Certified Tester

Advanced Level Syllabus
Test Analyst

VERSION 2012

International Software Testing Qualifications Board

Deutschsprachige Ausgabe. Herausgegeben durch
Austrian Testing Board, German Testing Board e.V. &
Swiss Testing Board

Arbeitsgruppe Advanced Level Test Analyst: Judy McKay (Vorsitzende), Mike Smith, Erik Van Veenendaal; 2010-2012.

Änderungsübersicht

Version	Datum	Bemerkungen
ISEB v1.1	4. September 2001	ISEB Practitioner Syllabus
ISTQB 1.2E	September 2003	ISTQB® Advanced Level Syllabus von EOQ-SG
V2007	12. Oktober 20007	Certified Tester Advanced Level Syllabus Version 2007 (Lehrplan Aufbaukurs Certified Tester)
D100626	26. Juni 2010	Einarbeitung von den in 2009 angenommenen Änderungen, Aufteilung der jeweiligen Kapitel für die einzelnen Module
D101227	27. Dezember 2010	Annahme von Formatänderungen und Korrekturen, die keinen Einfluss auf den Sinngehalt haben
D2011	31. Oktober 2011	Übergang zum geteilten Lehrplan, Überarbeitung von Lernzielen und Textänderungen zum Angleich an die Lernziele. Hinzufügen der geschäftlichen Ergebnisse
Alpha 2012	9. März 2012	Einarbeitung aller Reviewbemerkungen der nationalen Boards, die zur Version Oktober 2011 eingingen
Beta 2012	7. April 2012	Einarbeitung von Reviewbemerkungen der nationalen Boards zur Alpha-Version
Beta 2012	7. April 2012	Beta-Version zur Vorlage bei der Hauptversammlung
Beta 2012	8. Juni 2012	Beta-Freigabe für nationale Boards nach technischer Überarbeitung
Beta 2012	27. Juni 2012	Einarbeitung von EWG- und Glossar-Bemerkungen
Beta_1 German Edition	16. November 2012 – 02 März.2013	Lokalisierung nach Übersetzung durch ATB, GTB und STB
Beta_2 German Edition	16. November 2012 – 17. März.2013	Lokalisierung nach Übersetzung durch ATB, GTB und STB; Überarbeitung auf der Basis des Inputs der Arbeitsgruppe Glossar unter Leitung von Matthias Hamburg.
German Final	19. April 2012	Kommentare und Hinweise zur BETA_2 Fassung eingearbeitet

Inhaltsverzeichnis

Dank

Dieses Dokument wurde von einem Kernteam der Arbeitsgruppe „Advanced Level Syllabus – Advanced Test Analyst" des International Software Testing Qualifications Board erstellt. Dieser Arbeitsgruppe gehörten an: Judy McKay (Vorsitzende), Mike Smith, Erik Van Veenendaal.

Das Kernteam bedankt sich beim Reviewteam und bei den nationalen Boards für die konstruktiven Vorschläge und Beiträge.

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

Graham Bath, Rex Black, Maria Clara Choucair, Debra Friedenberg, Bernard Homès (stellvertretender Vorsitzender), Paul Jorgensen, Judy McKay, Jamie Mitchell, Thomas Mueller, Klaus Olsen, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Mike Smith (Vorsitzender), Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto.

Folgende Personen haben an Review, Kommentierung und der Abstimmung über diesen Lehrplan mitgearbeitet (in alphabetischer Reihenfolge):

Graham Bath, Arne Becher, Rex Black, Piet de Roo, Frans Dijkman, Mats Grindal, Kobi Halperin, Bernard Homès, Maria Jönsson, Junfei Ma, Eli Margolin, Rik Marselis, Don Mills, Gary Mogyorodi, Stefan Mohacsi, Reto Mueller, Thomas Mueller, Ingvar Nordstrom, Tal Pe'er, Raluca Madalina Popescu, Stuart Reid, Jan Sabak, Hans Schaefer, Marco Sogliani, Yaron Tsubery, Hans Weiberg, Paul Weymouth, Chris van Bael, Jurian van der Laar, Stephanie van Dijk, Erik van Veenendaal, Wenqiang Zheng, Debi Zylbermann.

