Mitarbeiter, physikalische Adressen, Details der internen Netzwerke, IP-Nummern, Typ der verwendeten Software/Hardware, Betriebssystem-Version.
- Schwachstellen scannen
  Mit den verbreiteten Werkzeugen wird das System gescannt. Diese Werkzeuge werden nicht dazu verwendet, in das System direkt einzudringen, sondern zur Identifizierung von Schwachstellen, die eine Verletzung der Sicherheitsrichtlinie darstellen oder dazu führen können. Spezifische Schwachstellen lassen sich auch mit Hilfe von Checklisten, wie etwa bei www.testingstandards.co.uk, identifizieren.
- Fehlerangriffe entwickeln
  Mit den gesammelten Informationen werden Testaktionen geplant, die die Sicherheitsrichtlinie eines bestimmten Systems durchdringen sollen. Für die Angriffspläne müssen mehrere

Eingaben über verschiedene Schnittstellen (beispielsweise Benutzerschnittstelle, Dateisystem) spezifiziert werden, um so die schwersten Sicherheitsdefekte aufzudecken.

- Die verschiedenen bei [Whittaker04] beschriebenen Fehlerangriffe sind eine wertvolle Quelle für Verfahren, die speziell für Sicherheitstests entwickelt wurden. Weitere Informationen zu Fehlerangriffen enthält Abschnitt 4.4.

## *5.3.2 Zuverlässigkeitstests*

Ein Ziel beim Testen der Zuverlässigkeit ist es, das statistische Maß an Softwarereife (Maturity) über eine Zeitspanne zu überwachen und mit der zu erzielenden Zuverlässigkeit zu vergleichen. Metriken können sein: Die durchschnittliche Zeit zwischen Ausfällen (MTBF, Mean Time Between Failures), die durchschnittliche Zeit für die Fehlerbehebung (MTTR, Mean Time to Repair) oder eine andere Messung der Fehlerdichte (beispielsweise wöchentliche Anzahl der Fehler mit einem bestimmten Schweregrad). Ein Modell zur Überwachung von Zuverlässigkeitswerten über eine Zeitspanne bezeichnet man auch als Zuverlässigkeits-Wachstumsmodell (Reliability Growth Model).

Es gibt verschiedene Formen von Zuverlässigkeitstests, beispielsweise eine Menge definierter Tests, die wiederholt werden, zufällig aus einem Pool gewählte Tests oder Testfälle, die aus einem statistischen Modell generiert wurden. Der Zeitaufwand für diese Tests kann daher erheblich sein, bis zu mehreren Tagen.

Die Analyse der über eine Zeitspanne gemessenen Softwarereife kann Testendekriterien (beispielsweise für die Produktionsfreigabe) liefern. Bestimmte Metriken, wie MTBF und MTTR, können in Service Level-Vereinbarungen umgesetzt und in der Produktion überwacht werden.

Software Reliability Engineering & Testing (SRET) ist ein Standardverfahren für das Zuverlässigkeitstesten.

### 5.3.2.1 Robustheitstest

Während funktionale Tests die Fehlertoleranz der Software gegenüber unerwarteten Eingaben bewerten (Negativtests), bewerten die technisch ausgerichteten Tests eher die Toleranz des Systems gegenüber Fehlern, die außerhalb der zu testenden Anwendung auftreten. Normalerweise meldet das Betriebssystem solche Fehler (beispielsweise Festplatte voll, Prozess oder Dienst nicht verfügbar, Datei nicht gefunden, Speicher nicht verfügbar). Für Tests der Fehlertoleranz auf Systemebene gibt es spezifische Werkzeuge.

### 5.3.2.2 Wiederherstellbarkeitstest

Weitere Formen von Zuverlässigkeitstests bewerten, wie sich das System nach Hardware- oder Softwareversagen mit einem definiertem Vorgehen wiederherstellen lässt, sodass anschließend der normale Betrieb fortgesetzt werden kann. Zu Wiederherstellbarkeitstests gehören auch Failover Tests und Backup- und Wiederherstellungstests.

Failover Tests werden durchgeführt, wenn die Konsequenzen eines Softwareversagens so gravierend sind, dass spezielle Hardware- und/oder Softwaremaßnahmen beim System implementiert wurden, um den weiteren Betrieb selbst nach einem Versagen sicherzustellen. Failover Tests können sinnvoll sein, wenn beispielsweise das Risiko finanzieller Verluste als extrem hoch einzustufen ist, oder wenn kritische Sicherheitsanforderungen vorliegen. Wenn die Ursache solcher Systemausfälle eine Katastrophe sein kann, bezeichnet man diese Art von Wiederherstellbarkeitstests auch als Katastrophentests.

Typische Hardware-basierte Maßnahmen können beispielsweise ein Lastausgleich über mehrere Prozessoren sein, das Clustern von Servern, Prozessoren oder Festplatten, sodass bei einem Ausfall sofort die eine Komponente von der anderen übernehmen kann (beispielsweise RAID, Redundant Array of Inexpensive Disks). Eine typische Software-basierte Maßnahme kann die Implementierung

von mehr als einer unabhängigen Instanz des Softwaresystems in so genannten redundanten Systemen sein (beispielsweise beim Steuerungssystem eines Flugzeugs). Redundante Systeme sind normalerweise eine Kombination aus Software- und Hardwaremaßnahmen und werden je nach Anzahl der unabhängigen Instanzen als zweifach/dreifach/vierfach redundante Systeme bezeichnet. Unterschiedliche Lösungen lassen sich dadurch erreichen, dass zwei (oder mehr) unabhängige Entwicklungsteams dieselben Softwareanforderungen zur Umsetzung erhalten. Die gleiche Leistung wird dann mit unterschiedlichen Softwarelösungen erbracht. Das schützt die mehrfach redundanten Systeme, weil nicht wahrscheinlich ist, dass ähnliche fehlerhafte Eingaben das gleiche Ergebnis haben. Diese Maßnahmen zur Verbesserung der Wiederherstellbarkeit eines Systems können auch dessen Zuverlässigkeit direkt beeinflussen und bei der Durchführung der Zuverlässigkeitstests berücksichtigt werden.

Durch die Simulation von Fehlerwirkungen oder das Ausführen in einer kontrollierten Umgebung sind Failover Tests ausdrücklich auf das Testen redundanter Systeme ausgelegt. Nach einer Fehlerwirkung wird der Failover-Mechanismus getestet, um sicherzustellen, dass der Vorfall weder Datenverlust noch Datenkorruption bewirkt hat, und dass die Service Level-Vereinbarungen eingehalten wurden (beispielsweise Funktionsverfügbarkeit, Antwortzeiten). Weitere Informationen zu Failover Tests unter www.testingstandards.co.uk.

Backup- und Wiederherstellungstests untersuchen Maßnahmen und Verfahren, die die möglichen Fehlerwirkungen minimieren sollen. Sie bewerten die (meist in einer Teststrategie dokumentierten) Verfahrensweisen, nach denen die verschiedenen Sicherungsarten erstellt und die Daten nach Datenverlust oder –korruption wiederhergestellt werden. Die Testfälle werden so entworfen, dass kritische Pfade im Verfahren abgedeckt sind. Als Trockenübung für diese Szenarien lassen sich technische Reviews durchführen, die die Handbücher für die tatsächliche Installationsprozedur validieren. Beim betrieblichen Abnahmetest werden diese Szenarien in einer echten Produktionsumgebung oder in einer produktionsähnlichen Umgebung eingesetzt, um ihre tatsächliche Anwendung zu validieren.

Mögliche Metriken für die Messung bei Backup- und Wiederherstellungstests sind

- benötigte Zeit für die verschiedenen Backup-Arten (beispielsweise vollständig, inkrementell)
- benötigte Zeit für die Wiederherstellung der Daten
- garantierte Sicherungsstufen (beispielsweise Wiederherstellung aller Daten, die nicht älter als 24 Stunden sind, oder spezifischer Transaktionsdaten, die nicht älter als eine Stunde sind).

#### 5.3.2.3 Zuverlässigkeitstest-Spezifikation

Zuverlässigkeitstests basieren meist auf Anwendungsmustern (Nutzungsprofilen); sie können formal oder je nach Risiko durchgeführt werden. Die Testdaten können mit Zufalls- oder Pseudozufallsmethoden generiert werden.

Die Wahl der Zuverlässigkeits-Wachstumskurve sollte gut begründet sein und der Satz von Fehlerdaten mit Werkzeugen analysiert werden, damit die Zuverlässigkeits-Wachstumskurve den vorhandenen Daten angemessen ist.

Zuverlässigkeitstest eignen sich auch für die gezielte Suche nach Speicherengpässen. Sie müssen dann so spezifiziert werden, dass sie besonders speicherintensive Vorgänge wiederholt ausführen und eine ordnungsgemäße Speicherfreigabe nachweisen.

### *5.3.3 Effizienztests*

Das Qualitätsmerkmal Effizienz wird mit Tests zu Zeit- und Ressourcenverhalten bewertet. Die folgenden Abschnitte behandeln Effizienztests bezogen auf das Zeitverhalten unter den Aspekten Performanz-, Last-, Stress- und Skalierbarkeitstests.

#### 5.3.3.1 Performanztests

Es gibt unterschiedliche Arten von Performanztests, je nach betrachteten nicht-funktionalen Anforderungen. Zu diesen Testarten gehören Performanz-, Last-, Stress- und Skalierbarkeitstests.

Performanztests untersuchen die Fähigkeit einer Komponente oder eines Systems, auf Eingaben des Nutzers oder Systems innerhalb einer definierten Zeit und unter den spezifizierten Bedingungen zu reagieren (siehe auch Lasttests und Stresstests in den folgenden Abschnitten). Dabei variieren die Performanzmessungen je nach Zielsetzung des Tests. So kann die Performanzmessung einer einzelnen Softwarekomponente die CPU-Zyklen berechnen; bei Client-basierten Systemen lässt sich dagegen die Zeit messen, die das System benötigt, um auf eine bestimmte Nutzeranfrage zu reagieren. Bei Systemarchitekturen, die aus mehreren Komponenten bestehen (beispielsweise Clients, Servern, Datenbanken) wird die Performanz zwischen den einzelnen Systemkomponenten gemessen, um Engpässe zu identifizieren.

#### 5.3.3.2 Lasttests

Lasttests messen die Fähigkeit eines Systems, ansteigende Grade erwarteter, realistischer Systemlasten zu bewältigen, die eine Anzahl paralleler Nutzer als Transaktionsanforderungen generiert. Die durchschnittlichen Antwortzeiten der Nutzer werden in typischen Nutzungsszenarien (Nutzungsprofile) gemessen und analysiert (siehe auch [Splaine01]).

Es gibt zwei Untergruppen von Lasttests, solche mit einer realistischen Nutzeranzahl und solche, wo ein hohes Datenvolumen im Vordergrund steht (große Anzahl von parallelen/gleichzeitigen Nutzern). Lasttests messen sowohl die Antwortzeiten als auch Netzwerkdurchsatz.

#### 5.3.3.3 Stresstests

Stresstests untersuchen die Fähigkeit eines Systems, Spitzenlasten an oder über den spezifizierten Kapazitätsgrenzen zu bewältigen. Mit steigender Überbelastung sollte die Systemleistung allmählich, vorhersehbar und ohne Ausfall abnehmen. Vor allem sollte die funktionale Integrität des Systems unter Spitzenlast getestet werden, um mögliche Fehlerzustände bei der funktionalen Verarbeitung oder Dateninkonsistenzen aufzudecken.

Mögliches Ziel von Stresstests kann es sein, die Grenze zu erreichen, an der das System tatsächlich ausfällt, um so das schwächste Glied in der Kette zu identifizieren. Bei Stresstests ist es erlaubt, dem System rechtzeitig zusätzliche Komponenten hinzuzufügen (beispielsweise Speicher, CPU-Kapazität, Datenbankspeicher).

In Spitzentests (spike tests) werden Bedingungen simuliert, die zu plötzlichen extremen Systemlasten führen. Bei Bounce-Tests wechseln solche Spitzenlasten mit Zeiten geringer Nutzung ab. Die Tests untersuchen, wie gut das System mit diesen Laständerungen fertig wird, und ob es nach Bedarf Ressourcen in Anspruch nimmt und wieder freigibt (siehe auch [Splaine01]).

#### 5.3.3.4 Skalierbarkeitstests

Skalierbarkeitstests untersuchen die Fähigkeit eines Systems, zukünftige Effizienzanforderungen zu erfüllen, die über den gegenwärtigen liegen können. Ziel dieser Tests ist es zu beurteilen, ob das System wachsen kann (beispielsweise mit mehr Nutzern oder größeren Mengen von gespeicherten Daten), ohne die vereinbarten Grenzen zu überschreiten oder zu versagen. Sind diese Grenzen bekannt, lassen sich Schwellenwerte definieren und in der Produktion überwachen, sodass bei bevorstehenden Problemen eine Warnung erfolgen kann.

#### 5.3.3.5 Prüfung der Ressourcennutzung

Effizienztests der Ressourcennutzung bewerten die Verwendung von Systemressourcen (beispielsweise Hauptspeicher- und Festplattenkapazitäten, Bandbreiten im Netz). Die Verwendung

dieser Ressourcen wird sowohl bei normaler Systemauslastung gemessen als auch in Stresssituationen durch beispielsweise hohe Transaktionsraten oder große Datenmengen.

So spielt beispielsweise bei eingebetteten Echtzeitsystemen die Speichernutzung (memory footprint) eine wichtige Rolle bei Performanztests.

#### 5.3.3.6 Spezifikation von Effizienztests

Testspezifikationen der Testarten Performanz-, Last- und Stresstests basieren auf definierten Nutzungsprofilen mit verschiedenen Formen des Nutzungsverhaltens bei der Interaktion mit der Anwendung. Für eine Anwendung kann es mehrere Nutzungsprofile geben.

Die Anzahl der Nutzer pro Nutzungsprofil lässt sich mit Monitorwerkzeugen bestimmen (wenn die tatsächliche oder eine vergleichbare Anwendung bereits zur Verfügung stehen), oder durch eine Voraussage. Voraussagen können auf Algorithmen basieren oder von der Betriebsorganisation stammen, sie sind besonders wichtig für die Spezifizierung der Nutzungsprofile bei Skalierbarkeitstests.

Nutzungsprofile sind Grundlage der Testfälle und werden in der Regel mit Testwerkzeugen generiert. Simulierte Nutzer im Nutzungsprofil werden normalerweise als virtuelle Anwender bezeichnet.

## *5.3.4 Wartbarkeitstests*

Wartbarkeitstests untersuchen, wie einfach die Software analysiert, geändert und getestet werden kann. Geeignete Verfahren für Wartbarkeitstests sind statische Analysen und Checklisten.

#### 5.3.4.1 Dynamische Wartbarkeitstests

Dynamische Wartbarkeitstests untersuchen vorrangig die dokumentierten Verfahren, die für die Wartung einer Anwendung entwickelt wurden (beispielsweise für ein Update der Software). Als Testfälle dienen Wartungsszenarien, um sicherzustellen, dass die geforderten Service Levels mit den dokumentierten Verfahren erreicht werden können.

Diese Art des Testens ist besonders wichtig bei komplexen Infrastrukturen, bei denen mehrere Abteilungen/Organisationen an den Supportverfahren beteiligt sind. Die Tests können Teil der betrieblichen Abnahmetests sein [siehe www.testingstandards.co.uk].

#### 5.3.4.2 Analysierbarkeit (korrektive Wartung)

Bei dieser Art von Wartbarkeitstests wird die Zeit gemessen, die für Diagnose und Behebung von im System erkannten Problemen nötig ist. Eine einfache Metrik kann der durchschnittliche Zeitaufwand für Diagnose und Korrektur des Fehlers sein.

#### 5.3.4.3 Änderbarkeit, Stabilität und Testbarkeit (adaptive Wartung)

Die Wartbarkeit eines Systems lässt sich auch auf den Aufwand bezogen messen, der für Änderungen nötig ist (beispielsweise Programmänderungen). Da der benötigte Aufwand von mehreren Faktoren abhängt, wie der verwendeten Softwaredesign-Methodik (beispielsweise objektorientiert), von Programmierstandards usw., sind auch Analysen oder Reviews für diese Wartbarkeitstests geeignet. Die Testbarkeit hängt in erster Linie vom Aufwand für das Testen der Änderungen ab. Die Stabilität hängt in erster Linie von der Reaktion des Systems auf Änderungen ab. Systeme mit niedriger Stabilität zeigen nach jeder Änderung eine große Anzahl an Folgeproblemen (ripple effect). [ISO9126] [www.testingstandards.co.uk]

## *5.3.5 Portabilitätstests*

Portabilitätstests untersuchen, wie einfach es ist, eine Software in ihre vorgesehene Umgebung zu übertragen, entweder bei der Erstinstallation oder aus einer bestehenden Umgebung. Bei

Portabilitätstests werden Installierbarkeit, Koexistenz/Kompatibilität, Anpassbarkeit und Austauschbarkeit getestet.

#### 5.3.5.1 Installationstests

Installationstests testen Software für die Software-Installation in einer Zielumgebung. Das kann eine Software sein, mit der ein Betriebssystem auf einem Prozessor installiert wird, oder ein Installationsprogramm (Wizard), mit dem ein Produkt auf einem Client-PC verwendet wird. Ziele von Installationstests sind

- Validieren, dass die Software erfolgreich installiert werden kann, indem man die Anweisungen des Installationshandbuchs befolgt (einschließlich der Ausführung von Installationsskripten) oder einen Installationswizard nutzt. Dabei sind auch die unterschiedlichen Optionen für mögliche HW-/SW-Konfigurationen sowie für verschiedene Installationsstufen (ganz oder teilweise) zu testen.
- Testen, ob die Installationssoftware richtig mit Fehlerwirkungen umgeht, die bei der Installation auftreten (beispielsweise DLLs nicht geladen werden), ohne das System in einem undefinierten Zustand zu lassen (beispielsweise mit nur teilweise installierter Software oder inkorrekten Systemkonfigurationen).
- Testen, ob sich eine nur teilweise Installation/Deinstallation der Software abschließen lässt.
- Testen, ob ein Installationswizard eine ungültige Hardware-Plattform oder Betriebssystem-Konfigurationen erfolgreich identifizieren kann.
- Messen, ob der Installationsvorgang im spezifizierten Zeitrahmen oder mit weniger als der spezifizierten Anzahl von Schritten abgeschlossen werden kann.
- Validieren, dass sich eine frühere Version installieren oder die Software deinstallieren lässt.

Nach dem Installationstest wird normalerweise die Funktionalität getestet, um Fehler aufzudecken, die durch die Installation eingeschleust worden sein können (beispielsweise inkorrekte Konfigurationen, nicht mehr verfügbare Funktionen). Parallel zu den Installationstests laufen in der Regel Benutzbarkeitstest (beispielsweise zur Validierung, dass die Anwender bei der Installation verständliche Anweisungen und Fehler-/Meldungen erhalten).

#### 5.3.5.2 Koexistenz

Computersysteme, die nicht miteinander interagieren, sind dann kompatibel, wenn sie in derselben Umgebung (beispielsweise auf derselben Hardware) laufen können, ohne sich gegenseitig zu beeinflussen (beispielsweise durch Ressourcenkonflikte). Kompatibilitätstests können dann durchgeführt werden, wenn eine neue Software oder eine Aktualisierung in einer neuen Umgebung (beispielsweise auf einem Server) installiert wird, in der bereits Anwendungen installiert sind.

Kompatibilitätsprobleme können entstehen, wenn eine Anwendung als die einzige in einer Umgebung getestet wurde und Inkompatibilitäten nicht erkennbar waren, wenn diese Anwendung dann in einer anderen, beispielsweise der Produktionsumgebung, installiert wird, in der bereits andere Anwendungen installiert sind.