Das Swiss Testing Board (STB), das Austrian Testing Board (ATB) und das German Testing Board (GTB) danken ebenso dem Review-Team der deutschsprachigen Fassung 2012 (in alphabetischer Reihenfolge): Andra Calancea (ATB), Renzo Cerquozzi (STB), Andreas Gäng (Leitung, STB), Matthias Hamburg (GTB), Martin Klonk (ATB), Clemens Mucker (ATB), Anke Mündler (ATB), Helmut Pichler (ATB), Horst Pohlmann (GTB), Thomas Puchter (ATB), Angela Schiehl (ATB), Maud Schlich (GTB), Stephanie Ulrich (GTB), Stephan Weissleder (GTB), Mario Winter (GTB), Peter Zimmerer (GTB), Wolfgang Zuser ATB).

Dieses Dokument wurde von der Hauptversammlung des ISTQB® am 19. 10. 2012 offiziell freigegeben.

0. Einführung in den Advanced Level Syllabus TA

0.1 Zweck dieses Dokuments

Dieser Lehrplan bildet die Grundlage für das Softwaretest-Qualifizierungsprogramm der Aufbaustufe (Advanced Level) für Test Analysten. Das ISTQB® stellt den Lehrplan folgenden Adressaten zur Verfügung:

1. Nationalen/regionalen Boards zur Übersetzung in die jeweilige Landessprache und zur Akkreditierung von Trainingsprovider. 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 Lehrpläne 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 der internationalen 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 Nutzung dieses Lehrplans für andere Zwecke genehmigen, wenn diese vorab eine entsprechende schriftliche Genehmigung einholen.

0.2 Übersicht

Der Advanced Level besteht aus drei separaten Lehrplänen:

- Testmanager
- Test Analyst
- Technical Test Analyst

Im Übersichtsdokument zum Advanced Level [ISTQB_AL_OVIEW] sind folgende Informationen enthalten:

- Mehrwert für jeden Lehrplan
- Zusammenfassung der einzelnen Lehrpläne
- Beziehung zwischen den Lehrplänen
- Beschreibung der kognitiven Stufen des Wissens
- Anhänge

0.3 Prüfungsrelevante Lernziele

Die Lernziele unterstützen die geschäftlichen Ziele und dienen zur Ausarbeitung der Prüfung für den Test Analyst Advanced Level (CTAL-TA). Allgemein gilt, dass der gesamte Inhalt des vorliegenden Advanced Level Lehrplans entsprechend der Lernziele der kognitiven Stufe K1 geprüft werden kann. Dies bedeutet, dass die Prüfungskandidaten Begriffe oder Konzepte erkennen, sich an sie erinnern und sie wiedergeben können. Die relevanten Lernziele der kognitiven Stufen K2 (Verstehen), K3 (Anwenden) und K4 (Analysieren) werden immer zu Beginn des jeweiligen Kapitels angegeben.

1. Testprozess - 300 Minuten

Begriffe

abstrakter Testfall, Endekriterien, konkreter Testfall, logischer Testfall, Testdurchführung, Testentwurf, Testplanung, Testrealisierung, Teststeuerung

Lernziele zum Thema Testprozess

1.2 Testen im Softwarelebenszyklus

TA-1.2.1 (K2) Sie können erklären, wie und warum sich der Zeitpunkt und Grad der Beteiligung des Test Analysten beim Einsatz von unterschiedlichen Lebenszyklusmodellen unterscheiden

1.3 Testplanung, -überwachung und –steuerung

TA-1.3.1 (K2) Sie können die Aktivitäten des Test Analysten in Zusammenhang mit der Testplanung und Teststeuerung zusammenfassen

1.4 Testanalyse

TA-1.4.1 (K4) Sie können ein vorgegebenes Szenario, einschließlich der Projektbeschreibung und des Lebenszyklusmodells, analysieren und die Aufgaben des Test Analysten in den Testanalyse- und Testentwurfsphasen bestimmen

1.5 Testentwurf

TA-1.5.1 (K2) Sie können erklären, weshalb die Stakeholder die Testbedingungen verstehen sollten

TA-1.5.2 (K4) Sie können ein Projektszenario analysieren und bestimmen, welche konkreten und logischen Testfälle am besten geeignet sind

1.6 Testrealisierung

TA-1.6.1 (K2) Sie können die typischen Endekriterien für Testanalyse und Testentwurf beschreiben und erläutern, wie sich die Einhaltung dieser Kriterien auf den Testrealisierungsaufwand auswirkt

1.7 Testdurchführung

TA-1.7.1 (K3) Sie können für ein vorgegebenes Szenario die erforderlichen Schritte und Überlegungen für die Durchführung der Tests festlegen

1.8 Bewertung von Endekriterien und Bericht

TA-1.8.1 (K2) Sie können erklären, warum genaue Statusinformationen über die Testfalldurchführung wichtig sind

1.9 Abschluss der Testaktivitäten

TA-1.9.1 (K2) Sie können Beispiele für Arbeitsergebnisse anführen, die der Test Analyst beim Abschluss der Testaktivitäten liefern sollte

1.1 Einführung

Der ISTQB® Foundation Level Lehrplan beschreibt den fundamentalen Testprozess, der aus folgenden Hauptaktivitäten besteht:

- Testplanung und Teststeuerung
- Testanalyse und Testentwurf
- Testrealisierung und Testdurchführung
- Bewertung von Endekriterien und Bericht
- Abschluss der Testaktivitäten

In den Advanced Level Lehrplänen werden einige dieser Aktivitäten getrennt behandelt. Hierdurch sollen eine weitere Verfeinerung und Optimierung der Prozesse, eine bessere Anpassung an den Softwareentwicklungs-Lebenszyklus, sowie eine effektive Testüberwachung und –steuerung ermöglicht werden. Die Testprozessaktivitäten werden nun wie folgt behandelt:

- Testplanung, -überwachung und –steuerung
- Testanalyse
- Testentwurf
- Testrealisierung
- Testdurchführung
- Bewertung von Endekriterien und Bericht
- Abschluss der Testaktivitäten

Diese Testprozessaktivitäten können sequenziell durchgeführt werden, oder es können einige davon parallel erfolgen. Beispielsweise kann der Testentwurf parallel zur Testrealisierung durchgeführt werden (beim explorativen Testen). Schwerpunktmäßig befasst sich der Test Analyst mit der Festlegung der richtigen Tests und Testfälle, sowie mit deren Entwurf und Durchführung. Auch wenn es wichtig ist, die anderen Stufen im Testprozess zu verstehen, so erfolgt der überwiegende Teil der Aufgaben des Test Analysten während der Testanalyse, Testentwurf, Testrealisierung und Testdurchführung des Testprojekts.

Fortgeschrittene Tester sind mit einigen schwierigen Aufgaben konfrontiert, wenn sie die in diesem Lehrplan beschriebenen unterschiedlichen Testaspekte in ihren Organisationen, Teams und Aufgabenbereichen einführen. Dabei ist es wichtig, die unterschiedlichen Softwarelebenszyklus-modelle sowie die den zu testenden Systemtyp zu berücksichtigen, da diese Faktoren die Vorgehensweise beim Testen erheblich beeinflussen können.

1.2 Testen im Softwarelebenszyklus

Das langfristige Einbinden des Testens in den Softwarelebenszyklus sollte als Teil der Teststrategie betrachtet und beschrieben werden. Je nach gewähltem Softwarelebenszyklusmodell unterscheidet sich der Zeitpunkt, zu dem der Test Analyst an den Testaktivitäten beteiligt wird. Auch der Grad der Beteiligung, Zeitaufwand, verfügbare Informationen und Erwartungen können stark variieren. Da Testprozesse nicht isoliert ausgeführt werden, muss der Test Analyst die Zeitpunkte kennen, wann Informationen für die anderen organisatorischen Aufgabenbereiche geliefert werden können, wie z.B.:

- Anforderungsanalyse und Anforderungsmanagement - Anforderungsreviews
- Projektmanagement - Eingaben für den Zeitplan
- Konfigurations- und Änderungsmanagement - Tests zur Verifizierung von Softwareversionen, Versionskontrolle
- Softwareentwicklung - Antizipieren, welche Software wann geliefert wird
- Softwarewartung - Fehlermanagement, Fehlerbearbeitungszeit (d.h. Zeit von Fehlerfindung bis Fehlerbehebung)

- technischer Support - genaue Dokumentierung von Lösungsalternativen
- Erstellen technischer Dokumentation (z.B. Datenbankdesignspezifikationen) - Eingaben für diese Dokumente sowie technisches Review dieser Dokumente

Testaktivitäten müssen an das gewählte Softwarelebenszyklusmodell angepasst werden, das sequenziell, iterativ oder inkrementell sein kann. Beim 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 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 Systemtest-Realisierung dauert dagegen oft bis wenige Tage vor Beginn der Testdurchführung.
- Die Testdurchführung beginnt, wenn alle Eingangskriterien 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 Endekriterien erfüllt sind.
- Während der gesamten Testdurchführung werden die Endekriterien bewertet und die Testergebnisse berichtet, normalerweise mit zunehmender Häufigkeit und Dringlichkeit, je näher der Projektendtermin rückt.
- Die Testabschlussaktivitäten folgen, wenn die Endekriterien 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.

Bei iterativen und inkrementellen Modellen werden die Aufgaben möglicherweise nicht in der gleichen Reihenfolge ausgeführt, oder es werden einzelne Aufgaben ganz weggelassen. Bei einem iterativen Modell wird beispielsweise für jede Iteration eine reduzierte Version des fundamentalen Testprozesses verwendet. Hierbei erfolgen Testanalyse, Testentwurf, Testrealisierung und –durchführung, sowie Bewertung und Bericht in jeder Iteration, während die Planung zu Beginn und die abschließenden Berichte zum Ende des Projektes erfolgen. In agilen Projekten wird oft ein leichtgewichtiger Prozess eingesetzt und eine deutlich engere Zusammenarbeit praktiziert. So können Änderungen im Projekt einfacher erfolgen. Da agile Lebenszyklusmodelle einen „leichtgewichtigen“ Prozess verwenden, wird weniger umfangreiche Testdokumentation erstellt zu Gunsten einer schnelleren Kommunikation, beispielsweise in den täglichen “Stand-ups” (kurze Besprechungen, die in der Regel 10 bis 15 Minuten dauern, bei denen die Teilnehmer sich nicht hinsetzen und engagiert bei der Sache sind).

Von allen Lebenszyklusmodellen wird der Test Analyst bei agilen Projekten am frühesten involviert. Es ist zu erwarten, dass der Test Analyst ab der Anfangsphase des Projekts involviert ist und mit den Entwicklern bei der Erstellung einer ersten Architektur und des Softwareentwurfs zusammenarbeitet. Reviews müssen nicht formal sein, werden aber fortlaufend im Zuge der Softwareentwicklung durchgeführt. Es ist damit zu rechnen, dass der Test Analyst im gesamten Projektverlauf involviert ist und dem Team zur Verfügung stehen sollte. Aufgrund dieser intensiven Beteiligung sollten die Projektmitglieder agiler Projekte einzelnen Projekten fest zugeordnet werden und voll in alle Aspekte des Projekts einbezogen sein.

Die iterativen/inkrementellen Lebenszyklusmodelle reichen von den agilen Methoden, bei denen Änderungen im Laufe der Softwareentwicklung antizipiert werden, zu den iterativen/inkrementellen Softwareentwicklungsmodellen, die im V-Modell vorhanden sind (diese werden manchmal auch als eingebettet-iterativ bezeichnet). Bei letzteren Modellen ist davon auszugehen, dass der Test Analyst

an den normalen Planungs- und Entwurfsaktivitäten beteiligt ist, dann aber in Zusammenhang mit Entwicklung, Testen, Änderung und Einführung der Software eine interaktivere Rolle übernimmt.