Typische Ziele von Kompatibilitätstests sind

- Bewertung möglicher negativer Auswirkungen auf die Funktionalität, wenn Anwendungen in derselben Umgebung geladen werden (beispielsweise Konflikte in der Ressourcennutzung, wenn mehrere Anwendungen auf demselben Server laufen),
- Bewertung der Auswirkungen auf jede Anwendung, die sich aus Modifikationen oder dem Installieren einer aktuelleren Betriebssystemversion ergeben.
- Kompatibilitätstests werden normalerweise nach dem erfolgreichen Abschluss der System- und Anwenderabnahmetests durchgeführt.

### 5.3.5.3 Anpassbarkeitstests

Anpassbarkeitstests sollen zeigen, ob eine bestimmte Anwendung in allen geplanten Zielumgebungen korrekt funktioniert (Hardware, Software, Middleware, Betriebssystem usw.). Für die Spezifikation der Anpassbarkeitstests müssen Kombinationen der beabsichtigten Zielumgebungen identifiziert, konfiguriert und dem Testteam zur Verfügung gestellt werden. Diese Umgebungen werden dann mit ausgewählten funktionalen Testfällen getestet, bei denen die verschiedenen Komponenten der Testumgebung geprüft werden.

Anpassbarkeit kann sich darauf beziehen, dass sich die Software durch Ausführung eines definierten Verfahrens auf verschiedene spezifizierte Umgebungen portieren lässt. Beim Testen kann das Verfahren bewertet werden.

Anpassbarkeitstests können zusammen mit Installierbarkeitstests durchgeführt werden. Normalerweise finden anschließend funktionale Tests statt, um Fehler aufzudecken, die durch das Anpassen der Software an die andere Umgebung entstanden sein können.

### 5.3.5.4 Austauschbarkeitstests

Austauschbarkeit drückt aus, ob sich Softwarekomponenten eines Systems gegen andere Komponenten austauschen lassen. Das ist besonders relevant bei Systemen, die kommerzielle Standardsoftwareprodukte für bestimmte Softwarekomponenten verwenden.

Austauschbarkeitstests können parallel zu den funktionalen Integrationstests durchgeführt werden, wenn mehr als eine Komponente alternativ für die Integration in das Gesamtsystem verfügbar ist. Die Austauschbarkeit lässt sich in technischen Reviews oder Inspektionen bewerten, bei denen Gewicht auf eine klare Definition der Schnittstellen zu möglichen Austauschkomponenten gelegt wird.

# 6. Review

*Begriffe*

Audit, IEEE 1028, informelles Review, Inspektion, Leiter einer Inspektion, Management-Review, Moderator, Review, Prüfer, technisches Review, Walkthrough.

## 6.1 Einführung

Zu einem erfolgreichen Review-Prozess gehören das Planen von Reviews, die Durchführung und die Nachbereitung. Ausbildungsanbieter müssen dafür sorgen, dass die Testmanager ihre Verantwortung bei den Planungs- und Nachbereitungsaufgaben verstehen. Tester müssen aktiv am Review-Prozess teilnehmen und ihre individuelle Perspektive einbringen. Sie sollten ein formales Review-Training erhalten haben, damit sie ihre jeweiligen Rollen bei einem technischen Review-Prozess besser verstehen. Alle Teilnehmerinnen und Teilnehmer müssen vom Nutzen gut durchgeführter Reviews überzeugt sein. Wenn sie ordnungsgemäß durchgeführt werden, leisten Reviews nicht nur den größten einzelnen, sondern auch den kosteneffektivsten Beitrag zur gelieferten Qualität. Ein internationaler Standard für Reviews ist IEEE 1028.

## 6.2 Grundsätze von Reviews

Ein Review ist ein statisches Testverfahren. Hauptziel des Reviews ist oft das Aufdecken von Fehlerzuständen. Die Prüfer finden Fehler, indem sie die Dokumente direkt untersuchen.

Weitere Informationen zu den grundlegenden Review-Arten enthalten Abschnitt 3.2 des Foundation-Level-Lehrplans (Version 2007) und der folgende Abschnitt 6.3.

Alle Review-Arten werden am besten ausgeführt, sobald die relevanten Quellendokumente (Dokumente, in denen die Projektanforderungen spezifiziert sind) und die Standards (die im Projekt einzuhalten sind) zur Verfügung stehen. Fehlt noch ein Dokument oder ein Standard, so lassen sich Fehler und Inkonsistenzen in der Gesamtdokumentation nicht aufdecken, sondern nur in einem einzelnen Dokument. Den Prüfern muss das zu prüfende Dokument mit einem geeigneten Vorlauf zur Verfügung gestellt werden, sodass sie ausreichend Zeit haben, um sich damit vertraut zu machen.

Jede Dokumentart kann Gegenstand eines Reviews sein, beispielsweise Quellcode, Anforderungsspezifikationen, Konzepte, Testkonzepte, Testdokumentationen usw. Das dynamische Testen findet normalerweise nach dem Review des Quellcodes statt und sucht die Fehler, die sich in der statischen Prüfung nicht finden lassen.

Ein Review kann drei mögliche Ergebnisse haben:

- Das Dokument lässt sich ohne oder mit nur geringfügigen Änderungen verwenden.
- Das Dokument muss geändert werden, aber es ist kein weiteres Review nötig.
- Das Dokument muss gründlich geändert werden, und ein weiteres Review ist nötig.

Rollen und Verantwortlichkeiten der Teilnehmer eines typischen formalen Reviews beschreibt der Foundation-Level-Lehrplan. Es sind Manager, Moderator oder Leiter, Autor, Prüfer und Protokollant. Außerdem können Entscheidungsträger oder Betroffene und Vertreter der Kunden oder Nutzer am Review teilnehmen. Eine zusätzliche optionale Rolle bei einigen Inspektionen ist die des Vorlesers. Er oder sie soll Textpassagen aus den Abschnitten der Arbeitsergebnisse in der Sitzung mit eigenen Worten ausdrücken und zusammenfassen. Zusätzlich zu den erwähnten Rollen beim Review können

einzelne Prüfer jeweils eine bestimmte fehlerorientierte Aufgabe übernehmen und sich bei der Suche auf bestimmte Fehlerarten konzentrieren.

Auf ein einzelnes Produkt lassen sich mehrere Review-Arten anwenden. So kann beispielsweise ein Team in einem technischen Review entscheiden, welche Funktionalitäten in der nächsten Iteration implementiert werden sollen. Danach könnte eine Inspektion der Spezifikationen für die eingearbeiteten Funktionalitäten durchgeführt werden.

## 6.3 Review-Arten

Im Foundation-Level-Lehrplan wurden folgende Review-Arten eingeführt:

- Informelles Review
- Walkthrough
- Technisches Review
- Inspektion

In der Praxis können auch Mischformen aus diesen Review-Arten vorkommen, wie ein technisches Review, das einen Satz von Regeln nutzt.

### *6.3.1 Management-Review und Audit*

Zusätzlich zu den im Foundation-Level-Lehrplan eingeführten Review-Arten beschreibt IEEE Standard 1028 auch die folgenden:

- Management-Review
- Audit

Schlüsselmerkmale eines Management-Reviews:

- Hauptzwecke sind: Fortschritt überwachen, Status beurteilen, Entscheidungen über zukünftige Maßnahmen treffen.
- Durchgeführt wird es von Managern bzw. für Manager mit direkter Verantwortung für das Projekt oder System.
- Außerdem wird es durchgeführt von bzw. für einen Betroffenen oder Entscheidungsträger, beispielsweise gehobenes Management oder Direktor.
- Es wird geprüft, ob Pläne konsistent eingehalten werden oder es Abweichungen gibt, und es wird festgestellt, ob die installierten Managementverfahren adäquat sind.
- Es enthält die Bewertung von Projektrisiken.
- Das Ergebnis nennt die durchzuführenden Maßnahmen und zu lösenden Probleme.
- Es wird erwartet, dass sich die Teilnehmer auf die Sitzungen vorbereiten, Entscheidungen werden dokumentiert.

Hinweis: Testmanager sollten an den Management-Reviews teilnehmen bzw. Management-Reviews zum Testfortschritt ansetzen.

Audits sind sehr formal und werden in der Regel dann durchgeführt, wenn Konformität mit bestimmten Erwartungen nachgewiesen werden soll, beispielsweise Konformität mit einem geltenden Standard oder einer vertraglichen Verpflichtung. Audits sind daher die am wenigsten effektive Methode zum Aufdecken von Fehlerzuständen.

Schlüsselmerkmale eines Audits:

- Hauptzweck ist die unabhängige Bewertung der Konformität mit bestimmten Verfahren, Vorschriften, Standards usw.
- Der Leiter des Audits ist für das Audit verantwortlich und übernimmt die Moderation.

- Die Prüfer sammeln Beweise für die Konformität durch Interviews, Beobachtungen und die Prüfung von Dokumenten.
- Zu den Ergebnissen des Audits gehören Beobachtungen, Empfehlungen, Abhilfemaßnahmen und eine abschließende Beurteilung (bestanden/nicht bestanden).

### 6.3.2 Reviews von bestimmten Arbeitsergebnissen

Reviews lassen sich anhand ihrer Arbeitsergebnisse oder der Aktivitäten benennen, die geprüft werden, beispielsweise:

- Review des Vertrags
- Review der Anforderungen
- Review des Designs
  - Preliminary Design Review
  - Critical Design Review
- Abnahme-Review/Qualifizierungs-Review
- Betriebsbereitschafts-Review

Ein Review des Vertrags kann im Zusammenhang mit einem vertraglich festgelegten Meilenstein durchgeführt werden; typischerweise ist es ein Management-Review für sicherheitskritische oder sicherheitsrelevante Systeme. Teilnehmer sind Manager, Kunden und technische Mitarbeiter.

Ein Review der Anforderungen kann ein Walkthrough, ein technisches Review oder eine Inspektion sein. Das Review kann Anforderungen an die Sicherheit und an die Systemstabilität, sowie funktionale und nicht-funktionale Anforderungen prüfen. Ein Review der Anforderungen kann auch Abnahmekriterien und Testbedingungen enthalten.

Reviews des Designs sind normalerweise technische Reviews oder Inspektionen, an denen technisches Personal und Kunden oder Betroffene teilnehmen. Im Preliminary Design Review werden erste Ansätze für technischen Entwurf und Tests geprüft, während das Critical Design Review alle vorgeschlagenen Entwurfslösungen, einschließlich Testfällen und –szenarien abdeckt.

Bei Abnahme-Reviews geht es um die Genehmigung des Systems durch das Management. Sie werden auch als Qualifizierungs-Reviews bezeichnet und sind normalerweise entweder Management-Reviews oder Audits.

### 6.3.3 Formales Review durchführen

Im Foundation-Level-Lehrplan werden die sechs Phasen eines formalen Reviews beschrieben: Planung, Kick-off, Individuelle Vorbereitung, Review-Sitzung, Überarbeitung und Nachbereitung. Für ein Arbeitsergebnis, das Gegenstand des Reviews ist, sollte ein Prüfer mit geeigneter Qualifikation ausgewählt werden, beispielsweise ein Testkonzept für einen Testmanager, fachliche Anforderungen oder ein Testentwurf für einen Test Analyst, und eine Funktionsspezifikation, Testfälle oder Testskripte für einen Technical Test Analyst.

## 6.4 Einführung von Reviews

Zur erfolgreichen Einführung von Reviews in einer Organisation sind die folgenden Schritte notwendig (nicht unbedingt in dieser Reihenfolge):

- Unterstützung durch das Management sicherstellen
- Information der Manager über Kosten, Nutzen und Implementierungsfragen

- Auswahl und Dokumentieren von Review-Verfahren, -templates und –infrastruktur (beispielsweise der Datenbank für Review-Metriken)
- Review-Techniken und -verfahren trainieren
- Um Unterstützung der Mitarbeiter werben, die die Reviews durchführen werden oder deren Arbeitsergebnisse in Reviews geprüft werden
- Pilot-Reviews abhalten
- Nutzen von Reviews in Form von Kosteneinsparungen darstellen
- Reviews für die wichtigsten Dokumente durchführen, beispielsweise Anforderungen, Vertrag, Planungsdokumente usw.

Wie erfolgreich die Einführung von Reviews ist, lässt sich anhand vom Metriken feststellen, wie die Reduzierung oder Vermeidung von Kosten für das Beheben von Fehlerwirkungen und/oder deren Folgen, die durch frühes Aufdecken von Fehlerzuständen und Beheben von Fehlerwirkungen eingespart werden. Einsparungen lassen sich auch in Zeiteinheiten messen, die durch frühes Aufdecken und Beheben von Fehlerwirkungen eingespart werden.

Review-Prozesse sollten kontinuierlich überwacht und im Lauf der Zeit optimiert werden. Manager sollten sich bewusst machen, dass das Erlernen einer neuen Review-Technik eine Investition ist, deren Nutzen sich nicht sofort auszahlt, sondern im Laufe der Zeit deutlich wächst.

## 6.5 Erfolgsfaktoren für Reviews

Es gibt eine ganze Reihe von Faktoren, die zum Erfolg von Reviews beitragen. Reviews sind zwar nicht schwierig in ihrer Durchführung, können aber ihr Ziel verfehlen, wenn diese Faktoren nicht beachtet werden.

**Technische Faktoren**

- Stellen Sie sicher, dass der für die Art des Reviews definierte Prozess korrekt befolgt wird, besonders bei den eher formalen Review-Arten wie beispielsweise Inspektionen.
- Halten Sie die Kosten der Reviews (einschließlich des Zeitaufwands) und den erzielten Nutzen fest.
- Prüfen Sie in Reviews vorläufige Entwürfe und Teildokumente, um Fehlermuster zu identifizieren, bevor sie in das Gesamtdokument eingehen.
- Stellen Sie sicher, dass das Dokument oder Teildokument reif für ein Review ist, bevor Sie mit dem Review-Prozess beginnen (wenden Sie Eingangsbedingungen an).
- Verwenden Sie organisationsspezifische Checklisten für übliche Fehler.
- Setzen Sie je nach Zielsetzung ggf. mehr als nur eine Review-Art ein, beispielsweise für die endgültige Fertigstellung des Dokuments, technische Verbesserung, Informationsaustausch oder Fortschrittsmanagement.
- Prüfen Sie vor einem Management-Review zur Bewilligung wichtiger Projektbudgets alle Dokumente durch Review oder Inspektion, die Grundlage für wichtige Entscheidungen sind, beispielsweise Angebot, Vertrag oder übergeordnete Anforderungen.
- Untersuchen Sie versuchsweise einen begrenzten Teil des Dokuments.
- Ermutigen Sie dazu, die wichtigsten Fehlerzustände aufzuspüren, und konzentrieren Sie sich dabei auf den Inhalt und nicht auf das Format.
- Verbessern Sie den Review-Prozess kontinuierlich.

**Organisatorische Faktoren**

- Stellen Sie sicher, dass das Management ausreichend Zeit für die Review-Aktivitäten bewilligt, auch unter Termindruck.
- Denken Sie daran: Zeit- und Kostenaufwand sind nicht proportional zur Menge der gefundenen Fehler.
- Planen Sie ausreichend Zeit ein für die Nacharbeiten zum Beheben identifizierter Fehler.
- Verwenden Sie die Metriken der Reviews niemals für personenbezogene Leistungsbeurteilungen.
- Stellen Sie sicher, dass die richtigen Personen an den jeweiligen Review-Arten beteiligt sind.
- Stellen Sie Schulungen in Review-Techniken zur Verfügung, besonders für die formaleren Arten von Reviews.
- Unterstützen Sie ein Forum für Review-Leiter zum Austausch von Erfahrungen und Ideen.
- Stellen Sie sicher, dass alle an Reviews teilnehmen, und dass alle ein Review ihrer Dokumente bekommen.
- Nutzen Sie für die wichtigsten Dokumente die formalsten Review-Techniken.
- Stellen Sie sicher, dass das Review-Team ausgewogen mit Personen unterschiedlicher Fähigkeiten und Erfahrungen besetzt ist.
- Unterstützen Sie Maßnahmen zur Prozessverbesserung, um systemische Probleme zu überwinden.
- Erkennen Sie die Verbesserungen an, die durch den Review-Prozess erreicht wurden.

**Personenbezogene Aspekte**

- Machen Sie den Betroffenen klar, dass bei einem Review Fehler zu erwarten sind, und dass Zeit für Nacharbeit und Wiederholung des Reviews einzuplanen ist.
- Stellen Sie sicher, dass das Review für die Autorin/den Autor eines Dokuments eine positive Erfahrung ist.
- Begrüßen Sie die Fehleridentifikation in einer offenen Atmosphäre ohne Schuldzuweisungen.
- Stellen Sie sicher, dass Kommentare konstruktiv, hilfreich und objektiv sind, und nicht subjektiv.
- Führen Sie kein Review durch, wenn der Autor nicht zustimmt oder nicht dazu bereit ist.
- Ermutigen Sie alle dazu, sich mit den wichtigsten Aspekten des zu prüfenden Dokuments eingehend auseinanderzusetzen.

Für weitere Informationen über Reviews und Inspektionen siehe [Gilb93].

# 7. Fehler- und Abweichungsmanagement

*Begriffe*

Abweichung, Abweichungsprotokollierung, Anomalie, Fehlerwirkung, Fehlerzustand, Fehlhandlung, Grundursachenanalyse, IEEE 829, IEEE 1044, IEEE 1044.1, Configuration Control Board, Priorität, Schweregrad.

## 7.1 Einführung

Testmanager und Tester müssen mit dem Prozess des Fehler- und Abweichungsmanagements vertraut sein. Testmanager konzentrieren sich vor allem auf diesen Prozess, mit Methoden zum Erkennen, Verfolgen und Beheben von Fehlerzuständen. Tester dagegen befassen sich vor allem damit, Probleme richtig aufzuzeichnen, die sie in ihren jeweiligen Testbereichen finden. Für jeden Schritt im Lebenszyklus haben Test Analysts und Technical Test Analysts eine unterschiedliche Ausrichtung. Test Analysts bewerten das Verhalten des Systems aus der geschäftlichen und Anwenderperspektive, beispielsweise: Würden Nutzer wissen, was zu tun ist, wenn diese Meldung erscheint, oder wenn das System sich so verhält? Technical Test Analysts bewerten das Verhalten der Software selbst, sie werden das Problem eher unter technischen Gesichtspunkten untersuchen, indem sie beispielsweise die gleiche Fehlerwirkung auf verschiedenen Plattformen oder mit unterschiedlichen Speicherkonfigurationen testen.

## 7.2 Wie lässt sich ein Fehlerzustand aufdecken?

Eine Abweichung ist ein unerwartetes Ereignis, das näher untersucht werden muss. Das Erkennen einer Fehlerwirkung, die durch einen Fehlerzustand verursacht wurde, ist eine Abweichung. Eine Abweichung muss kein Fehler sein und kann zur Erstellung eines Abweichungsberichts führen oder nicht. Ein Fehlerzustand ist ein tatsächliches Problem, das festgestellt wurde und durch eine Änderung gelöst werden muss.

Ein Fehlerzustand lässt sich durch statische Tests finden; eine Fehlerwirkung dagegen nur durch dynamisches Testen. Für jede Phase des Softwarelebenszyklus sollte es eine Methode zum Erkennen und Beseitigen möglicher Fehlerwirkungen geben. Während der Entwicklungsphase sollten das beispielsweise Code-Reviews und Reviews des Designs zum Aufdecken von Fehlerzuständen sein. Bei dynamischen Tests werden Testfälle zum Finden von Fehlerwirkungen verwendet. Je früher eine Fehlerwirkung erkannt und korrigiert wird, desto geringer sind die Kosten der Qualität für das Gesamtsystem. Hinweis: Fehlerwirkungen können sowohl im Testobjekt als auch in den Testmitteln auftreten.

## 7.3 Fehlerlebenszyklus

Alle Fehlerzustände haben einen Lebenszyklus, auch wenn einige Fehler nicht alle Phasen durchlaufen. Der Fehler- und Abweichungslebenszyklus (gemäß IEEE 1044-1993) besteht aus vier Schritten:

- Schritt 1: Erkennung (Recognition)
- Schritt 2: Analyse (Investigation)

- Schritt 3: Bearbeitung3 (Action)
- Schritt 4: Abschluss (Disposition)

In jedem dieser Schritte gibt es drei Aktivitäten zur Informationserfassung:

- Aufzeichnen
- Klassifizieren
- Auswirkungen identifizieren

### 7.3.1 Schritt 1: Erkennung (Recognition)

Der Erkennungsschritt beginnt, wenn ein möglicher Fehler (eine Abweichung) entdeckt wird. Das kann in jeder Phase des Softwarelebenszyklus sein. Zum Zeitpunkt der Entdeckung werden die Datenelemente aufgezeichnet, die den Fehler kennzeichnen. Dazu gehören Informationen über die Umgebung, in der der Fehler beobachtet wurde, meldende Organisation oder Person, Beschreibung, Zeitpunkt und ggf. Anbieter. Beim Erkennungsschritt wird der möglichen Fehler anhand bestimmter Merkmale klassifiziert, wie Projektaktivität, Projektphase, vermutete Ursache, Wiederholbarkeit, Symptom(e) und Produktstatus als Folge der Anomalie. Mit diesen Informationen lassen sich die Auswirkungen anhand des Fehlerschweregrads und der Folgen für Projektzeitplan und Projektkosten bewerten.

### 7.3.2 Schritt 2: Analyse (Investigation)

Nach der Erkennung werden mögliche Fehlerzustände untersucht. Dieser Schritt wird dazu genutzt, eventuelle weitere Probleme in diesem Zusammenhang aufzudecken und Lösungsvorschläge zu erarbeiten. Eine Lösung kann auch sein, keine Maßnahmen einzuleiten (wenn beispielsweise der potenzielle Fehler nicht mehr als tatsächlicher Fehler betrachtet wird). In diesem Schritt werden zusätzliche Daten aufgezeichnet und Fehlerklasse und Auswirkungen aus dem vorherigen Schritt erneut bewertet.

### 7.3.3 Schritt 3: Bearbeitung (Action)

Der Maßnahmenschritt arbeitet mit den Ergebnissen der Untersuchung. Sie erstreckt sich sowohl auf Maßnahmen, die zur Behebung des Fehlers notwendig sind, als auch auf Maßnahmen, mit denen sich Prozesse und Methode überprüfen und verbessern lassen, damit ähnliche Fehler in Zukunft nicht mehr vorkommen. Nach jeder Änderung müssen Regressions- und Fehlernachtests vorgenommen und der Testfortschritt kontrolliert werden. In diesem Schritt werden zusätzliche Daten aufgezeichnet und Fehlerklasse und Auswirkungen aus dem vorigen Schritt erneut bewertet.

### 7.3.4 Schritt 4: Abschluss (Disposition)

Die Abweichung erreicht damit den Abschlussschritt, in der etwaige weitere Datenelemente aufgezeichnet werden und der Abschluss klassifiziert wird als: geschlossen, zurückgestellt, zusammengefasst oder an ein anderes Projekt verwiesen.

## 7.4 Pflichtfelder für die Erfassung von Fehlern und Abweichungen

IEEE Standard 1044-1993 legt eine Reihe von Pflichtfeldern fest, die zu verschiedenen Zeiten im Fehlerlebenszyklus gefüllt werden. Er definiert auch eine große Menge optionaler Felder. Gemäß IEEE Standard 1044.1 ist es bei der Implementierung von IEEE Standard 1044-1993 zulässig,

[3] Synonym: Behebung

Begriffe in einem Unternehmen transparent auf die Begriffe des IEEE-Standards für die Abweichungsfelder abzubilden. IEEE Standard 1044-1993 lässt sich so einhalten, ohne dass man sich bis ins Detail an die vorgegebene Terminologie halten muss. Die Konformität mit dem IEEE-Standard erlaubt den Vergleich von Informationen über Abweichungen zwischen unterschiedlichen Unternehmen und Organisationen.

Unabhängig davon, ob die Konformität mit dem IEEE-Standard ein Projektziel ist, sollen die Abweichungsfelder ausreichend Informationen für entsprechende Maßnahmen liefern. Ein für Maßnahmen geeigneter Abweichungsbericht ist

- vollständig,
- kurz und prägnant,
- richtig,
- objektiv.

Der Bericht muss nicht nur das Beheben des spezifischen Fehlerzustands ermöglichen, er muss auch die notwendigen Informationen für eine korrekte Klassifizierung des Fehlerzustands, eine Risikoanalyse und ggf. eine Prozessverbesserung enthalten.

## 7.5 Metriken und Abweichungsmanagement

Die Informationen über Abweichungen müssen geeignet sein für die Testfortschrittsüberwachung, die Analyse der Fehlerdichte, Messungen gefundener gegenüber behobenen Fehlerzuständen und Konvergenzmetriken (offen/geschlossen). Außerdem sollen Abweichungsinformationen die Initiativen zur Prozessverbesserung unterstützen, indem sie die phasenspezifischen Informationen verfolgen; sie sollen eine Grundursachenanalyse liefern und Fehlertendenzen als Input für strategische Maßnahmen zur Risikobeherrschung identifizieren.

## 7.6 Abweichungen kommunizieren

Das Abweichungsmanagement braucht eine effektive Kommunikation ohne gegenseitige Schuldzuweisungen. Es unterstützt das Sammeln und Interpretieren von objektiven Informationen. Damit die Beziehungen zwischen den Menschen, die Fehler berichten, und denjenigen, die sie beheben, professionell bleiben, müssen Abweichungsberichte korrekt sein, die Klassifizierungen stimmen, und unverkennbare Objektivität ist nötig. Tester können gefragt werden, welche Wichtigkeit ein Fehlerzustand hat, und sollten dann objektiv die verfügbaren Informationen beisteuern.

Besprechungen zur Abweichungsbeurteilung (triage meetings) können bei einer angemessenen Priorisierung helfen. Ein Fehlerverfolgungs-Werkzeug kann und sollte eine gute Kommunikation nicht ersetzen. Genauso wenig sollten aber die Besprechungen zur Abweichungsbeurteilung als Ersatz für ein gutes Fehlerverfolgungs-Werkzeug dienen. Ein effektiver Fehlerverfolgungs-Prozess braucht beides: Kommunikation und eine angemessene Unterstützung durch Werkzeuge.

# 8. Standards in der Testprozess-Verbesserung

*Begriffe*

Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI), Test Maturity Model (TMM), Test Maturity Model Integrated (TMMi), Test Process Improvement (TPI).

## 8.1 Einführung

Verschiedene Quellen unterstützen die Einführung und kontinuierliche Verbesserung von Testprozessen. Dieses Kapitel befasst sich zunächst mit den Standards, die eine nützliche und manchmal obligatorische Informationsquelle für eine ganze Reihe von Testthemen sind. Für Testmanager und Test Analysts ist es ein Lernziel zu wissen, welche Standards es gibt und wo oder wie sie anzuwenden sind. Ausbildungsanbieter sollten die für das jeweilige Trainingsmodul besonders relevanten Standards herausstellen.

Ist ein Testprozess eingeführt, sollte er kontinuierlich verbessert werden. Abschnitt 8.3 beleuchtet zunächst allgemeine Themen zur Testprozessverbesserung; es folgt eine Einführung in einige spezifische Modelle für die Testprozessverbesserung. Testmanager müssen den gesamten Inhalt dieses Abschnitts verstehen, aber auch für Test Analysts und Technical Test Analysts ist es wichtig zu wissen, welche Testprozessverbesserungs-Modelle es gibt, weil Test Analysts und Technical Test Analysts eine Schlüsselrolle bei der Implementierung der Verbesserungen spielen.

## 8.2 Normen und Standards

Dieser Lehrplan nennt einige Standards, wie schon der Foundation-Level-Lehrplan. Es gibt Standards zu vielen Softwarethemen:

- Lebenszyklus der Software und in der Softwareentwicklung
- Softwaretesten und Methoden
- Softwarekonfigurationsmanagement
- Softwarewartung
- Qualitätssicherung
- Projektmanagement
- Anforderungen
- Programmiersprachen
- Softwareschnittstellen
- Abweichungsmanagement

Es ist nicht Sinn und Zweck dieses Lehrplans, bestimmte Standards aufzulisten oder zu empfehlen. Tester sollten aber wissen, wie Standards entstehen und wie sie in der Anwendungsumgebung einzusetzen sind.

Standards können aus unterschiedlichen Quellen stammen:

- Internationale Standards oder solche mit internationalen Zielsetzungen
- Nationale Standards oder nationale Anwendung internationaler Standards
- Branchenspezifische Standards, für die internationale oder nationale Standards für bestimmte Industriezweige angepasst oder eigene Standards für bestimmte Branchen entwickelt wurden

Bei der Anwendung von Standards sind verschiedene Aspekte zu beachten, die in diesem Abschnitt genauer beschrieben sind.

## 8.2.1 Allgemeine Aspekte von Standards

### 8.2.1.1 Quellen der Standards

Standards werden von Expertinnen und Experten erstellt und spiegeln ihr kollektives Wissen wider. Es gibt zwei Hauptquellen internationaler Standards: IEEE und ISO (siehe 8.2.2). Darüber hinaus stehen wichtige nationale und branchenspezifische Standards zur Verfügung.

Tester sollten wissen, welche Standards für ihr Testumfeld und ihren Testkontext zutreffen. Das können formale Standards sein (international, national oder branchenspezifisch), oder unternehmensinterne Standards und empfohlene Praktiken. Da Standards sich weiterentwickeln und ändern, ist es für die Konformität wichtig, immer die jeweils gültige Version des Standards anzugeben (Datum der Veröffentlichung). Fehlt Datum oder Version in einem Verweis auf einen Standard, geht man von der aktuellen Version aus.

### 8.2.1.2 Nützlichkeit von Standards

Standards unterstützen die konstruktive Qualitätssicherung, die das Minimieren von Fehlern anstrebt und weniger ihre Aufdeckung beim Testen (wie die analytische Qualitätssicherung). Nicht alle Standards sind für alle Projekte relevant; Informationen in einem Standard können für ein Projekt nützlich sein, aber auch hinderlich. Wenn Tester einen Standard um des Standards willen einhalten, wird ihnen das nicht helfen, mehr Fehler in einem Arbeitsprodukt aufzudecken [Kaner02]. Normen und Standards können aber ein Referenzrahmen sein und eine Basis für die Definition von Testlösungen bieten.

### 8.2.1.3 Konsistenz und Konflikte

Einige Standards lassen eine Konformität mit anderen Standards vermissen oder geben widersprüchliche Definitionen. Die Anwender von Standards entscheiden, wie nützlich ein Standard in der eigenen Umgebung und im Kontext ist.

## 8.2.2 Internationale Standards

### 8.2.2.1 ISO

ISO [www.iso.org] ist die Abkürzung für: International Standards Organization (kürzlich umbenannt in International Organization for Standardization). ISO setzt sich zusammen aus Mitgliedern verschiedener nationaler Organisationen, die für die Standardisierung in ihrem Land repräsentativ sind. Die internationale Organisation hat einige hilfreiche Standards und Normen für Softwaretester herausgegeben:

- ISO 9126:1998, jetzt aufgeteilt in die folgenden Standards und technischen Berichte (TR, Technical Report):
  - ISO/IEC 9126-1:2001 Software engineering – Product quality – Part 1: Quality model
  - ISO/IEC TR 9126-2:2003 Software engineering – Product quality – Part 2: External metrics
  - ISO/IEC TR 9126-3:2003 Software engineering - Product quality – Part 3: Internal metrics
  - ISO/IEC TR 9126-4:2004 Software engineering – Product quality – Part 4: Quality in use metrics
- ISO 12207:1995/Amd 2:2004 Systems and Software Engineering – Software Lifecycle Processes
- ISO/IEC 155044-2:2003 Information technology – Process assessment – Part 2: Performing an assessment

[4] ISO 15504 ist auch bekannt unter dem Begriff SPICE, der vom SPICE-Projekt abgeleitet wurde.

Die Liste ist nicht vollständig; weitere ISO-Standards können für einen bestimmten Testkontext und -umgebung gelten.

#### 8.2.2.2 IEEE

IEEE [www.ieee.org] ist die Abkürzung für: Institute of Electrical and Electronics Engineers, einen Berufsverband mit Sitz in den Vereinigten Staaten. Nationale Repräsentanten dieser Organisation gibt es in mehr als hundert Ländern. IEEE hat einige hilfreiche Standards und Normen für Softwaretester herausgegeben:

- IEEE 610:1991 IEEE Standard computer dictionary, eine Zusammenstellung von IEEE-Standard-Computerglossaren
- IEEE 829:1998 IEEE Standard for software test documentation
- IEEE 1028:1997 IEEE Standard for software reviews
- IEEE 1044:1995 IEEE Guide to classification for software anomalies

Die Liste ist nicht vollständig; weitere IEEE-Standards können für einen bestimmten Testkontext und -umgebung gelten.

### *8.2.3 Nationale Standards*

Viele Länder haben eigene spezifische Standards, von denen einige für das Softwaretesten anwendbar und/oder nützlich sind. Der britische BS 7925-2:1998 „Software testing. Software component testing“ ist ein solcher Standard. Er liefert Informationen für viele der in diesem Lehrplan erläuterten Testverfahren:

- Äquivalenzklassenbildung
- Grenzwertanalyse
- Zustandsbasierter Test
- Ursache-Wirkungs-Graph-Analyse
- Anweisungstest
- Zweigtest/Entscheidungsüberdeckungstest
- Bedingungstest
- Zufallstest

BS 7925-2 enthält auch eine Beschreibung für das Komponententesten.

### *8.2.4 Branchenspezifische Standards*

Standards gibt es auch für verschiedene technische Bereiche. Manche Branchen passen andere Standards an die eigene technische Fachrichtung an. Auch hier gibt es interessante Aspekte für das Softwaretesten, für Softwarequalität und -entwicklung.

#### 8.2.4.1 Avioniksysteme

Der branchenspezifische Standard RTCA DO-178B/ED 12B „Software Considerations in Airborne Systems and Equipment Certification” ist anwendbar auf Software für die zivile Luftfahrt. Der Standard gilt auch für Software, mit der Software für die zivile Luftfahrt erstellt oder verifiziert wird. Die United States Federal Aviation Administration (FAA) und die internationale Joint Aviation Authorities (JAA) schreiben für Avionik-Software bestimmte strukturelle Abdeckungskriterien vor, je nach Kritikalität der zu testenden Software:

| Kritikalität | Mögliche Auswirkung bei Versagen | Benötigte strukturelle Überdeckung |
|---|---|---|

| Kritikalität | Mögliche Auswirkung bei Versagen | Benötigte strukturelle Überdeckung |
|---|---|---|
| A Katastrophal | • Verhindert die sichere Fortsetzung des Flugs und die Landung | Definierte Bedingungsüberdeckung, Entscheidungen und Anweisungen |
| B gefährlich/ schwer- wiegend bis bedeutend | • große Verminderung von Sicherheit und Funktionsfähigkeit<br>• Besatzung kann möglicherweise ihre Aufgaben nicht korrekt und vollständig ausführen<br>• schwere Verletzungen oder Verletzungen mit Todesfolge bei einer kleinen Zahl von Flugzeuginsassen | Entscheidungen und Anweisungen |
| C bedeutend | • bedeutende Verminderung der Sicherheit<br>• bedeutend erhöhte Arbeitsbelastung der Besatzung<br>• Unannehmlichkeiten für Flugzeuginsassen, einschließlich Verletzungen | Anweisungen |
| D geringfügig | • leichte Verminderung der Sicherheitsmargen und der Funktionsfähigkeit<br>• leicht erhöhte Arbeitsbelastung der Besatzung<br>• einige Unannehmlichkeiten für die Flugzeuginsassen | keine |
| E keine Auswirkung | • keine Auswirkung auf die Funktionsfähigkeit des Flugzeugs<br>• keine erhöhte Arbeitsbelastung der Besatzung | keine |

Das angemessene Niveau struktureller Überdeckung hängt ab von der Kritikalitätsstufe der Software, die für den Einsatz in der zivilen Luftfahrt zu zertifizieren ist.

#### 8.2.4.2 Raumfahrtindustrie

Einige Industriezweige nutzen abgeänderte und andere angepasste Standards für die eigene Branche. Das hat die Raumfahrtindustrie mit ihrem Standard ECSS (European Cooperation on Space Standardization, [www.ecss.org]) getan. Je nach Kritikalität der Software empfiehlt der ECSS-Standard Methoden und Verfahren, die mit den ISTQB® Lehrplänen für Foundation und Advanced Level übereinstimmen, u.a:

- SFMECA - Software-Fehlermöglichkeits-, Einfluss- und Kritikalitäts-Analyse
- SFTA - Softwarefehlerbaum-Analyse
- HSIA - Hardware Software Interaction Analysis (Hardware-Software-Interaktionsanalyse)
- SCCFA - Software Common Cause Failure Analysis (Analyse der gemeinsamen Ursachen von Ausfällen)

#### 8.2.4.3 Food & Drug Administration (FDA)

- Für medizinische Systeme, die Title 21 CFR Part 820 unterliegen, empfiehlt die US-amerikanische Bundesbehörde zur Überwachung von Nahrungs- und Arzneimitteln (FDA) bestimmte strukturelle und funktionale Testverfahren.

Außerdem empfiehlt die FDA Teststrategien und –grundsätze, die mit den ISTQB® Lehrplänen für Foundation und Advanced Level übereinstimmen.

### *8.2.5 Sonstige Standards*

Es gibt sehr viele Standards für die verschiedenen Industriezweige. Einige sind Anpassungen an die eigene Branche oder das eigene Fachgebiet; andere gelten für bestimmte Aufgaben oder erklären, wie ein Standard anzuwenden ist.

Tester müssen die verschiedenen Standards (einschließlich firmeninterner Standards, empfohlene Praktiken usw.) in ihrer Branche, ihrem Fachbereich oder Kontext kennen. Manchmal sind die gültigen Standards hierarchisch nach ihrem Geltungsbereich für bestimmte Verträge festgelegt. Testmanager müssen wissen, welche Standards eingehalten werden müssen, und sie müssen auf angemessene Konformität achten.

## 8.3 Testverbesserungs-Prozess

So wie das Testen dazu dient, die Software zu verbessern, werden Softwarequalitäts-Prozesse gewählt und eingesetzt, um den Softwareentwicklungs-Prozess (und die daraus resultierenden Arbeitsergebnisse) zu verbessern. Prozessverbesserung ist auch für Testprozesse möglich. Es gibt unterschiedliche Möglichkeiten und Methoden, durch die das Testen von Software und von Systemen mit Software verbessert werden kann. Diese Methoden haben das Ziel, den Prozess (und damit die Arbeitsergebnisse) zu verbessern, indem sie Richtlinien und Einsatzbereiche für die Verbesserungen zur Verfügung stellen.

Das Testen macht oft einen wesentlichen Teil der Gesamtkosten eines Projekts aus, trotzdem findet der Testprozess in den diversen Modellen zur Softwareprozessverbesserung, wie CMM (der Vorgänger von CMMI) und.CMMI (siehe unten), nur begrenzte Beachtung.

Testprozessverbesserungs-Modelle, wie Test Maturity Model (TMM), Systematic Test and Evaluation Process (STEP), Critical Testing Processes (CTP) und Test Process Improvement (TPI) wurden entwickelt, um diese Lücke zu schließen. Die Modelle TPI und TMM liefern ein gewisses Maß an organisationsübergreifenden Metriken für Benchmark-Vergleiche. Es gibt in der Industrie heute viele Verbesserungsmodelle. Neben den Modellen in diesem Kapitel sind Test Organization Maturity (TOM), Test Improvement Model (TIM) und Software Quality Rank (SQR) beachtenswert. Darüber hinaus gibt es noch eine große Anzahl regionaler Modelle, die eingesetzt werden. Testexperten sollten sich über alle verfügbaren Modelle informieren, um festzustellen, welches in ihrer spezifischen Situation am besten passt.