Unabhängig vom eingesetzten Softwareentwicklungslebenszyklusmodell ist es wichtig, dass der Test Analyst versteht, welche Erwartungen bezüglich seines Einsatzes bestehen, und wann dieser Einsatz erfolgen sollte. In der Praxis kommen viele Mischformen zum Einsatz, wie beispielsweise das iterative Modell innerhalb des V-Modells (siehe oben). Häufig muss der Test Analyst die effektivste Rolle bestimmen und sich für diese Rolle einsetzen, anstatt sich auf ein definiertes, festgelegtes Modell zu verlassen, das den richtigen Zeitpunkt einer Involvierung vorgibt.

1.3 Testplanung, -überwachung und -steuerung

Dieses Kapitel behandelt die Prozesse der Testplanung, -überwachung und -steuerung.

1.3.1 Testplanung

Die Testplanung findet größtenteils bei der Einleitung des Testprozesses statt. Dazu gehört die Identifizierung der Aktivitäten und Ressourcen, die zur Erfüllung der in der Testrichtlinie festgelegten Prinzipien und Zele nötig sind. Bei der Testplanung ist es wichtig, dass der Test Analyst in Zusammenarbeit mit dem Testmanager die folgenden Punkte berücksichtigt bzw. plant:

- Es muss sicherstellt werden, dass die Testkonzepte nicht nur das funktionale Testen betreffen. Im Testkonzept sollten alle Testarten berücksichtigt und entsprechend eingeplant werden. Beispielsweise kann der Test Analyst zusätzlich zum funktionalen Testen auch für den Benutzbarkeitstest zuständig sein. Auch diese Testart muss im Testkonzept dokumentiert werden.
- Die Testschätzungen müssen mit dem Testmanager überprüft werden, um sicherzustellen, dass für die Beschaffung und Validierung der Testumgebung ausreichend Zeit vorgesehen wird.
- Der Konfigurationstest muss geplant werden. Wenn mehrere Arten von Prozessoren, Betriebssystemen, virtuellen Maschinen, Browsern und diversen Peripheriegeräten zu vielen unterschiedlichen Konfigurationen kombiniert werden können, müssen Testverfahren eingeplant werden, die diese Kombinationen angemessen überdecken.
- Das Prüfen der Dokumentation muss geplant werden. Benutzer erhalten sowohl Software als auch Dokumentation. Die Dokumentation muss genau und richtig sein, denn nur dann ist sie effektiv. Der Test Analyst muss Zeit für die Verifizierung der Dokumentation vorsehen. Möglicherweise muss der Test Analyst mit technischen Redakteuren zusammenarbeiten und bei der Erstellung von Daten für Screenshots und Videoclips behilflich sein.
- Das Testen der Installationsprozeduren muss geplant werden. Installationsprozeduren, wie auch Backup- und Wiederherstellungsprozeduren, müssen ausreichend getestet werden. Diese Prozeduren können kritischer sein als die Software selbst; wenn die Software sich nicht installieren lässt, dann wird sie überhaupt nicht verwendet. Die Planung dieser Aufgabe kann schwierig sein, da der Test Analyst häufig die ersten Testaktivitäten an einem System durchführt, das vorkonfiguriert ist, bei dem aber die endgültigen Installationsprozesse noch nicht festgelegt sind.
- Die Anpassung des Testens an den Softwarelebenszyklus muss geplant werden. Eine sequenzielle Durchführung der Aufgaben passt häufig nicht in den Zeitplan. In der Praxis müssen oft einige Aufgaben (zumindest teilweise) gleichzeitig ausgeführt werden. Der Test Analyst muss wissen, welches Lebenzyklusmodell gewählt wurde und was von ihm während des Entwurfs, der Entwicklung und Implementierung der Software erwartet wird. Dazu gehört auch, dass Zeit für Bestätigungstests und Regressionstests eingeplant wird.
- Es muss ausreichend Zeit für die Identifizierung und Analyse von Risiken zusammen mit dem funktionsübergreifenden Team vorgesehen werden. Auch wenn der Test Analyst in der Regel nicht für die Organisation der Risikomanagementsitzungen zuständig ist, ist zu erwarten, dass

er aktiv an diesen Aktivitäten beteiligt ist

Zwischen Testbasis, Testbedingungen und Testfällen 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. Der Test Analyst ist meist die Person, die diese Beziehungen am besten bestimmen und die Abhängigkeiten so weit wie möglich voneinander trennen kann.