Die Modelle in diesem Kapitel sind nicht als Empfehlungen zu verstehen, sie sollen vielmehr eine repräsentative Darstellung von Funktionsweise und Inhalt von Modellen sein.

### *8.3.1 Einführung in die Prozessverbesserung*

Prozessverbesserungen sind sowohl für den Softwareentwicklungs- als auch für den Testprozess relevant. Eine Organisation lernt aus den eigenen Fehlern und kann dann die Prozesse verbessern, die sie für die Entwicklung und das Testen von Software einsetzt. Der PDCA-Zyklus nach Deming (Plan/Do/Check/Act = Planen/Ausführen/Prüfen/Handeln) ist seit vielen Jahrzehnten im Einsatz und ist auch heute noch relevant, wenn Tester den verwendeten Prozess verbessern müssen.

Prozessverbesserung geht davon aus, dass die Qualität eines Systems stark von der Qualität des Prozesses abhängt, der zur Entwicklung der Software verwendet wird. Qualitätsverbesserungen in der Softwareindustrie verringern die für die Wartung der Software benötigten Ressourcen und stellen dadurch einen größeren zeitlichen Freiraum für das Ausarbeiten von mehr und besseren Lösungen für die Zukunft zur Verfügung.

Prozessmodelle bieten einen Ansatz für Verbesserungen, indem sie die Prozessreife in der Organisation mit Hilfe des Modells messen. Das Modell dient außerdem als Rahmenwerk für die Verbesserung der Prozesse in einer Organisation anhand der Bewertungsergebnisse.

Eine Prozessbewertung führt zu einer Bewertung der Prozessreife, die wiederum eine Prozessverbesserung anregt. Später lässt sich in einer weiteren Prozessbewertung die Wirkung der Verbesserungen messen.

### 8.3.2 Arten der Prozessverbesserung

Es gibt zwei verschiedene Modelle der Prozessverbesserung: Prozessreferenzmodelle und Inhaltsreferenzmodelle.

1. Das Prozessreferenzmodell dient als Rahmenwerk für die Bewertung, wenn die Fähigkeiten eines Unternehmens mit dem Modell verglichen werden, und dazu, anhand des Rahmenwerks die Organisation zu bewerten.
2. Das Inhaltsreferenzmodell dient der Verbesserung des Prozesses, nachdem die Bewertung durchgeführt wurde.

Einige Modelle verbinden beide Teile, während andere nur aus einem Teil bestehen.

## 8.4 Testprozess verbessern

Seit einiger Zeit nutzt die IT-Branche Modelle zur Testprozessverbesserung bei ihren Bemühungen um mehr Reife und Professionalität. Dabei dienen Standardmodelle dazu, organisationsübergreifende Metriken und Maße zu entwickeln, die einen Vergleich ermöglichen. Stufenmodelle, wie TMMi liefern Standards für Vergleiche zwischen verschiedenen Unternehmen und Organisationen. Kontinuierliche Modelle erlauben es den Organisationen, die Themen mit der höchsten Priorität freier in der Reihenfolge der Implementierung zu handhaben. Da es in der Testbranche einen klaren Bedarf für Prozessverbesserungen gibt, haben sich mehrere Standards herausgebildet, beispielsweise STEP, TMMi, TPI und CTP, die alle in diesem Abschnitt behandelt werden.

Mit jedem dieser vier Modelle für die Testprozessbewertung kann eine Organisation herausfinden, wo sie gegenwärtig mit ihrem derzeitigen Testprozess steht. Nach der Bewertung liefern TMMi und TPI eine verbindlichen Weg für die Verbesserung des Testprozesses. Die Modelle STEP und CTP dagegen helfen einem Unternehmen herauszufinden, wo sich Investitionen in die Testprozessverbesserung am besten bezahlt machen, und überlassen ihm die Wahl des geeigneten Verbesserungswegs.

Wenn Einigkeit herrscht, dass die Testprozesse geprüft und verbessert werden sollen, sind folgende Prozessschritte durchzuführen:

- **I**nitiate (initiieren)
- **M**easure (messen)
- **P**rioritize and Plan (priorisieren und planen)
- Define and **R**edefine (definieren und neu definieren)
- **O**perate (betreiben)
- **V**alidate (validieren)
- **E**volve (weiterentwickeln)

(Die Anfangsbuchstaben der englischen Begriffe ergeben das Wort IMPROVE, verbessern)

Initiieren

In diesem Schritt werden Zielsetzungen, Umfang und Überdeckungsgrad der vorgesehenen Prozessverbesserungen vereinbart und die Zustimmung der Betroffenen eingeholt. Außerdem wird das Prozessmodell zur Identifizierung der Verbesserungen ausgewählt. Zur Wahl stehen entweder die veröffentlichten Modelle oder ein individuell definiertes Modell.

Bevor die Aktivitäten zur Prozessverbesserung beginnen, sollten Erfolgskriterien definiert sowie die Methode implementiert werden, die während der Prozessverbesserungsaktivitäten zur Messung dieser Erfolgskriterien anzuwenden ist.

Messen

Der vereinbarte Bewertungsansatz wird durchgeführt und liefert als Ergebnis eine Liste möglicher Prozessverbesserungen.

Priorisieren und planen

Die Liste der möglichen Prozessverbesserungen wird nach Priorität geordnet. Die Priorität kann sich orientieren an: Rentabilität, Risiken, Angleichung an die Gesamtstrategie der Organisation, messbarem quantitativen oder qualitativen Nutzen.

Nach der Priorisierung sollte ein Plan für die Durchführung der Verbesserungen entwickelt und auf den Weg gebracht werden.

Definieren und neu definieren

Anhand der erkannten Anforderungen an die Prozessverbesserung werden benötigte Prozesse neu definiert oder bestehende Prozesse nach Bedarf umdefiniert und einsatzbereit gemacht.

Betreiben

Nachdem ihre Entwicklung abgeschlossen ist, werden die Prozessverbesserungen umgesetzt. Dazu können Schulungen oder Coachings anfallen, es kann die Pilotierung von Prozessen einschließen und führt schließlich zur vollständigen Implementierung der Verbesserungen.

Validieren

Wenn die Prozessverbesserungen im Einsatz sind, ist unbedingt zu prüfen, ob sich die vorab vereinbarten Vorteile verwirklicht haben (beispielsweise Nutzenrealisierung), und ob alle Erfolgskriterien für die Prozessverbesserungsmaßnahmen erfüllt wurden.

Weiterentwickeln

Je nach verwendetem Prozessmodell ist dies der Schritt im Verbesserungsprozess, mit dem die nächste Reifestufe anvisiert wird, und die Entscheidung fällt,ob der Verbesserungszyklus von vorne beginnen oder vorerst an diesem Punkt enden soll.

Bewertungsmodelle sind eine übliche Methode, sie bieten einen standardisierten Ansatz für die Verbesserung der Testprozesse nach erprobten und bewährten Verfahren. Eine Testprozessverbesserung ist aber auch ohne Modell möglich, beispielsweise durch analytische Ansätze und Bewertungssitzungen.

## 8.5 Testprozess mit TMM verbessern

Das Test Maturity Model besteht aus fünf Stufen und soll CMM ergänzen. Jede der fünf Stufen enthält definierte Prozessbereiche, die vollständig erfüllt sein müssen, bevor die Organisation mit der nächsten Stufe fortfahren kann (Stufendarstellung). TMM bietet sowohl ein Prozessreferenzmodell als auch ein Inhaltsreferenzmodell.

Die TMM-Stufen sind

Stufe 1: Initiierung

Auf dieser ersten Stufe gibt es keinen formal dokumentierten oder strukturierten Testprozess. Tests werden normalerweise nach dem Programmieren ad hoc entwickelt und das Testen als Debugging verstanden. Ziel des Testens ist nach allgemeiner Auffassung der Nachweis, dass die Software funktioniert.

Stufe 2: Definition

Die zweite Stufe wird erreicht durch das Formulieren von Testrichtlinien und Testzielen, Einführen eines Testplanungs-Prozesses und Implementieren grundlegender Testverfahren und -methoden.

Stufe 3: Integration

Die dritte Stufe wird erreicht, wenn ein Testprozess in den Softwarelebenszyklus integriert und in formalen Standards, Verfahren und Methoden dokumentiert wird. Es sollte eigene Rollen für Softwaretester geben, und das Testen sollte gesteuert und überwacht werden.

Stufe 4: Management und Messung

Die vierte Stufe wird erreicht, wenn sich der Testprozess effektiv messen und steuern lässt, und wenn er für spezifische Projekte angepasst werden kann.

Stufe 5: Optimierung

In der letzten Stufe ist eine Testprozessreife erreicht, bei der sich Daten aus dem Testprozess nutzen lassen, um Fehlerzuständen vorzubeugen, und die weitere Optimierung des etablierten Prozesses in den Fokus rückt.

Um eine bestimmte Stufe zu erreichen, müssen eine Reihe definierter Reifeziele und untergeordneter Teilziele erfüllt sein. Die Ziele werden definiert als Aktivitäten, Aufgaben und Verantwortlichkeiten, und sie werden bewertet aus den festgelegten Perspektiven von Manager, Entwickler/Tester und Kunde/Anwender. Weitere Informationen enthält [Burnstein03].

Die TMMi Foundation [www.tmmifoundation.org] hat das Nachfolgemodell von TMM definiert: TMMi. TMMi ist ein detailliertes Modell für die Testprozessverbesserung. Es basiert auf dem TMM-Rahmenwerk, wie es vom Illinois Institute of Technology entwickelt wurde, und auf praktischen Erfahrungen bei dessen Anwendung. Es soll das CMMI ergänzen.

TMMi basiert in seiner Struktur weitgehend auf der des CMMI, beispielsweise mit Prozessbereichen, generischen Zielen, generischen Praktiken, spezifischen Zielen und spezifischen Praktiken.

## 8.6 Testprozess mit TPI verbessern

TPI (Test Process Improvement) basiert auf einer kontinuierlichen Darstellung, statt auf einer stufenweisen wie TMM.

Der bei [Koomen99] beschriebene Plan zur Testprozessoptimierung umfasst eine Anzahl von Kernbereichen, die auf die vier Eckpfeiler Lebenszyklus, Organisation, Infrastruktur/Werkzeuge und Verfahren verteilt sind. Die Kernbereiche lassen sich auf den Ebenen A bis D bewerten, A ist die unterste Ebene. Sind Kernbereiche noch sehr unreif, erreichen sie die Einstiegsebene A möglicherweise nicht. Es gibt Kernbereiche, die nur A, welche die nur A und B (beispielsweise Schätzung und Planung), welche, die nur A.B und C, sowie welche, die A,B,C und D erreichen können (beispielsweise Metriken).

Welche Ebene ein Kernbereich erreicht hat, lässt sich durch die Bewertung von Kontrollpunkten bestimmen, die im TPI-Modell definiert sind. Sind beispielsweise alle Kontrollpunkte für den Kernbereich Berichterstattung in den Ebenen A und B erfüllt, dann hat dieser Kernbereich die Ebene B erreicht.

Das TPI-Modell definiert Abhängigkeiten zwischen den verschiedenen Kernbereichen und Ebenen. Diese Abhängigkeiten dienen dazu, dass sich der Testprozess gleichmäßig entwickelt. So ist es beispielsweise nicht möglich, im Kernbereich Metriken die Ebene A zu erreichen, wenn der Kernbereich Berichterstattung sie noch nicht erreicht hat (es nützt beispielsweise wenig Metriken zu erheben, wenn nicht darüber berichtet wird). Der Einsatz von Abhängigkeiten im TPI-Modell ist nicht zwingend.

TPI ist in erster Linie ein Prozessreferenzmodell.

Es stellt eine Testreifematrix zur Verfügung, die die Ebenen (A, B, C oder D) für jeden der Kernbereiche in 13 Entwicklungsstufen des gesamten Testprozesses abbildet. Diese Stufen sind in drei Kategorien eingeteilt:

- beherrschbar
- effizient
- optimierend

Bei einem TPI-Assessment werden quantitative Metriken und qualitative Befragungen verwendet, um die Ebenen der Testprozessreife zu bewerten.

## 8.7 Testprozess mit CTP verbessern

Wie in [Black02] beschrieben, ist der Grundsatz beim CTP-Bewertungsmodell (Critical Testing Process), dass bestimmte Testprozesse als kritisch gelten. Wenn diese kritischen Prozesse gut ausgeführt werden, dann sind sie eine Unterstützung für erfolgreiche Testteams. Das gilt auch umgekehrt: Wenn diese Aktivitäten schlecht ausgeführt werden, bleiben selbst begabte Tester und Testmanager wahrscheinlich erfolglos. Das CTP-Modell kennt zwölf kritische Testprozesse.

CTP ist in erster Linie ein Inhaltsreferenzmodell.

Das Modell der kritischen Testprozesse ist ein kontextabhängiger Ansatz, der sich anpassen lässt, beispielsweise durch:

- Identifizieren von spezifischen Herausforderungen
- Erkennen von Merkmalen guter Prozesse
- Auswahl der Reihenfolge und der Wichtigkeit für die Implementierung von Prozessverbesserungen

Das CTP-Modell lässt sich für alle Vorgehensmodelle der Softwareentwicklung anpassen.

Prozessverbesserungen mit dem CTP-Modell beginnen mit einer Bewertung des bestehenden Testprozesses. Dabei wird identifiziert, welche Prozesse stark und welche schwach sind. Die Bewertung liefert priorisierte Empfehlungen für Verbesserungen, die auf die Bedürfnisse der Organisation zugeschnitten sind. Abhängig von ihrem Kontext beim Einsatz variieren die Bewertungen, normalerweise werden bei einem CTP-Assessment üblicherweise folgende quantitativen Metriken untersucht:

- Fehlerfindungsrate
- Rentabilität der Investition ins Testen
- Anforderungs- und Risikoüberdeckung
- Indirekte Release-bezogene Kosten
- Anteil der zurückgewiesenen Fehlerberichte

Bei einem CTP-Assessment werden normalerweise folgende qualitativen Faktoren bewertet:

- Rolle und Effektivität des Testteams
- Nützlichkeit des Testkonzepts
- Kompetenz des Testteams bzgl. Testen, Branchenkenntnissen, Technologie
- Nützlichkeit der Fehlerberichte
- Nützlichkeit der Ergebnisberichte
- Nützlichkeit und Ausgewogenheit des Änderungsmanagements

Sind im Assessment die Schwachstellen identifiziert, werden Pläne für die Verbesserung umgesetzt. Das Modell liefert allgemeine Verbesserungspläne für jeden der kritischen Testprozesse; es wird jedoch vom Assessment-Team erwartet, dass es sie deutlich anpasst.

## 8.8 Testprozess mit STEP verbessern

STEP (Systematic Test and Evaluation Process) setzt wie CTP (und anders als TMMi und TPI) nicht voraus, dass Verbesserungen in einer bestimmten Reihenfolge erfolgen.

Grundvoraussetzungen der Methode sind u.a.:

- Eine anforderungsbasierte Teststrategie;
- das Testen beginnt am Anfang des Softwarelebenszyklus;
- Tests werden als Anforderungs- und Nutzungsmodelle eingesetzt;
- das Design der Testmittel führt zum Softwaredesign;
- Fehlerzustände werden früher gefunden oder ganz vermieden;
- Fehlerzustände werden systematisch analysiert;
- Tester und Entwickler arbeiten zusammen.

STEP ist in erster Linie ein Inhaltsreferenzmodell.

Die STEP-Methode folgt der Vorstellung, dass das Testen eine Aktivität über den gesamten Softwarelebenszyklus ist, dass es mit der Formulierung der Anforderungen beginnt und fortgeführt wird, bis das System außer Betrieb genommen wird. Die STEP-Methode verfährt nach dem Grundsatz „erst testen, dann programmieren“. Ihre anforderungsbasierte Teststrategie soll erreichen, dass früh erstellte Testfälle die Anforderungsspezifikation noch vor Systemdesign und -programmierung validieren. Die Methode setzt auf die Verbesserung der folgenden drei Hauptphasen des Testens:

- Planung
- Durchführung
- Messung

Bei einem STEP-Assessment werden quantitative Metriken und qualitative Befragungen verwendet. Mögliche quantitative Metriken sind

- Entwicklung des Teststatus im Zeitablauf
- Anforderungs- oder Risikoüberdeckung
- Fehlertendenzen mit Aufdeckung, Schwere und Anhäufung
- Fehlerdichte
- Effektivität der Fehlerbehebung
- Fehlerfindungsrate
- Phasen der Fehlereinführung, -findung und –behebung
- Testkosten in Zeit, Aufwand und Geld

Mögliche qualitative Faktoren sind

- Nutzung des definierten Testprozesses
- Kundenzufriedenheit

Das STEP-Assessment-Modell wird manchmal mit dem TPI-Reifemodell kombiniert.

## 8.9 Capability Maturity Model Integration, CMMI

CMMI kann bzgl. zweier unterschiedlicher Darstellungen implementiert werden: gestuft oder kontinuierlich. Bei der gestuften Darstellung gibt es fünf Reifegrade oder -ebenen, wobei jede Ebene auf den Prozessbereichen aufbaut, die in den vorangegangenen Ebenen erfüllt wurden. Bei der kontinuierlichen Darstellung kann die Organisation die Prozessbereiche nach dem eigenen dringenden Bedarf verbessern, sie muss sich nicht an den Vorgängerebenen orientieren.

Die gestufte Darstellung ist im CMMI vor allem deshalb vorhanden, damit eine gemeinsame Basis mit dem CMM-Modell besteht. Die kontinuierliche Darstellung wird allgemein als die flexiblere Methode betrachtet.

Beim CMMI beziehen sich die Prozessbereiche Validierung und Verifizierung sowohl auf statische als auch auf dynamische Testprozesse.

# 9. Testwerkzeuge und Automatisierung

*Begriffe*

Debuggingwerkzeug, dynamisches Analysewerkzeug, Emulator, Fehlereinpflanzungswerkzeug, Hyperlink-Werkzeug, Lasttestwerkzeug, schlüsselwortgetriebener Test, Simulator, statischer Analysator, Testausführungswerkzeug, Testmanagementwerkzeug, Testorakel

## 9.1 Einführung

Dieser Abschnitt vertieft den Foundation-Level-Lehrplan zunächst mit einigen allgemeinen Konzepten. Anschließend werden einige Werkzeuge ausführlich erklärt.

Die Konzepte können unterschiedlich relevant entweder für Testmanager, Test Analysts oder Technical Test Analysts sein. Professionelle Tester brauchen aber Grundkenntnisse von allen. Die Grundkenntnisse lassen sich dann bei Bedarf erweitern.

Werkzeuge lassen sich unterschiedlich gruppieren, unter anderem nach ihren tatsächlichen Nutzerinnen und Nutzern, beispielsweise Testmanager, Test Analysts oder Technical Test Analysts. Diese Einteilung entspricht den Modulen dieses Lehrplans, sie wird deshalb auch in diesem Kapitel verwendet. Generell sind die Werkzeuge in den folgenden Abschnitten für ein bestimmtes der drei Module relevant; es gibt aber auch bestimmte Werkzeuge (beispielsweise Testmanagementwerkzeuge) mit breiterer Bedeutung. In diesen Fällen nennen die Ausbildungsanbieter Beispiele für den Einsatz des Werkzeugs im spezifischen Kontext.

## 9.2 Testwerkzeugkonzepte

Mit Testwerkzeugen lassen sich Effizienz und Effektivität des Testens erheblich steigern, allerdings nur dann, wenn die geeigneten Werkzeuge fachgerecht eingesetzt und implementiert werden. Testwerkzeuge müssen als ein zusätzlicher Aspekt einer gut geführten Testorganisation gemanagt werden. Testautomatisierung wird oft mit Testdurchführung gleichgesetzt, es gibt aber für die meisten manuellen Tätigkeiten unterschiedliche Arten der Testautomatisierung, deshalb lassen sich die meisten Aufgabenbereiche des Testens bis zu einem gewissen Grad automatisieren, falls die richtigen Werkzeuge vorhanden sind.

Jede Version eines Testwerkzeugs, alle Testskripte, Testsitzungen und jede Testbasis sollten dem Konfigurationsmanagement unterliegen und mit der Softwareversion verbunden sein, für die sie eingesetzt wurden. Jedes Testwerkzeug ist wichtiger Bestandteil der Testmittel und sollte dementsprechend gemanagt werden:

- die Architektur vor dem Testwerkzeug entwerfen
- korrektes Konfigurationsmanagement von Skripten und Werkzeugversionen, Patches usw. sicherstellen, einschließlich Versionsinformation
- Bibliotheken erstellen und pflegen (für die Wiederverwendung ähnlicher Konzepte bei Testfällen), Implementierung des Testwerkzeugs dokumentieren (beispielsweise wann das Werkzeug im Prozess in der Organisation eingesetzt und wie es benutzt wird)
- vorausschauend planen durch Strukturierung der Testfälle für eine zukünftige Weiterentwicklung, indem sie beispielsweise erweiterbar und wartbar gestaltet werden

### 9.2.1 Kosten, Nutzen und Risiken von Testwerkzeugen und Automatisierung

Eine Kosten-Nutzen-Analyse sollte grundsätzlich immer durchgeführt werden, um die Rentabilität zu belegen. Als wichtigste Elemente einer Kosten-Nutzen-Analyse sollte sie folgende Kostenfaktoren erfassen und dabei jeweils die Kosten für die aktuelle manuelle Vorgehensweise (ohne Einsatz von Werkzeugen) vergleichen mit denen für werkzeugbasiertes Testen (Stunden umgerechnet in Kosten, direkte Kosten, wiederkehrende Kosten, einmalige Kosten):

- Einführungskosten für
  - Wissensaneignung (Lernkurve für das Werkzeug)
  - Evaluierung (Werkzeuge vergleichen), falls zutreffend
  - Integration mit anderen Werkzeugen
  - Kauf, Anpassung oder Eigenentwicklung des Werkzeugs
- Wiederkehrende Kosten für
  - Werkzeugbesitz (Wartung, Lizenzgebühren, Supportgebühren, Kenntnisse aufrechterhalten)
  - Portabilität
  - Verfügbarkeit und Abhängigkeiten (wenn Testwerkzeuge nicht vorhanden sind)
  - kontinuierliche Kostenbewertung
  - Qualitätsverbesserung, um die optimale Nutzung der gewählten Werkzeuge sicherzustellen

Eine Wirtschaftlichkeitsrechnung (Business Case) nur auf Grundlage von Automatisierungs-Pilotprojekten übersieht oft wichtige Kosten, beispielsweise Kosten für Wartung, Aktualisierung und Erweiterung der Testskripte bei Systemänderungen. Die Lebensdauer eines Testfalls ist die Dauer, für die er ohne Überarbeitung gültig bleibt. Die erste Version eines automatisierten Testskripts zu implementieren, dauert oft wesentlich länger als eine manuelle Ausführung; über einen längeren Zeitraum gesehen, macht die automatisierte Variante es aber möglich, zahlreiche ähnliche Testskripte viel schneller zu erstellen und so die Anzahl der guten Testfälle zu erhöhen. Beim zukünftigen Einsatz der Automatisierung nach der Implementierungsphase lassen sich außerdem Testüberdeckung und Testeffizienz wesentlich verbessern. Die Wirtschaftlichkeitsrechnung für die Einführung von Testwerkzeugen muss daher auf einer langfristigen Perspektive basieren.

Jeder spezifische Testfall muss betrachtet werden, um festzustellen, ob sich eine Automatisierung lohnt. Viele Automatisierungsprojekte basieren auf der Implementierung naheliegender manueller Testfälle, ohne den tatsächlichen Nutzen jedes einzelnen Testfalls zu prüfen. Wahrscheinlich kann jeder gegebene Satz von Testfällen oder jede Testsuite manuelle, halbautomatisierte und automatisierte Tests enthalten.

Zusätzlich zu den im Foundation-Level-Lehrplan behandelten Themen, sollten folgende Aspekte berücksichtigt werden:

Zusätzlicher Nutzen:

- Die Zeit für die automatisierte Testdurchführung lässt sich leichter schätzen.
- Regressionstests und Fehlervalidierung in späteren Projektphasen sind schneller und sicherer, wenn die Testfälle automatisiert sind.
- Der Einsatz von Automationswerkzeugen kann den Status und die technische Kompetenz des Testers oder des Testteams erhöhen.
- Automatisierung lässt sich bei der parallelen, iterativen und inkrementellen Entwicklung einsetzen, um besseres Regressionstesten bei jedem Build zu ermöglichen.
- Das Werkzeug deckt bestimmte Testarten ab, die sich manuell nicht abdecken lassen (beispielsweise Performanz und Zuverlässigkeit).

Zusätzliche Risiken:

- Unvollständiges oder inkorrektes Testen wird so automatisiert, wie es ist.
- Es ist schwierig, die Testmittel zu warten; wenn die zu testende Software geändert wird, müssen mehrere Änderungen nachgezogen werden.
- Dadurch, dass die Tester bei der Testdurchführung nicht mehr direkt involviert sind, kann die Fehlerfindungsrate sinken, weil ausschließlich automatisierte Tests mit Testskript durchgeführt werden.

### 9.2.2 Testwerkzeugstrategien

Testwerkzeuge sollen normalerweise für mehr als ein Projekt eingesetzt werden. Je nach Höhe der Investition und Projektdauer kann es sein, dass sich der Einsatz für nur ein Projekt zunächst nicht rentiert, sondern erst bei nachfolgenden Versionen der Software. So ist beispielsweise die Wartungsphase oft sehr testintensiv, bei jeder Änderung ist eine große Regressions-Testsuite auszuführen. Deshalb kann es kostengünstig sein, ein System in der Wartungsphase zu automatisieren, wenn sich das aufgrund einer langen Systemlebensdauer lohnt. Ein weiteres Beispiel: Beim manuellen Testen können den Testern beispielsweise leicht Tippfehler unterlaufen, sodass die Kosten-Nutzen-Rechnung positiv ausfällt, wenn in einer Testsuite die Dateneingaben und der Vergleich der Ergebnisdaten mit den Daten des Testorakels automatisiert werden.

Für Unternehmen, die viele Testwerkzeuge benutzen (für unterschiedliche Teststufen und -zwecke) und von diesen abhängig sind, empfiehlt sich eine langfristige Testwerkzeugstrategie, die als Entscheidungshilfe dient, wenn es um die Einführung oder das Auslaufen von bestimmten Werkzeugversionen und Werkzeug-Support geht. Für größere Unternehmen mit intensiver Werkzeugnutzung kann es ratsam sein, eine allgemeine Richtlinie für die Werkzeugbeschaffung, Strategien, Werkzeug-Paradigmen oder zu verwendende Skriptsprachen zu erstellen.

### 9.2.3 Integration und Informationsaustausch zwischen Werkzeugen

In einem Test- (und Entwicklungs-)prozess wird in der Regel mehr als ein Testwerkzeug eingesetzt. So kann ein Unternehmen beispielsweise gleichzeitig ein statisches Analysewerkzeug einsetzen, ein Testmanagement- und -berichtswerkzeug, ein Konfigurationsmanagementwerkzeug, ein Abweichungsmanagement- und ein Testausführungswerkzeug. Dann sollte das Unternehmen prüfen, ob die Werkzeuge sich integrieren lassen, und ob ein nützlicher Informationsaustausch möglich ist. Es wäre beispielsweise günstig, wenn alle Statusinformationen aus einer Testdurchführung direkt an das Testmanagement-System berichtet würden, und sich damit der Testfortschritt sofort aktualisieren und Anforderungen sich direkt zu einem bestimmten Testfall zurückverfolgen ließen. Es ist nicht nur aufwändiger, sondern auch eine mögliche Fehlerquelle, wenn die Testskripte sowohl in einer Testmanagementdatenbank als auch in einem Konfigurationsmanagement-System gespeichert werden. Will ein Tester mitten in einem Testfall einen Abweichungsbericht herausgeben, dann müssen Fehlerverfolgungs- und Testmanagement-Systeme integriert sein. Statische Analysewerkzeuge können zwar getrennt von anderen Werkzeugen arbeiten, es wäre aber viel effizienter, wenn diese Werkzeuge Abweichungen, Warnungen und andere Rückmeldungen direkt an das Testmanagement-System berichten würden.

Die Beschaffung von Testwerkzeugen vom selben Anbieter bedeutet nicht zwangsläufig, dass die Werkzeuge in der beschriebenen Weise miteinander arbeiten, selbst wenn dies eine realistische Erwartung wäre. All diese Aspekte sollten berücksichtigt werden: Von den Kosten für eine Automatisierung des Informationsaustauschs bis zu den Risiken durch Datenverfälschung oder -verlust bei rein manuellen Abläufen (wenn die Organisation überhaupt Zeit hat für diese Arbeiten).

Neue Konzepte, wie integrierte Entwicklungsumgebungen (beispielsweise Eclipse), zielen darauf ab, Integration und Einsatz verschiedener Werkzeuge in derselben Umgebung zu erleichtern, indem sie eine gemeinsame Schnittstelle für Entwicklungs- und Testwerkzeuge bieten. So können Werkzeuganbieter ein Plug-in zum Eclipse-Framework bereitstellen und damit erreichen, dass ihr Werkzeug Eclipse-konform wird. Das Werkzeug sieht dann so aus und lässt sich benutzen wie jedes andere Werkzeug im Eclipse-Rahmenwerk, was für die Nutzer vorteilhaft ist. Hinweis: Auch wenn die Benutzerschnittstellen ähnlich sind, bedeutet das nicht, dass die Werkzeuge automatisch integriert sind und untereinander Informationen austauschen können.

### *9.2.4 Automatisierungssprachen: Skripte, Skriptsprachen*

Skripte und Skriptsprachen können für die bessere Implementierung und Ausweitung von Testbedingungen und Testfällen eingesetzt werden. So kann beispielsweise beim Testen einer Web-Anwendung ein Skript verwendet werden, um die Benutzerschnittstelle zu umgehen und die Schnittstelle des Anwendungsprogramms selbst (API, Application Programming Interface) angemessener zu testen. Ein anderes Beispiel ist die Automatisierung des Tests einer Benutzerschnittstelle, um alle möglichen Kombinationen von Eingaben zu testen. Mit einem manuellen Vorgehen wäre das nicht umsetzbar.

Skriptsprachen haben sehr unterschiedliche Fähigkeiten. Hinweis: Skriptsprachen können normale Programmiersprachen bis zu sehr spezifischen standardisierten Notationen sein, beispielsweise TTCN-3.

### *9.2.5 Konzept der Testorakel*

Testorakel werden verwendet, um erwartete Ergebnisse zu bestimmen. Als solche erfüllen sie die gleiche Funktion wie die zu testende Software und sind daher selten verfügbar. Sie können jedoch dann eingesetzt werden, wenn ein altes System durch ein neues mit identischer Funktionalität ersetzt werden soll; das alte System kann dann als Testorakel dienen. Testorakel können auch dann verwendet werden, wenn die Performanz des zu liefernden Systems wichtig ist. Ein Orakel mit niedriger Leistung kann gebaut oder verwendet werden, um die erwarteten Ergebnisse der funktionalen Tests der leistungsstärkeren Software zu erzeugen.

### *9.2.6 Testwerkzeuge in Betrieb nehmen*

Jedes automatisierte Werkzeug ist eine eigene Software, die Hardware- oder Softwareabhängigkeiten haben kann. Auch das Werkzeug sollte dokumentiert und getestet sein, ob es fertig gekauft, adaptiert oder in der Organisation erstellt wurde. Einige Werkzeuge lassen sich gut in die Umgebung integrieren, andere funktionieren besser im Einzelbetrieb.

Wenn das zu testende System auf proprietärer Hardware, Betriebssystemen, eingebetteter Software läuft, oder wenn es nicht standardmäßige Konfigurationen nutzt, kann es notwendig sein, ein Werkzeug zu erstellen (zu entwickeln) oder ein Werkzeug anzupassen, damit es in die spezifische Umgebung passt. Es ist immer eine Kosten-Nutzen-Analyse empfehlenswert, die die anfänglichen Kosten und die langfristigen Wartung berücksichtigt.

Bei Inbetriebnahme eines Testautomatisierungswerkzeugs ist es nicht ratsam, die manuellen Testfälle eins zu eins zu übernehmen, sondern sie vielmehr für die Automatisierung neu zu definieren. Dazu müssen die Testfälle formatiert werden, wiederverwendbare Muster berücksichtigt, die Eingaben durch Variablen statt fest programmierter Werte erweitert und die Vorteile des Testwerkzeugs genutzt werden (beispielsweise seine Fähigkeit zum Verbinden von Testfällen, Wiederholen, Ändern der Reihenfolge, bessere Analyse- und Berichtsfunktionen). Viele Testautomatisierungswerkzeuge setzen Programmierkenntnisse voraus, um effiziente und effektive Testprogramme (Skripte) und Testsuiten

zu erstellen. Große Testsuiten sind oft schwierig zu aktualisieren und zu managen, wenn sie nicht mit Sorgfalt entworfen wurden. Deshalb sind geeignete Schulungen für Anwendung der Testwerkzeuge, Programmierung und Entwurfsverfahren sehr wertvoll, damit sich die Vorteile der Werkzeuge vollständig nutzen lassen.

Auch wenn manuelle Testfälle automatisiert wurden, ist es wichtig, sie regelmäßig auch manuell auszuführen, damit das Wissen nicht verloren geht, wie der Test funktioniert, und um die korrekte Arbeitsweise zu verifizieren.

Wenn ein Werkzeug im Einsatz ist und die Anzahl der Testskripte stetig zunimmt, kann es nötig werden, bestimmte Funktionen hinzuzufügen, die andere Werkzeuge bieten. Das ist nicht immer möglich, da nicht alle Werkzeuge eine offene Schnittstelle bieten und manchmal eigene, nicht standardisierte Skriptsprachen verwenden. Es ist daher ratsam, Werkzeuge einzusetzen, die sich über Plug-ins in offene Rahmenwerke einfügen lassen oder über API-Schnittstellen verfügen. Das garantiert eine bessere Zukunftsfähigkeit der Testskripte als Testmittel.

Für jede Art von Werkzeug sollten die nachfolgend aufgelisteten Eigenschaften betrachtet werden, unabhängig von der Testphase, in der das Werkzeug eingesetzt werden soll. Diese Eigenschaften sind sowohl bei der Evaluierung von Werkzeugen als auch bei der Eigenentwicklung nützlich. Ein Werkzeug kann in jedem dieser Aspekte schwach oder stark sein. Eine solche Liste eignet sich deshalb auch zum Vergleich der Fähigkeiten unterschiedlicher Werkzeuge.

- Analyse: Konzept, Eingaben und manuell oder automatisch eingebrachte Informationen
- Entwurf: manuelle oder automatische Erstellung
- Auswahl: manuelle oder automatische Auswahl auf Basis von Kriterien, beispielsweise Überdeckung
- Ausführung: manuelle oder automatische Ausführung, Steuerung, Wiederanlauf etc.
- Bewertung (beispielsweise Testorakel) und Präsentation: Sie werden oft als manuelle oder automatische Protokollier- und Berichtsfunktionen bezeichnet, die beispielsweise Vergleiche mit einer Vorlage, einem Standard oder mit einem bestimmten Kriterium durchführen.

### *9.2.7 „Open Source“-Testwerkzeuge verwenden*

Werkzeuge für den Test von sicherheitskritischen Systemen müssen je nach Verwendungszweck und geltenden Standards zertifiziert sein. „Open Source“-Werkzeuge sollten bei sicherheitskritischen Systemen nur eingesetzt werden, wenn eine entsprechende Zertifizierung des Werkzeugs vorliegt.

Die Qualität von „Open Source“-Software hängt von ihrer Verbreitung, Vorgeschichte und jeweiligen Verwendung ab. Es lässt sich nicht davon ausgehen, dass die Qualität von „Open Source“-Software besser oder schlechter als kommerzielle Werkzeuge ist.

Für jedes Testwerkzeug sollte immer eine Qualitätsbewertung durchgeführt werden, um dessen Eignung zu bewerten. Bei einigen Werkzeugarten kann es zu irreführenden Ergebnissen in der Bewertung kommen, wenn ein positives Bewertungsergebnis durch eine inkorrekte Werkzeuganwendung erzielt wurde (wenn beispielsweise ein Schritt nicht ausgeführt wurde, aber nicht berichtet wurde welcher). Lizenzgebühren sollten genau betrachtet werden. Es könnte (von der Open Source Community) erwartet werden, dass man Verbesserungen bzw. Erweiterungen, die man macht, allgemein zur Verfügung stellt.

### *9.2.8 Eigene Testwerkzeuge entwickeln*

Viele Testwerkzeuge werden entwickelt, weil Tester oder Entwickler ihre Arbeit beschleunigen wollen oder müssen. Andere Gründe für die Eigenentwicklung von Testwerkzeugen können sein, dass es keine geeigneten Werkzeuge auf dem Markt gibt, oder dass proprietäre Hardware oder

Testumgebungen verwendet werden müssen. Diese Werkzeuge sind oft sehr effizient beim Ausführen der vorgesehenen Aufgaben; sie hängen aber oft auch sehr von der Person ab, die sie entwickelt hat. Deshalb sollten sie so dokumentiert werden, dass auch andere Personen sie pflegen können. Bevor sie in einer Organisation weiter verbreitet werden, sollten unbedingt ihre Zwecke, Ziele, Vor- und Nachteile geprüft werden. Es kommt oft vor, dass solche Werkzeuge für neue Anforderungen verwendet werden, oder dass sie weit über ihren ursprünglichen Gebrauch hinaus erweitert werden, was nicht immer von Vorteil ist.

### *9.2.9 Testwerkzeuge klassifizieren*

Testwerkzeuge lassen sich danach klassifizieren, welche Aktivitäten sie unterstützen (Foundation-Level-Lehrplan). Weitere Möglichkeiten der Klassifizierung sind

- Einteilung der Werkzeuge nach Teststufe (Komponenten-, Integrations-, System-, Abnahmetest)
- Einteilung der Werkzeuge nach Fehlerzuständen, die mit ihnen bearbeitet und unterstützt werden
- Einteilung der Werkzeuge nach Testvorgehensweise oder Testverfahren (siehe unten)
- Einteilung der Werkzeuge nach Zweck des Testens, beispielsweise Messung, Treiber, Protokollierung, Vergleiche
- Einteilung der Werkzeuge nach bestimmten Fachbereichen, beispielsweise Verkehrssimulation und -leitung, Netzwerke, Protokolle, Transaktionen, TV-Bildschirme, Expertensysteme
- Einteilung der Werkzeuge nach Aufgabenbereichen, die sie unterstützen, beispielsweise Dateneingabe, Umgebung, Konfiguration oder andere konzeptionelle Bereiche
- Einteilung der Werkzeuge nach der vorgesehenen Anwendung: als Standardprodukt, Rahmenwerk (zur Anpassung), Plug-in (beispielsweise Eclipse), Standard-Testsuite oder Testsuite für Zertifizierungen, firmeninterne Werkzeugentwicklung

Außerdem lassen sich Werkzeuge nach ihren tatsächlichen Nutzern klassifizieren, wie Testmanager, Test Analysts und Technical Test Analysts. Diese Einteilung entspricht den Modulen dieses Lehrplans, sie wird deshalb auch in den folgenden Abschnitten verwendet. Der Foundation-Level-Lehrplan enthält bereits ein Kapitel über Testwerkzeuge, die nachfolgenden Abschnitte ergänzen Aspekte der Testwerkzeuge.