1.3.2 Testüberwachung und -steuerung

Während die Testüberwachung und Teststeuerung normalerweise zu den Aufgaben des Testmanagers gehören, trägt der Test Analyst die Messungen bei, die die Steuerung durch den Testmanager ermöglichen.

Während des gesamten Softwarelebenszyklus sollte eine Vielzahl von quantitativen Daten gesammelt werden (z.B. der prozentuale Anteil der abgeschlossenen Planungsaktivitäten, der prozentuale Anteil der erzielten Überdeckung, die Anzahl der bestandenen und nicht bestandenen Testfälle). In jedem Fall muss eine Baseline (Referenzkonfiguration) definiert werden, damit der Fortschritt in Bezug auf diese verfolgt werden kann. Während sich Testmanager mit dem Erstellen und Berichten der zusammengefassten Metrikinformationen befassen, ist der Test Analyst für das Sammeln der Informationen für jede der Metriken zuständig. Jeder abgeschlossene Testfall, jeder Fehlerbericht und jeder erzielte Meilenstein fließt in die Projektmetriken ein. Es ist wichtig, dass die Informationen, die in die verschiedenen Werkzeuge eingegeben werden, so genau wie möglich sind, damit die Metriken der Realität entsprechen.

Genaue Metriken ermöglichen es dem Management, ein Projekt zu managen (überwachen) und bei Bedarf Änderungen herbeizuführen (steuern). Wenn beispielsweise für einen Bereich der Software eine hohe Anzahl von Fehlern berichtet wird, kann dies ein Indiz dafür sein, dass in diesem Bereich zusätzlicher Testaufwand benötigt wird. Informationen über Anforderungs- und Risikoüberdeckung (Rückverfolgbarkeit) können dazu dienen, die noch ausstehenden Aufgaben zu priorisieren und Ressourcen zuzuweisen. Anhand von Informationen über Grundursachen lassen sich Bereiche für Prozessverbesserungen identifizieren. Wenn die aufgezeichneten Daten genau sind, dann lässt sich das Projekt steuern, und es können genaue Statusinformationen an die Stakeholder berichtet werden. Zukünftige Projekte lassen sich effektiver planen, wenn bei der Planung Daten aus zurückliegenden Projekten berücksichtigt werden. Für genaue Daten gibt es unzählige Verwendungsmöglichkeiten. Es ist Teil der Aufgaben des Test Analysten sicherzustellen, dass die Daten genau und objektiv sind, und dass sie rechtzeitig zur Verfügung stehen.

1.4 Testanalyse

Bei der Testplanung wird der Umfang des Testprojekts definiert. Diese Definition nutzt der Test Analyst zur:

- Analyse der Testbasis
- Identifizierung der Testbedingungen

Damit der Test Analyst die Testanalyse effektiv durchführen kann, sollten die folgenden Eingangskriterien erfüllt sein:

- Es gibt ein Dokument, in dem das Testobjekt beschrieben ist, und das als Testbasis dienen kann.
- Das Review dieses Dokuments hat brauchbare Ergebnisse geliefert, und das Dokument wurde nach dem Review je nach Bedarf aktualisiert.
- Für die Durchführung der restlichen Testaktivitäten zu diesem Testobjekt steht ein angemessenes Budget und Zeit zur Verfügung.