## 9.3 Testwerkzeugkategorien

Dieser Abschnitt hat folgende Ziele:

- Er liefert zusätzliche Informationen für die bereits in Kapitel 6 des ISTQB® Foundation-Level-Lehrplans beschriebenen Werkzeuge (beispielsweise Testmanagementwerkzeuge, Testausführungs- und Lasttestwerkzeuge).
- Er führt neue Werkzeugkategorien ein.

Allgemeine Informationen über Werkzeugkategorien, die in diesem Abschnitt nicht behandelt werden, enthält ISTQB® Foundation-Level-Lehrplan, Kapitel 6.

### *9.3.1 Testmanagementwerkzeuge*

Allgemeine Informationen über Testmanagementwerkzeuge enthält ISTQB® Foundation-Level-Lehrplan, Abschnitt 6.1.2.

Die folgenden Informationen sollten Testmanagementwerkzeuge liefern können:

- Rückverfolgbarkeit von Testartefakten
- Erfassung der Testumgebungsdaten in komplexen Umgebungen
- Daten zur gleichzeitigen Ausführung von Testsuiten in unterschiedlichen Testumgebungen während derselben Testsitzung an mehreren Einsatzorten (in verschiedenen Organisationen)
- Metriken und Indikatoren, wie
  - Anzahl der Testbedingungen
  - Anzahl der Testfälle
  - Dauer der Ausführung (beispielsweise des Testfalls, einer Testsuite, einer Regressionstestsuite) und andere wichtige Zeiten, einschließlich Durchschnittswerten, die für Managemententscheidungen wichtig sein könnten
  - Anzahl der Testfälle, Testartefakte und Testumgebungen
  - Anteil der bestandenen/nicht bestandenen Tests
  - Anzahl der noch offenen Testfälle (und der Gründe, weshalb sie nicht ausgeführt wurden)
  - Tendenzen
  - Anzahl der Anforderungen
  - Anzahl der Beziehungen zwischen und Verfolgbarkeit von Testartefakten
- Konzepte, die von den Testmanagementwerkzeugen unterstützt werden, wie
  - Organisation von Testartefakten, Repositories und Treibern von Testfällen
  - Testbedingungen und Testumgebungen
  - Regressionstestsuiten, Testsitzung
  - Protokollierung und Information zur Fehlerbearbeitung
  - Wiederanlauf der Umgebung (und Neuinitialisierung)
  - Testmetriken über die Testartefakte (Testdokumentation), um den Testfortschritt zu dokumentieren

Testmanagementwerkzeuge werden von Testmanagern, Test Analysts und Technical Test Analysts benutzt.

### *9.3.2 Testausführungswerkzeuge*

Allgemeine Informationen über Testausführungswerkzeuge enthält ISTQB® Foundation-Level-Lehrplan 2007, Abschnitt 6.1.5.

Testausführungswerkzeuge werden überwiegend von Test Analysts und Technical Test Analysts auf allen Teststufen eingesetzt, um Tests auszuführen und die Ergebnisse zu kontrollieren. Testausführungswerkzeuge verfolgen normalerweise eins oder mehrere der folgenden Ziele:

- Kosten reduzieren (Aufwand oder Zeit)
- mehr Tests ausführen
- Tests leichter wiederholbar machen

Testausführungswerkzeuge werden meist zur Automatisierung der Regressionstests eingesetzt.

Testausführungswerkzeuge führen eine Reihe von Anweisungen in einer Programmiersprache aus, die oft auch Skriptsprache genannt wird. Die Anweisungen für das Werkzeug sind sehr ausführlich und spezifizieren Details, wie einzelne Schaltflächenaktivierungen, Tastendrucke und Mausbewegungen. Diese detaillierten Skripte werden dadurch sehr anfällig für Änderungen an der Software unter Test (SUT), vor allem wenn sie die graphische Benutzerschnittstelle (GUI) betreffen.

Ausgangspunkt für ein Skript kann ein Mitschnitt (Capture/Replay) sein, oder ein erstelltes oder bearbeitetes Skript auf der Grundlage vorhandener Skripte, Vorlagen oder Schlüsselwörter. Ein Skript ist ein Programm und funktioniert wie jede Software. Beim Erfassen (oder Aufzeichnen) lässt sich ein Audit Trail für das nicht-systematische Testen aufzeichnen. Die meisten Testausführungswerkzeuge haben einen Testkomparator, der die tatsächlichen mit den gespeicherten erwarteten Testergebnissen

vergleicht. Beim Testen (wie beim Programmieren) geht der Trend weg von detaillierten Anweisungen und hin zu höheren Sprachen, auch hier werden Bibliotheken, Makros und Unterprogramme benutzt. Folgen von Anweisungen in einem Skript können mit einem Namen (Schlüsselwort oder Aktionswort) gekennzeichnet werden; daher werden diese schlüsselwortgetriebene oder aktionswortgetriebene Tests genannt. Hauptvorteil ist die Trennung der Skriptanweisungen von den Daten. Wie bei der Nutzung von Vorlagen für die Skripterstellung soll dieses Konzept den Aufwand minimieren.

Hauptgrund für das Versagen von Testwerkzeugen in der Praxis sind geringe Programmierfähigkeiten und ein mangelndes Verständnis dafür, dass ein Testwerkzeug durch die automatisierte Testdurchführung nur einen Teil der Probleme löst. Jede Automatisierung der Testdurchführung bedeutet Management, Aufwand, besondere Qualifikationen und Sorgfalt, beispielsweise für Testarchitektur und Konfigurationsmanagement. Das heißt auch, dass Testskripte Fehler enthalten können. Eine Testmittel-Architektur kann eine gewisse Unabhängigkeit von einem bestimmten Werkzeuglieferanten bieten. Wenn ein Werkzeug beschafft wird, meinen viele, dass seine Vorgaben befolgt werden müssen, beispielsweise bei Aufbau und Namenskonventionen von Skripten. Mit der Automatisierung des Testaufbaus lassen sich aber die eigene optimale Weise, die Tests zu organisieren, mit den Ablagestrukturen verbinden, die das Werkzeug fordert, um die Tests ausführen zu können.

### 9.3.3 Debugging und Fehleranalysewerkzeuge

Die Fehleranalyse kann mit Werkzeugen den Bereich eingrenzen, im dem ein Fehlerzustand auftritt. Das kann auch nötig sein, wenn in einem System nicht offensichtlich ist, welcher Fehlerzustand die Fehlerwirkung ausgelöst hat. Fehleranalysewerkzeuge enthalten eine Ablaufverfolgung und simulierte Umgebungen für die Interaktion mit der Software, oder sie ziehen Informationen aus dem System, um so den Ort der Fehlerwirkung einzugrenzen.

Mit Debugging- und Verfolgungswerkzeugen reproduzieren Programmierer Fehler und untersuchen den Zustand von Programmen. Sie können damit:

- Programme Zeile für Zeile ausführen
- Programme an einer beliebigen Programmanweisung anhalten
- Programmvariable setzen und untersuchen

Hinweis: Debugging und die Werkzeuge dafür haben zwar Gemeinsamkeiten mit dem Testen, sind aber nicht damit gleichzusetzen. Debuggingwerkzeuge sind keine Testwerkzeuge. Tester können Debugging- und Verfolgungswerkzeuge zur Fehleranalyse einsetzen, um den Ursprung einer Fehlerwirkung zu orten, und um zu bestimmen, wohin der Abweichungsbericht zu senden ist. Debugging-, Verfolgungs- und Fehleranalysewerkzeuge werden hauptsächlich von Technical Test Analysts benutzt.

### 9.3.4 Fehlereinpflanzungs- und Fehlerinjektionswerkzeuge

Fehlereinpflanzung und Fehlerinjektion sind zwei unterschiedliche Verfahren, die beim Testen eingesetzt werden können. Die Fehlereinpflanzung nutzt ein Werkzeug, das einem Compiler ähnelt, und generiert damit systematisch einzelne oder bestimmte Arten von Codefehlern. Diese Werkzeuge werden oft beim Mutationstestverfahren eingesetzt und werden manchmal auch Mutationstest-Werkzeug genannt.

Fehlerinjektion soll die vorhandenen Schnittstellen zu Testkomponenten ändern, für die kein Quellcode verfügbar ist. Sie kann aber auch gezielt einen bestimmten Fehlerzustand einbringen, um einerseits zu prüfen, ob die Software damit umgehen kann (Fehlertoleranz), oder um festzustellen, ob ein Test in der Testsuite den absichtlich eingeschleusten Fehler findet. Fehlereinpflanzungs- und Fehlerinjektionswerkzeuge werden vor allem von Technical Test Analysts auf Codeebene verwendet,

Test Analysts können damit aber auch Daten in einer Datenbank manipulieren oder Fehlerzustände in den Datenstrom einschleusen, um das Systemverhalten zu testen.

## 9.3.5 Simulations- und Emulationswerkzeuge

Simulatoren unterstützen das Testen dann, wenn Code oder andere Systeme nicht verfügbar, teuer oder ungeeignet sind (beispielsweise beim Testen von Software, die mit einer nuklearen Kernschmelze umgehen muss). Einige Simulatoren und Testrahmen können auch das Fehlerverhalten unterstützen oder abbilden, sodass sich Fehlerzustände oder Fehlerszenarien prüfen lassen. Das größte Risiko beim Einsatz dieser Werkzeuge ist, dass sie Fehler in Zusammenhang mit den Ressourcen (beispielsweise Timing-Probleme) möglicherweise nicht finden, die für manche Arten von Systemen sehr wichtig sind.

Emulatoren gehören zur Kategorie der Simulatoren, sie sind Software, die zum Abbilden einer Hardware geschrieben wird. Ihr Vorteil ist, dass sie ausgefeiltere Tests ermöglichen, und vor allem, dass Ablaufverfolgung, Debugging und zeitabhängige Ursachen nachgestellt werden können, was in einem echten System vielleicht nicht möglich wäre. Emulatoren sind teuer zu erstellen, der Vorteil einer Analyse, bei der das System quasi in Zeitlupe läuft, ist aber für manche parallelen, zeitabhängigen und komplexen Systeme unbezahlbar.

Diese Werkzeuge werden, je nach der benötigten Art der Emulation, von Test Analysts und Technical Test Analysts eingesetzt.

## 9.3.6 Statische und dynamische Analysewerkzeuge

Allgemeine Informationen über statische und dynamische Analysewerkzeuge enthält ISTQB® Foundation-Level-Lehrplan 2007, Abschnitt 6.1.6 Werkzeugunterstützung für Performanzmessung und Testmonitore.

### 9.3.6.1 Statische Analysewerkzeuge

Statische Analysewerkzeuge lassen sich in allen Phasen des Softwarelebenszyklus einsetzen und auch in allen Phasen oder Stufen der Softwareentwicklung, je nachdem, welche Messungen sie liefern.

Statische Analysewerkzeuge berichten gefundene Fehler als Warnungen. Unbegründete Warnungen bezeichnet man als falsch positiv. Wahr positive Meldungen sind tatsächliche Fehlerzustände, die in der Ausführung zu Fehlerwirkungen oder Ausfällen führen können. Es kann schwierig und zeitaufwändig sein, zwischen falschen und wahren Positiven zu unterscheiden, weil dazu eine fachgerechte Fehleranalyse nötig ist. Neuere statische Analysewerkzeuge können Informationen aus dem dynamischen Binden während des Kompilierens nutzen, und sie sind daher wirkungsvoller beim Finden echter Fehler (weniger falsch Positive). Diese Werkzeuge werden von Technical Test Analysts benutzt.

### 9.3.6.2 Dynamische Analysewerkzeuge

Dynamische Analysewerkzeuge liefern Laufzeitdaten über den Status der ausführenden Software. Diese Werkzeuge werden überwiegend eingesetzt, um nicht zugeordnete Zeiger zu identifizieren, die Zeigerarithmetik zu prüfen, um Zuweisung, Nutzung und Freigabe von Speicherplatz zu überwachen und Engpässe anzuzeigen, und zum Aufdecken anderer, mit statischen Methoden schwer auffindbarer Fehler. Speicherwerkzeuge sollten in großen und komplexen Systemen wiederholt auf mehreren Ebenen eingesetzt werden, weil Speicherprobleme dynamisch entstehen. Hinweis: unterschiedliche kommerzielle Testwerkzeuge können unterschiedlich implementiert sein und deshalb auch unterschiedliche Speicher- oder Ressourcenprobleme (Stacks, Heaps) suchen und berichten. Das bedeutet, dass zwei verschiedene Speicherwerkzeuge auch unterschiedliche Probleme

identifizieren könnten. Speicherwerkzeuge sind besonders nützlich für das Testen bestimmter Programmiersprachen (beispielsweise C, C++), bei denen die Programmierer das Speichermanagement übernehmen. Diese Werkzeuge werden von Technical Test Analysts benutzt.

### 9.3.7 Schlüsselwortgetriebene Testautomatisierung

Schlüsselwörter (manchmal Aktionswörter genannt) werden meist (aber nicht ausschließlich) dazu verwendet, übergeordnete Geschäftsinteraktionen mit einem System zu benennen (beispielsweise: Auftrag stornieren). Jedes Schlüsselwort bezeichnet dabei normalerweise eine Reihe detaillierter Interaktionen mit dem zu testenden System. Die Testfälle werden als Sequenzen von Schlüsselwörtern (mit den relevanten Testdaten) spezifiziert [Buwalda01].

Beim automatisierten Testen werden die einzelnen Schlüsselwörter als ein oder mehrere auszuführende Testskripte implementiert. Die Werkzeuge lesen die mit Schlüsselwörtern erstellten Testfälle und rufen die entsprechenden Testskripte auf. Die Testskripte sind sehr modular implementiert, damit sie sich leicht auf bestimmte Schlüsselwörter abbilden lassen. Für die Implementierung der modularen Skripte sind Programmierkenntnisse nötig.

Die wichtigsten Vorteile der schlüsselwortgetriebenen Testautomatisierung sind:

- Experten für einen bestimmten Anwendungs- oder Geschäftsbereich können die Schlüsselwörter definieren. Das macht die Spezifikation der Testfälle effizienter.
- Für eine Person, die vorwiegend Fachexpertise hat, ist die automatische Testfalldurchführung vorteilhaft (nachdem die Schlüsselwörter in Form von Testskripten implementiert wurden).
- Es ist einfacher, Testfälle zu warten, die unter Verwendung der Schlüsselwörter geschrieben wurden, weil sie mit geringerer Wahrscheinlichkeit modifiziert werden müssen, wenn sich Details der zu testenden Software ändern.
- Testfallspezifikationen sind unabhängig von der Implementierung. Die Schlüsselwörter können mit verschiedenen Skriptsprachen und Werkzeugen implementiert werden.

Die schlüsselwortgetriebene Testautomatisierung wird überwiegend von Fachexperten und Test Analysts benutzt.

### 9.3.8 Performanztestwerkzeuge

Allgemeine Informationen über Performanztestwerkzeuge enthält ISTQB® Foundation-Level-Lehrplan 2007, Abschnitt 6.1.6 Werkzeugunterstützung für Performanzmessung und Testmonitore.

Performanztestwerkzeuge leisten zweierlei:

- Lastgenerierung
- Messung und Analyse des Systemverhaltens bei der generierten Last

Zur Lastgenerierung wird ein definiertes Nutzungsprofil (siehe Abschnitt 5.3.3) als Skript implementiert. Das Skript kann zunächst für einen einzigen Nutzer erfasst werden (beispielsweise mit einem Mitschnittwerkzeug), und wird dann durch das Lasttestwerkzeug für das festgelegte Nutzungsprofil implementiert. Die Implementierung muss die Datenschwankungen pro Transaktion (oder Menge von Transaktionen) berücksichtigen.

Performanztestwerkzeuge generieren eine Last, indem sie eine große Nutzerzahl (virtuelle Nutzer) mit bestimmten Mengen von Eingabedaten simulieren. Im Unterschied zu den Mitschnittwerkzeugen, die die Nutzungsinteraktion über eine graphische Benutzerschnittstelle simulieren, erzeugen viele Performanztestskripte die Nutzungsinteraktionen mit dem System auf der Ebene des Kommunikationsprotokolls. Nur wenige Lastgenerationswerkzeuge können die Last generieren, indem sie die Anwendung über die Benutzerschnittstelle steuern.

Performanztestwerkzeuge liefern eine Vielzahl von Messungen für die Analyse während oder nach Durchführung des Tests. Typische Metriken und Berichte dieser Testwerkzeuge sind

- Anzahl simulierter Nutzer
- Anzahl und Art der von den simulierten Nutzern erzeugten Transaktionen
- Antwortzeiten für bestimmte, von den Nutzern angeforderte Transaktionen
- Berichte auf Basis von Testprotokollen oder -logs und Graphen, die Last und Antwortzeiten. gegenüberstellen
- Systemmonitordaten (wie z.B. CPU Auslastung)

Wichtige Faktoren beim Implementieren von Performanztestwerkzeugen sind

- Hardware und die Netzwerkbandbreiten, die für die Lastgenerierung benötigt werden
- Kompatibilität des Werkzeugs mit dem Kommunikationsprotokoll des zu testenden Systems
- ausreichende Flexibilität des Werkzeugs für die einfache Implementierung unterschiedlicher Nutzungsprofile
- benötigte Überwachungs-, Analyse- und Berichtsfunktionen

Performanztestwerkzeuge werden in der Regel gekauft, weil deren Entwicklung sehr aufwändig ist. Es kann aber sinnvoll sein, ein bestimmtes Performanztestwerkzeug zu entwickeln, wenn technische Einschränkungen den Einsatz eines Produkts nicht zulassen, oder wenn Nutzungsprofil und benötigte Funktionen relativ einfach sind. Performanztestwerkzeuge werden in der Regel von Technical Test Analysts benutzt.

Hinweis: Performanzbezogene Fehlerzustände haben oft tief greifende Auswirkungen auf die zu testende Software. Wenn die Anforderungen an die Performanz des Systems sehr wichtig sind, ist es meist sinnvoll, die Performanz der kritischen Komponenten zu testen (mittels Treibern und Platzhaltern), und nicht bis zu den Systemtests zu warten.

### *9.3.9 Hyperlink-Testwerkzeuge*

Mit Hyperlink-Werkzeugen werden Webseiten darauf analysiert und kontrolliert, ob keine Hyperlinks ungültig sind oder fehlen. Einige Werkzeuge liefern außerdem zusätzliche Informationen, beispielsweise einen Graphen mit der Architektur oder Baumstruktur der Seite, Geschwindigkeit und Größe des Downloads (je URL), Trefferzahl und Volumen. Mit Hyperlink-Testwerkzeugen lässt sich auch die Konformität mit Service Level-Vereinbarungen (Service Level Agreement, SLA) überwachen. Hyperlink-Testwerkzeuge werden von Test Analysts und Technical Test Analysts benutzt.

# 10. Soziale Kompetenz und Teamzusammensetzung

*Begriffe*

Unabhängigkeit des Testens

## 10.1 Einführung

Professionelle Tester sollten wissen, welche individuellen Fähigkeiten sie zur fachgemäßen Erfüllung ihrer spezifischen Aufgaben brauchen. Dieses Kapitel beleuchtet zunächst die individuellen Fähigkeiten, bevor es sich mit Themen befasst, die für Testmanager besonders wichtig sind, beispielsweise Gruppendynamik, Organisation, Motivation und Kommunikation.

## 10.2 Individuelle Fähigkeiten

Eine Person kann entweder durch ihre Erfahrung oder durch Schulung in verschiedenen Arbeitsbereichen zum Testen von Software befähigt sein. Folgende Aspekte können zu ihrer Wissensbasis beitragen:

- Nutzung von Softwaresystemen
- Kenntnis des Geschäfts- oder Fachbereichs
- Tätigkeiten in den verschiedenen Phasen der Softwareentwicklung, einschließlich Analyse, Entwicklung und technischen Support
- Tätigkeiten im Bereich Softwaretesten

Anwender von Softwaresystemen sind vertraut mit der Nutzerseite der Systeme und haben gute Kenntnisse darüber, wie die Systeme angewendet werden, in welchen Bereichen Fehlerwirkungen die schwerwiegendsten Auswirkungen haben würden, und welches Systemverhalten erwartet werden kann. Fachexperten wissen, welche Bereiche für das Geschäft von größter Wichtigkeit sind, und kennen den Einfluss dieser Bereiche auf die Fähigkeit des Unternehmens, die geschäftlichen Anforderungen zu erfüllen. Dieses Wissen lässt sich nutzen für die Priorisierung der Testaktivitäten, den Entwurf realistischer Testdaten bzw. Testfällen und die Verifizierung oder Beschaffung von Anwendungsfällen.

Die Kenntnis des Softwareentwicklungs-Prozesses (Anforderungsanalyse, Entwurf, Programmierung) erklärt, wie sich Fehler einschleichen, wo sie zu finden sind, und wie Fehlern vorgebeugt werden kann. Erfahrung im technischen Support liefert Wissen über die Praxis der Nutzer und ihre Erwartungen und Anforderungen an die Benutzbarkeit. Erfahrung im Bereich Softwareentwicklung ist wichtig für den Einsatz von anspruchsvollen Testautomatisierungswerkzeugen, die Programmier- und Entwurfsspezialisten erfordern.

Zu den spezifischen Fertigkeiten beim Softwaretesten gehören die Fähigkeit zur Analyse von Spezifikationen, Teilnahme an Risikoanalysen, Entwurf von Testfällen sowie Fleiß und Sorgfalt beim Durchführen der Tests und der Aufzeichnung der Ergebnisse.

Für Testmanager sind Wissen, Fähigkeiten und Erfahrung im Projektmanagement besonders wichtig, weil das Management eines Tests der Leitung eines Projekts entspricht - mit Tätigkeiten, wie Erstellen des Plans, Überwachung und Kontrolle des Fortschritts sowie Berichten an die Betroffenen.

Zwischenmenschliche Fähigkeiten, wie der Umgang mit Kritik, Einflussnahme und Verhandlungsgeschick sind alle wichtig für die Rolle des Testers. Technisch kompetente Tester können ihrer Rolle als Tester kaum gerecht werden, wenn sie die nötigen zwischenmenschlichen

Fähigkeiten nicht haben und einsetzen. Erfolgreiche Tester müssen nicht nur mit anderen effektiv zusammenarbeiten, sondern auch ihre Arbeit gut organisieren können, ein Auge fürs Detail haben und ausgeprägte kommunikative Fähigkeiten besitzen, sowohl schriftlich als auch mündlich.

## 10.3 Dynamik im Testteam

Eine der wichtigsten Funktionen des Managers in einer Organisation ist die Auswahl der Mitarbeiter. Neben einer individuellen Befähigung für die Aufgabe sind viele Aspekte zu berücksichtigen. Wenn eine Person für das Team ausgewählt wird, ist auch die Dynamik im Team zu beachten. Wird diese Person die im Testteam vorhandenen Fähigkeiten und verschiedenen Persönlichkeiten sinnvoll ergänzen? Es kann Vorteile haben, wenn in einem Testteam sowohl unterschiedliche Persönlichkeiten als auch unterschiedliche technische Fähigkeiten zusammenkommen. Ein starkes Testteam kann mit mehreren Projekten unterschiedlicher Komplexität umgehen und zugleich die zwischenmenschlichen Beziehungen zu den anderen Mitgliedern im Projektteam erfolgreich gestalten.

Neue Teammitglieder müssen schnell vom Testteam integriert werden und eine angemessene Betreuung erhalten. Jede Person im Team sollte ihre definierte Rolle haben, die sich aus einer individuellen Beurteilung ergeben kann. Ziel ist es, dass jedes Mitglied als einzelne Person erfolgreich ist und gleichzeitig zum Erfolg des gesamten Teams beiträgt. Das lässt sich zum Einen durch passende Teamrollen für die jeweiligen Persönlichkeitstypen erreichen, und zum Anderen, indem man sowohl auf die vorhandenen Fähigkeiten des Einzelnen baut als auch sein oder ihr Portfolio an Fähigkeiten erweitert.

Hinweis: Es gibt kaum je die perfekt geeignete Person, aber man kann ein starkes Team auch bilden, indem man die Stärken und Schwächen der einzelnen Teammitglieder ausgewogen miteinander kombiniert. Durch Wissenstransfer im Team lässt sich das Teamwissen pflegen und aufbauen, was letztendlich die Flexibilität erhöht.

## 10.4 Testen in der Organisationsstruktur etablieren

Unternehmen ordnen das Testen sehr unterschiedlich in ihre Organisationsstruktur ein. Die Qualität gehört zwar während des gesamten Softwarelebenszyklus in die gemeinsame Verantwortung aller, ein unabhängiges Testteam kann aber entscheidend zur einem hochwertigen Produkt beitragen. In der Praxis ist das Testen in unterschiedlichem Maße unabhängig, wie die folgenden Auflistung zeigt, die von der geringsten zur höchsten Unabhängigkeit reicht:

- Keine unabhängigen Tester
  - Es gibt keine Unabhängigkeit, und der Entwickler testet seinen eigenen Programmcode.
  - Falls ihm Zeit für das Testen zur Verfügung steht, wird der Entwickler feststellen, ob das Programm so funktioniert, wie von ihm beabsichtigt. Ob damit die tatsächlichen Anforderungen erfüllt sind, bleibt offen.
  - Der Entwickler kann aufgedeckte Fehlerzustände schnell korrigieren.
- Ein Entwickler testet, der das Programm nicht geschrieben hat.
  - Es gibt nur wenig Unabhängigkeit zwischen Entwickler und Tester.
  - Ein Entwickler, der den Programmcode eines anderen Entwicklers testet, wird Fehlerzustände möglicherweise ungern berichten.
  - Die Mentalität eines Entwicklers beim Testen führt meist zu einem Fokus auf positive Testfälle.
- Ein Tester (oder Testteam) aus dem Entwicklungsteam testet.
  - Tester (oder Testteam) berichten an das Projektmanagement.


  - Die Mentalität eines Testers führt meist zu einem Fokus darauf, die Einhaltung von Anforderungen zu verifizieren.
  - Weil der Tester Mitglied des Entwicklungsteams ist, kann er zusätzlich Entwicklungsaufgaben haben.
- Tester kommen aus den Fachabteilungen des Unternehmens, aus dem Kreis der Nutzer oder einer anderen technischen Organisation, die nicht mit Softwareentwicklung befasst ist.
  - Es wird unabhängig an die Betroffenen berichtet.
  - Qualität ist das Hauptinteresse dieses Teams.
  - Entwicklung von Fähigkeiten und Schulung konzentrieren sich auf das Testen.
- Externe Testexperten testen für spezifische Testziele.
  - Testziele könnten beispielsweise Benutzbarkeit, Sicherheit oder Performanz sein.
  - Qualität sollte das Hauptinteresse dieser Experten sein; das hängt aber vom Berichtsweg ab.
- Eine externe Organisation testet.
  - Es gibt maximale Unabhängigkeit.
  - Der Wissenstransfer kann unzureichend sein.
  - Es sind eindeutige Anforderungen und eine gut definierte Kommunikationsstruktur notwendig.
  - Die Qualität der externen Organisation muss regelmäßig in einem Audit bewertet werden

Der Grad der Unabhängigkeit zwischen Entwicklungs- und Testorganisationen variiert stark. Hinweis: Ein Mehr an Unabhängigkeit kann mit mehr Isolation und weniger Wissenstransfer einhergehen. Ein geringeres Maß an Unabhängigkeit kann das Wissen über das System verbessern, es kann aber zu Interessenskonflikten durch widersprüchliche Zielsetzungen führen. Der Grad der Unabhängigkeit wird auch vom Softwareentwicklungs-Modell bestimmt: bei einer agilen Entwicklungsmethode sind Tester beispielsweise meist Teil des Entwicklungsteams.

Die oben erwähnten Optionen können in einem Unternehmen gemischt vorkommen. Es kann in der Entwicklungsabteilung getestet werden und zusätzlich durch eine unabhängige Testabteilung; abschließend kann eine externe Organisation zertifizieren. Es ist wichtig, Zuständigkeiten und Erwartungen für die einzelnen Teststufen zu verstehen, und die Anforderungen an sie so zu stellen, dass sich die bestmögliche Qualität des Endprodukts im Rahmen der zeitlichen und finanziellen Grenzen erzielen lässt.

Outsourcing ist eine Möglichkeit, eine externe Organisation mit dem Testen zu betrauen. Es kann sich dabei um ein anderes Unternehmen handeln, das die Testleistungen vor Ort, an einem externen Standort im eigenen Land, oder im Ausland (off-shore) erbringt. Outsourcing ist eine Herausforderung, vor allem, wenn die externe Organisation im Ausland angesiedelt ist. Folgende Aspekte sollten berücksichtigt werden:

- kulturelle Unterschiede
- Überwachung der Outsourcing-Ressourcen
- Informationstransfer, Kommunikation
- Schutz geistigen Eigentums
- Bestehende Fähigkeiten, Entwicklung weitere Fähigkeiten und Schulungen
- Fluktuation der Mitarbeiter
- genaue Kostenschätzungen
- Qualität

## 10.5 Motivieren

Es gibt viele Möglichkeiten, die einzelnen Testmitarbeiter zu motivieren, darunter

- Anerkennung für die bewältigten Aufgaben
- Einverständnis des Managements
- Respekt im Projektteam und in der Gruppe
- angemessene materielle Anerkennung der geleisteten Arbeit (einschließlich Gehalt, Leistungszulagen und Bonuszahlungen)

Bestimmte Projekteinflüsse können die Anwendung dieser Anreize erschweren. So arbeitet ein Tester möglicherweise sehr hart an einem Projekt mit einem unmöglichen Endtermin. Er oder sie kann dann alles Menschenmögliche unternehmen (Überstunden, Mehrarbeit), um die Qualitätsziele im Team voranzutreiben, aber das Produkt wird aufgrund externer Einflüsse vorzeitig ausgeliefert und kann trotz aller Bemühungen des Testers eine schlechte Qualität haben. Solche Vorkommnisse können demotivieren, vor allem, wenn der Einsatz des Testers nicht verstanden und angerechnet wird, unabhängig davon, ob das Endprodukt erfolgreich ist.

Das Testteam muss geeignete Metriken festhalten, um nachzuweisen, dass bei Durchführung des Testens gute Arbeit geleistet, Risiken minimiert und Ergebnisse richtig protokolliert wurden.Werden diese Informationen nicht erfasst und mitgeteilt, kann ein Team leicht demotiviert werden, weil ihm die Anerkennung für seine gute Arbeit fehlt.

Anerkennung zeigt sich nicht nur in abstrakten Konzepten, wie Respekt und Zustimmung, sie wird deutlich wahrnehmbar in konkreten Beförderungschancen, Gehaltsstufen und Laufbahnen. Wird die Testgruppe nicht respektiert, dann können solche Chancen fehlen.

Anerkennung und Respekt gewinnen die Tester, wenn es offensichtlich wird, dass sie den Mehrwert eines Projekts steigern. Bei einem einzelnen Projekt lässt sich das am schnellsten erreichen, indem die Tester bereits an der Konzeption des Projekts beteiligt werden und es über den gesamten Softwarelebenszyklus bleiben. Mit der Zeit werden die Tester Anerkennung und Respekt für ihren Beitrag zur positiven Entwicklung des Projekts erhalten; ihr Beitrag sollte aber auch als geringere Kosten für die Qualität und Risikobeherrschung quantifiziert werden.

## 10.6 Kommunizieren

Die Kommunikation im Testteam findet vorwiegend auf drei Ebenen statt:

- Dokumentation der Testprodukte: Teststrategie, Testkonzept, Testfälle, Testberichte, Fehlerberichte usw.
- Review-Feedback zu geprüften Dokumenten: Anforderungen, Funktionsspezifikationen, Anwendungsfälle, Komponententestdokumentation usw.
- Sammeln und Verteilen von Informationen: Interaktion mit Entwicklern, anderen Testteammitgliedern, Management usw.

Die Kommunikation muss immer professionell, objektiv und effektiv sein, damit das Testteam von anderen respektiert wird. Wenn Tester anderen Beteiligten Rückmeldungen, vor allem konstruktive Kritik, zu deren Arbeitsergebnissen mitteilen, brauchen sie Fingerspitzengefühl und Objektivität.

Zweck der Kommunikation sollte immer sein, die Testziele zu erreichen und sowohl die Qualität des Produkts zu verbessern als auch die der Prozesse für die Produktion der Softwaresysteme. Tester kommunizieren mit einem breiten Personenkreis, wie Nutzern, Projektteammitgliedern, Management, externen Testgruppen und Kunden. Die Kommunikation muss für die jeweiligen Adressaten effektiv sein. So kann beispielweise ein Bericht für die Entwickler, der die wöchentliche Menge der gefundenen Fehlern mit einem bestimmten Schweregrad aufzeigt, für eine Managementpräsentation auf Vorstandsebene zu detailliert und damit ungeeignet sein.

# 11. Referenzen

## 11.1 Standards

Dieser Abschnitt nennt die in diesem Lehrplan erwähnten Standards.

### *11.1.1 Nach Kapiteln*

Folgende Kapitel verweisen auf Standards:

Kapitel 2
BS 7925-2, IEEE 829, DO-178B/ED-12B

Kapitel 3
IEEE 829, D0-178B/ED-12B

Kapitel 4
BS 7925-2

Kapitel 5
ISO 9126, BS 7925-2

Kapitel 6
IEEE 1028

Kapitel 7
IEEE 829, IEEE 1044, IEEE 1044.1.

Kapitel 8
IEEE 829, IEEE 1028, IEEE 1044, IEEE 1044.1, BS 7925-2

### *11.1.2 Alphabetisch*

Folgende alphabetisch aufgelisteten Standards werden in den angegebenen Kapiteln erwähnt:

[BS 7925-2] BS 7925-2 (1998) Software Component Testing

Kapitel 2 und 4

[IEEE 829] IEEE Std 829™ (1998) IEEE Standard for Software Test Documentation

Kapitel 2, 3, 7 und 8

[IEEE 1028] IEEE Std 1028™ (1997) IEEE Standard for Software Reviews

Kapitel 6 und 8

[IEEE 1044] IEEE Std 1044™ (1993) IEEE Standard Classification for Software Anomalies

Kapitel 7 und 8

[ISO 9126] ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality

Kapitel 5

[ISTQB®] ISTQB® Glossary of terms used in Software Testing, Version 2.0, 2007

Für die entsprechende deutschsprachige Fassung des Glossars, siehe bitte die Web-Site des German Testing Boards ( www.german-testing-board.info ).

[RTCA DO-178B/ED-12B]: Software Considerations in Airborne systems and Equipment certification, RTCA/EUROCAE ED12B.1992.

Kapitel 2 und 3

## 11.2 Literatur

[Beizer95] Beizer Boris, „Black-box testing", John Wiley & Sons, 1995, ISBN 0-471-12094-4

[Black02] Rex Black, „Managing the Testing Process (2nd edition)", John Wiley & Sons: New York, 2002, ISBN 0-471-22398-0

[Black03] Rex Black, „Critical Testing Processes", Addison-Wesley, 2003, ISBN 0-201-74868-1

[Black07] Rex Black, „Pragmatic Software Testing", John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Burnstein03] Ilene Burnstein, „Practical Software Testing", Springer, 2003, ISBN 0-387-95131-8

[Buwalda01] Hans Buwalda, „Integrated Test Design and Automation" Addison-Wesley Longman, 2001, ISBN 0-201-73725-6

[Copeland03] Lee Copeland, „A Practitioner's Guide to Software Test Design", Artech House, 2003, ISBN 1-58053-791-X

[Craig02] Craig, Rick David; Jaskiel, Stefan P., „Systematic Software Testing", Artech House, 2002, ISBN 1-580-53508-9

[Gerrard02] Paul Gerrard, Neil Thompson, „Risk-based e-business testing", Artech House, 2002, ISBN 1-580-53314-0

[Gilb93] Gilb Tom, Graham Dorothy, „Software inspection", Addison-Wesley, 1993, ISBN 0-201-63181-4

[Graham07] Dorothy Graham, Erik van Veenendaal, Isabel Evans, Rex Black „Foundations of Software Testing", Thomson Learning, 2007, ISBN 978-1-84480-355-2

[Grochmann94] M. Grochmann (1994), „Test case design using Classification Trees", in: conference proceeedings STAR 1994

[Jorgensen02] Paul C.Jorgensen, „Software Testing, a Craftsman's Approach second edition", CRC press, 2002, ISBN 0-8493-0809-7

[Kaner02] Cem Kaner, James Bach, Bret Pettichord; „Lessons Learned in Software Testing"; Wiley, 2002, ISBN: 0-471-08112-4

[Koomen99] Tim Koomen, Martin Pol, „Test Process Improvement", Addison-Wesley, 1999, ISBN 0-201-59624-5

[Myers79] Glenford J.Myers, „The Art of Software Testing", John Wiley & Sons, 1979, ISBN 0-471-46912-2

[Pol02] Martin Pol, Ruud Teunissen, Erik van Veenendaal, „Software Testing: A Guide to the Tmap Approach", Addison-Wesley, 2002, ISBN 0-201-74571-2

[Splaine01] Steven Splaine, Stefan P.,Jaskiel, „The Web-Testing Handbook", STQE Publishing, 2001, ISBN 0-970-43630-0

[Stamatis95] D.H. Stamatis, „Failure Mode and Effect Analysis", ASQC Quality Press, 1995, ISBN 0-873-89300

[vanVeenendaal02] van Veenendaal Erik, „The Testing Practitioner", UTN Publsihing, 2002, ISBN 90-72194-65-9


[Whittaker03] James Whittaker, „How to Break Software", Addison-Wesley, 2003, ISBN 0-201-79619-8

[Whittaker04] James Whittaker and Herbert Thompson, „How to Break Software Security", Peason / Addison-Wesley, 2004, ISBN 0-321-19433-0

## 11.3 Sonstige Referenzen

Die folgenden Referenzen verweisen auf Informationen im Internet.

Die Referenzen wurden zum Zeitpunkt der Veröffentlichung dieses Advanced Level Lehrplans vom ISTQB® geprüft, das ISTQB® kann aber keine Verantwortung für ihre Verfügbarkeit übernehmen.

- Kapitel 5
  - www.testingstandards.co.uk
- Kapitel 8
  - TMMi www.tmmifoundation.org
- Weitere
  - www.sei.cmu.edu/cmmi/adoption/pdf/cmmi-overview06.pdf
  - Bug Taxonomy: www.testingeducation.org/a/bsct2.pdf
  - Sample Bug Taxonomy based on Boris Beizer's work: inet.uni2.dk/~vinter/bugtaxst.doc
  - Gute Übersicht über verschiedene Taxonomien: testingeducation.org/a/bugtax.pdf
  - Heuristic Risk-Based Testing von James Bach. James Bach, Interview auf "What is Testing.com". www.whatistesting.com/interviews/jbach.htm
  - www.satisfice.com/articles/et-article.pdf
  - From „Exploratory & Risk-Based Testing (2004)" www.testingeducation.org
  - Exploring Exploratory Testing, Cem Kaner and Andy Tikam, 2003
  - Pettichord, Bret, „An Exploratory Testing Workshop Report", www.testingcraft.com/exploratorypettichord

# 12. Anhang A – Hintergrundinformationen zum Lehrplan

## *Ziele für die Advanced Level Zertifizierung und Qualifizierung*

- Anerkennung des Testens als eine eigene essentielle und professionelle Disziplin der Softwareentwicklung
- Schaffung eines standardisierten Rahmens für die berufliche Laufbahn von Softwaretestern
- Verbesserung der Anerkennung von professionellen und qualifizierten Testern durch Arbeitgeber, Kunden und Arbeitskollegen; Schärfung ihres Profils
- Förderung konsistenter und guter Testpraktiken in allen Disziplinen der Softwareentwicklung
- Identifizierung von Testthematiken, die für die Industrie relevant und wertvoll sind
- Softwarelieferanten zu ermöglichen, dass sie zertifizierte Tester einstellen und mit dieser Personalpolitik einen Wettbewerbsvorteil gegenüber der Konkurrenz erzielen
- Testern und am Testen interessierten Personen zu ermöglichen, dass sie eine international anerkannte Qualifizierung erlangen

## *Zulassungsanforderungen für diese Qualifikation*

Für die Zulassung zur Prüfung für das ISTQB® Advanced Level Zertifikat „Softwaretesten" gelten folgende Kriterien:

- Basiszertifikat Foundation Level, ausgestellt von einer vom ISTQB® anerkannten Prüfungsinstitution oder einem ISTQB® Mitgliedsboard.
- angemessene Anzahl von Jahren der Erfahrung im Bereich Softwaretesten oder Softwareentwicklung, gemäß Festlegung durch die Prüfungsinstitution oder das ISTQB® Mitgliedsboard, die für die Advanced Level-Zertifizierung zuständig sind
- Anerkennung des in diesem Lehrplan enthaltenen Ethik-Kodex.

Es wird weiterhin empfohlen, dass die Prüfungskandidaten an einem Kurs teilnehmen, der von einem ISTQB® Mitgliedsboard akkreditiert ist. Für die Teilnahme an einer ISTQB® Prüfung ist die Schulung nicht zwingend vorgeschrieben.

Bestehende Zertifizierungen im Softwaretesten („Practitioner Certificate" oder „Advanced Certificate"), die von einem vom ISTQB® anerkannten Mitgliedsboard oder Prüfungsinstitution vor Einführung dieses internationalen Zertifikats ausgestellt wurden, sind als gleichwertig mit dem Internationalen Zertifikat anerkannt. Das ISTQB® Advanced Level Zertifikat „Softwaretesten" wird nicht ungültig und muss nicht erneuert werden. Das Datum der Ausstellung ist auf dem Zertifikat angegeben.

In den teilnehmenden Ländern werden lokale Aspekte vom jeweiligen vom ISTQB® anerkannten nationalen Testing-Board kontrolliert. Die Pflichten der Mitgliedsboards sind vom ISTQB® festgelegt, werden aber durch die jeweiligen nationalen Organisationen umgesetzt. Zu den Pflichten der Mitgliedsboards gehört die Akkreditierung von Ausbildungsanbietern und das Ansetzen von Prüfungen, die direkt oder indirekt durch eine oder mehrere vertraglich verpflichtete Prüfungsinstitutionen durchgeführt werden.

# 13. Anhang B – Hinweise für die Leser

## 13.1 Prüfungsinstitutionen

Dieser Advanced Level Lehrplan des Aufbaukurses setzt die Kenntnis des Inhalts des Lehrplans des Grundkurses („ISTQB® Certified Tester Foundation Level", Version 2005) voraus, mit der im erwähnten Grundkurs-Lehrplan festgelegten Wissensstufe.

Vom ISTQB® anerkannte Prüfungsinstitutionen können Prüfungsfragen aus allen Themen dieses Lehrplans erarbeiten.

Es wird empfohlen, dass die Prüfungsfragen unterschiedlich gewertet werden, je nach den Lernzielen des entsprechenden Themas. Beispiel: Einer Prüfungsfrage der kognitiven Ebene K1 werden weniger Punkte zugeordnet als einer der kognitiven Ebene K3, eine Prüfungsfrage der Ebene K4 bekäme mehr Punkte.

## 13.2 Prüfungskandidaten und Ausbildungsanbieter

Für die Zertifizierung als ISTQB® Certified Tester Advanced Level müssen die Prüfungskandidaten das Zertifikat ISTQB® Certified Tester Foundation Level vorweisen und der für sie zuständigen Prüfungsinstitution nachweisen, dass sie ausreichend praktische Erfahrung haben, um als für die Aufbaustufe (Advanced Level) qualifiziert zu gelten. Bitte wenden Sie sich an die für Sie zuständige Prüfungsinstitution, um die spezifischen Kriterien zum Nachweis der notwendigen praktischen Erfahrung zu erfahren [5]. Um ein angemessenes Können für den Advanced Level-Status im Berufsfeld Softwaretesten zu erreichen, wird von den Prüfungskandidaten mehr verlangt als nur die Inhalte dieses Lehrplans zu kennen. Prüfungskandidaten und Ausbildungsanbieter sollten mehr Zeit als in diesem Lehrplan angegeben aufbringen, um sich mit der Thematik zu beschäftigen, sei es durch Lesen oder Recherche.

Dieser Lehrplan enthält eine Liste der Referenzen, Bücher und Standards, die als zusätzliche Lektüre für Prüfungskandidaten und Ausbildungsanbieter gedacht sind, damit sie die Themen des Lehrplans detaillierter verstehen.

[5] Für mögliche Aktualisierungen siehe bitte die Web-Site des German Testing Boards (www.german-testing-board.info)

# 14. Anhang C – Hinweise für die Ausbildungsanbieter

## 14.1 Module

Dieser Lehrplan enthält die Inhalte der drei folgenden Module:

1. Advanced Level Testmanager
2. Advanced Level Test Analyst
3. Advanced Level Technical Test Analyst

Mit Bestehen der Prüfungen zu allen drei Modulen ist der Prüfungskandidat zertifiziert als „Full Advanced Level Testing Professional".

## 14.2 Ausbildungszeiten

### *14.2.1 Ausbildung je Modul*

Der Unterricht in jeder der 3 unterschiedlichen Rollen sollte die folgende Zeit dauern:

| | | |
|---|---|---|
| 1. | Advanced Level Testmanager | 5 Tage |
| 2. | Advanced Level Test Analyst | 5 Tage |
| 3. | Advanced Level Technical Test Analyst | 5 Tage |

Die Kursdauer basiert auf der Anzahl der Kapitel je Modul und den spezifischen Lernzielen des jeweiligen Kapitels. Die minimale Zeitdauer je Kapitel und Rolle ist festgelegt.

Ausbildungsanbieter können mehr als die angegebene Zeit ansetzen, und die Kandidaten können darüber hinaus zusätzliche Zeit für Studium und Recherche aufwenden. Das Kursprogramm muss die Reihenfolge der Kapitel in diesem Lehrplan nicht einhalten.

Die Kurse müssen nicht an aufeinander folgenden Tagen stattfinden. Es ist den Ausbildungsanbietern freigestellt, ihre Kurse anders zu organisieren, beispielsweise 3 + 2 Tage für Testmanager, oder 2 gemeinsame Tage gefolgt von jeweils 3 Tagen für Test Analysts und Technical Test Analysts.

### *14.2.2 Gemeinsamkeiten*

Ausbildungsanbieter können sich entscheiden, gemeinsame Themen nur einmal zu vermitteln, um so die Gesamtdauer zu verringern und Wiederholungen zu vermeiden. Die Ausbildungsanbieter werden aber darauf hingewiesen, dass dasselbe Thema manchmal aus unterschiedlichen Perspektiven zu betrachten ist, je nach Kursmodul.

### *14.2.3 Quellen*

Der Lehrplan enthält Referenzen auf gängige Standards, die für die Vorbereitung der Kursmaterialien zu verwenden sind. Jeder Standard muss dabei in der im Lehrplan referenzierten Version verwendet werden. Weitere Publikationen, Vorlagen oder Standards, die in diesem Lehrplan nicht angegeben sind, können verwendet und referenziert werden, sind aber nicht Gegenstand der Prüfung.

## 14.3 Praktische Übungen

Praktische Arbeiten (kurze Übungen) sollten bei allen Lerninhalten eingesetzt werden, bei denen die Kandidaten ihr Wissen anwenden sollen (Lernziele der Ebene K3 oder höher). Die Kursvorträge und


Übungen sollten auf den Lernzielen dieses Lehrplans sowie auf den im Lehrplan enthaltenen Beschreibungen der einzelnen Themen basieren.

# 15. Anhang D – Empfehlungen

Die folgenden Empfehlungen beziehen sich auf mehrere Kapitel dieses Lehrplans und wurden deshalb in einem Anhang zusammengefasst. Diese Empfehlungen sind prüfungsrelevant.

Die Liste enthält eine Reihe hilfreicher Empfehlungen, wie sich die Herausforderungen beim Testen bewältigen lassen, und basiert auf den im Foundation-Level-Lehrplan eingeführten sieben Grundsätzen des Softwaretestens. Die nachfolgende Liste erhebt keinen Anspruch auf Vollständigkeit, sondern ist gedacht als eine Auswahl gewonnener Erkenntnisse (lessons learned), die berücksichtigt werden sollten. Ausbildungsanbieter wählen aus der Liste die Punkte aus, die für das jeweilige Modul am ehesten relevant sind.

## 15.1 Empfehlungen für die Industrialisierung

Beim Implementieren von Tests in einer industriellen Umgebung sind Tester mit verschiedenen Herausforderungen konfrontiert. Fortgeschrittene Tester sollten die verschiedenen Empfehlungen dieses Lehrplans im Kontext ihrer eigenen Organisation, Teams, Aufgaben und Softwarekomponenten anwenden können. Die folgende Liste beschreibt Aspekte, die sich bei der Durchführung von Testaufgaben immer wieder negativ ausgewirkt haben. Die Liste erhebt keinen Anspruch auf Vollständigkeit.

- Testkonzepte werden mit dem Schwerpunkt auf funktionalen Tests erstellt.
  Fehlerzustände sind nicht auf funktionale Aspekte beschränkt, oder auf einen einzelnen Nutzer. Die Interaktion von mehreren Nutzern kann sich auf die zu prüfende Software auswirken.
- Konfiguration wird nicht ausreichend getestet.
  Wenn mehrere Arten von Prozessoren, Betriebssystemen, virtuellen Maschinen, Browsern und diversen Peripheriegeräten zu vielen unterschiedlichen Konfigurationen kombiniert werden können, können viele potenzielle Fehlerzustände unentdeckt bleiben, wenn das Testen auf einige wenige Konfigurationen begrenzt wird.
- Stress- und Lasttests werden auf den letzten Moment verschoben.
  Die Ergebnisse der Stress- und Lasttests können große Änderungen in der Software erforderlich machen (bis hin zur Grundarchitektur). Da dafür möglicherweise beträchtliche Ressourcen benötigt werden, kann es sich sehr negativ auf das Projekt auswirken, wenn die Tests erst kurz vor der geplanten Produktionseinführung erfolgen.
- Dokumentation wird nicht getestet.
  Nutzer erhalten Software und Dokumentation. Wenn die Dokumentation nicht zur Software passt, können die Nutzer das volle Potenzial der Software nicht ausschöpfen, oder werden die Software möglicherweise gar nicht verwenden.
- Installationsprozedur wird nicht getestet.
  Installationsprozeduren, wie auch Backup- und Wiederherstellungsprozeduren, werden nicht oft durchgeführt, sind aber kritischer als die Software. Wenn sich die Software nicht installieren lässt, wird sie überhaupt nicht verwendet.
- Es wird darauf bestanden, dass eine Testaufgabe vor Beginn der nächsten vollständig abgeschlossen ist.
  Obwohl manche Softwarelebenszyklusmodelle die sequenzielle Durchführung der Aufgaben empfehlen, müssen in der Praxis oft einige Aufgaben (zumindest teilweise) gleichzeitig ausgeführt werden.

- Risikobereiche werden nicht korrekt identifiziert.
  Bereiche, die als riskant eingestuft wurden, werden gründlicher getestet. Für Bereiche, die nur wenig oder gar nicht getestet wurden, kann sich das Risiko später als höher herausstellen als ursprünglich angenommen.
- Testeingaben und Testprozeduren werden zu genau spezifiziert.
  Lässt man Testern nicht genügend Freiraum für eigene Initiative bei der Definition von Testeingaben und –vorgehen, dann werden sie womöglich vielversprechende Bereiche (mit möglicherweise vorhandenen, versteckten Fehlern) nicht näher untersuchen.
- „Irrelevante" Merkwürdigkeiten (Nebeneffekte) werden nicht bemerkt oder untersucht. Scheinbar irrelevante Beobachtungen oder Ergebnisse sind oft Hinweise auf mögliche Fehler, die sich (wie Eisberge) unter der Oberfläche verstecken.
- Es wird geprüft, ob das Produkt tut, was es tun soll, aber nicht, ob es nicht tut, was es nicht tun soll.
  Wird nur getestet, was das Produkt können soll, werden möglicherweise Aspekte der Software übersehen, die sie nicht haben soll (beispielsweise zusätzliche, unerwünschte Funktionen).
- Testsuiten sind nur für ihre Eigentümer verständlich.
  Wenn Tester zu anderen Zuständigkeitsbereichen wechseln, müssen andere Tester die spezifizierten Tests lesen und verstehen. Wenn keine nachvollziehbaren und verständlichen Testspezifikationen vorhanden sind, kann sich das negativ auswirken. Testziele werden möglicherweise nicht verstanden, oder der Test wird ganz gestrichen.
- Es wird nur über die für die Nutzer sichtbare Schnittstelle getestet.
  Schnittstellen einer Software beschränken sich nicht auf die Benutzerschnittstelle. Auch Kommunikation zwischen Prozessen, Batch-Verarbeitung und andere Interrupts beeinflussen die Software und können Fehlerwirkungen auslösen.
- Abweichungsberichte und Konfigurationsmanagement sind schlecht.
  Fehler-/Abweichungsmanagement und -berichte sowie das Konfigurationsmanagement sind außerordentlich wichtig für den Erfolg des Gesamtprojekts (für Entwicklung und Test). Die Arbeit eines Testers ist dann erfolgreich, wenn gefundene Fehler korrigiert und behoben werden, nicht dann, wenn zwar viele Fehler gefunden, für eine Korrektur aber nicht gut genug beschrieben werden.
- Es kommen nur Regressionstests hinzu.
  Testsuiten über die Zeit zu entwickeln, beschränkt sich nicht darauf zu prüfen, ob Regressionsfehler vorkommen. Der Programmcode wird sich im Laufe der Zeit weiterentwickeln, und es müssen zusätzliche Tests implementiert werden, um diese neuen Funktionalitäten abzudecken und mögliche Regressionen in anderen Bereichen der Software zu prüfen.
- Es werden keine Notizen für kommende Testaufgaben gemacht.
  Die Testaufgaben sind nicht abgeschlossen, wenn die Software an die Nutzer übergeben wird oder auf den Markt kommt. Wahrscheinlich wird es eine neue Version oder Release geben, deshalb sollte das entsprechende Wissen gesichert und an die Tester weitergegeben werden, die für die nächsten Testaufgaben zuständig sind.
- Es wird versucht, Tests vollständig zu automatisieren.
  Automatisierung kann sinnvoll erscheinen, ist aber ein eigenes Entwicklungsprojekt. Es sollten nicht alle Tests automatisiert werden, manche Tests lassen sich schneller manuell durchführen als automatisieren.
- Es wird erwartet, dass alle manuellen Tests wiederholt werden.
  Wenn die manuellen Tests wiederholt werden, ist es oft unrealistisch zu erwarten, dass alle wiederholt werden. Die Aufmerksamkeit der Tester wird schwanken, und sie werden sich, ob bewusst oder unbewusst, auf bestimmte Bereiche der Software konzentrieren.

- GUI-Automatisierungswerkzeuge werden eingesetzt, um die Testerstellungskosten zu reduzieren.
  Ein GUI-Mitschnittwerkzeug (Capture/Replay-Werkzeug) ist eine erhebliche Anfangsinvestition. Das Werkzeug sollte im Rahmen einer definierten Strategie eingesetzt und alle damit verbundenen Kosten klar sein und bewertet werden.
- Es wird erwartet, dass Regressionstests einen hohen Anteil neuer Fehlerzustände aufdecken.
  Regressionstests finden im Allgemeinen keinen hohen Anteil neuer Fehlerzustände, vor allem weil die Tests bereits einmal durchgeführt wurden (beispielsweise bei einer früheren Version der Software). Die Fehlerzustände sollten bereits zu diesem Zeitpunkt aufgedeckt worden sein. Das bedeutet nicht, dass Regressionstests ganz entfallen sollten. Ihre Effektivität (die Fähigkeit neue Fehler zu finden) ist aber geringer als die anderer Testarten.
- Die Codeüberdeckung begeistert allein wegen der schönen Zahlen.
  Codeüberdeckungen und Metriken sind aus Sicht des Managements äußerst interessant, weil sie Zahlen und Graphiken liefern. Zahlen sagen aber wenig aus über Effektivität oder Sachdienlichkeit eines Tests. Beispiel: 100% Überdeckung ist zwar eine schöne Zielsetzung für Codeüberdeckung; ist dieses Ziel aber auch realistisch und geeignet (d.h., sollte es eine Anweisungs-, eine Bedingungs- oder eine modifizierte Bedingungs-/Entscheidungsüberdeckung sein)?
- Tests werden aus einer Regressions-Testsuite gestrichen, nur weil sie ein bestimmtes Überdeckungsmaß nicht erhöhen.
  In einer Regressions-Testsuite dürfen (oder sollten) einige Tests gestrichen und andere hinzugefügt werden. Die Entscheidung, bestimmte Tests zu streichen, sollte aber nicht nur davon abhängen, ob der Test ein einzelnes Überdeckungsmaß erhöht, weil der Testfall (oder die Art des Fehlers, die der Test aufdeckt) möglicherweise keinen Einfluss auf die betrachtete Testüberdeckung hat. Beispiel: Codeüberdeckung ist nicht die einzige Art von Überdeckung. Tests wurden möglicherweise aus anderen Gründen erstellt (beispielsweise für spezifische Werte oder Ereignisketten).
- Überdeckung dient als Kriterium für die Leistung der Tester.
  Überdeckungsgrad ist ein Maß für die Vollständigkeit, nicht für Leistung oder Effizienz der Mitarbeiter. Für die Bewertung der Testereffizienz im Kontext der Organisation oder des Projekts lassen sich andere Metriken definieren. Ihr Einsatz muss aber sehr sorgfältig überlegt werden, um unerwünschte, dysfunktionale Effekte zu vermeiden.
- Die Überdeckung wird überhaupt nicht mehr gemessen.
  Es gibt unterschiedliche Arten von Überdeckung (beispielsweise von Programmanweisungen, Bedingungen, Modulen, Funktionen usw.), und der Arbeitsaufwand für die Erhebung von geeigneten Metriken kann beträchtlich sein. Das ist aber kein Grund, auf Überdeckungsmetriken ganz zu verzichten, weil sie für die Testaufgabe sehr nützlich sein können.

Viele dieser Punkte werden in einzelnen Abschnitten dieses Lehrplans angesprochen.

Wenn Testmaßnahmen und -praktiken in einem Unternehmen eingeführt werden sollen, sollten nicht nur die oben aufgelisteten Punkte, sondern weitere Informationsquellen berücksichtigt werden, beispielsweise:

- Die ISTQB® Lehrpläne, Foundation und Advanced Level
- Im Kapitel Literatur dieses Lehrplans genannte Bücher
- Referenzmodelle, wie beispielsweise in Abschnitt 8.3 beschrieben.

# 16. Index