Testbedingungen werden typischerweise durch eine Analyse der Testbasis und der Testziele identifiziert. Falls die Dokumentation veraltet ist oder falls es keine Dokumentation gibt, können die Testbedingungen anhand von Gesprächen mit den relevanten Betroffenen identifiziert werden (z.B. in Workshops oder bei der Planung der Sprints). Anhand dieser Bedingungen wird dann bestimmt, was mit den in der Teststrategie und/oder im Testkonzept festgelegten Testverfahren getestet wird.

Während die Testbedingungen gewöhnlich spezifisch für das zu testende Element bzw. Objekt sind, sollte der Test Analyst grundsätzlich folgendes beachten:

- In der Regel empfiehlt es sich, die Testbedingungen mit unterschiedlichem Detaillierungsgrad zu definieren. Zunächst werden die abstrakten Bedingungen identifiziert, die die allgemeinen Ziele des Testens definieren, wie z.B. “Nachweisen, dass Bildschirm X funktioniert”. Danach werden detailliertere Bedingungen als Basis für spezifische Testfälle identifiziert, wie z.B. “Nachweisen, dass Bildschirm X eine Kontonummer zurückweist, die ein Zeichen zu wenig hat”. Wenn die Testbedingungen nach dieser hierarchischen Methode definiert werden, kann dies sicherstellen, dass die Abdeckung der abstrakten Elemente ausreichend ist.
- Falls Produktrisiken definiert wurden, müssen die Testbedingungen, die für jedes der Produktrisiken notwendig sind, identifiziert werden, und sie müssen zum jeweiligen Risiko rückverfolgbar sein.

Nach Abschluss der Testanalyse sollte der Test Analyst wissen, welche spezifischen Tests entworfen werden müssen, damit die Erfordernisse der festgelegten Bereiche des Testprojekts abgedeckt werden.

1.5 Testentwurf

Auch der Testentwurf basiert auf dem in der Testplanung bestimmten Umfang. Der Test Analyst entwirft die Tests, die im weiteren Verlauf des Testprozesses realisiert und durchgeführt werden. Der Testentwurfsprozess beinhaltet die folgenden Aktivitäten:

- Es wird bestimmt, in welchen Bereichen konkrete und in welchen Bereichen abstrakte Testfälle am besten geeignet sind
- Es werden die Testfallentwurfsverfahren festgelegt, die die benötigte Testabdeckung liefern.
- Es werden Testfälle entworfen, mit denen die identifizierten Testbedingungen ausgeführt 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.

Je nach Art der Tests, die entworfen werden, kann eines der Eingangskriterien für den Testentwurf die Verfügbarkeit von Werkzeugen sein, die für die Entwurfsaufgaben eingesetzt werden.

Beim Entwerfen der Tests sind die folgenden Punkte unbedingt zu berücksichtigen:

- Für manche Testelemente ist es besser, nur die Testbedingungen zu spezifizieren, anstatt detaillierte Testablaufbeschreibungen zu definieren. In diesen Fällen sollten die Testbedingungen so spezifiziert werden, dass sie als Leitfaden für Tests ohne Testablaufspezifikation verwendet werden können.
- Die Endekriterien für die Tests müssen eindeutig festgelegt werden.
- Tests sollten so entworfen werden, dass sie für andere Tester verständlich sind, und nicht nur für den Autor. Falls der Autor nicht die Person ist, die den Test durchführt, dann müssen andere Personen die zuvor spezifizierten Tests lesen und verstehen, damit sie die Testziele und die relative Wichtigkeit des Tests nachvollziehen können.

- Die Tests müssen auch für andere Stakeholder verständlich sein, beispielsweise für Entwickler, da diese die Tests prüfen werden, und für die Leiter eines Audits, die die Tests eventuell genehmigen müssen.
- Tests sollten so entworfen werden, dass sämtliche Interaktionen der Software mit Akteuren (z.B. Endnutzer, andere Systeme) abgedeckt sind, und nicht nur die Interaktionen, die über die Benutzerschnittstelle für den Nutzer sichtbar sind. Auch Kommunikationen zwischen Prozessen, Batch-Verarbeitung und Interrupts beeinflussen die Software und können potenziell Fehlerwirkungen auslösen. Daher muss der Test Analyst auch Tests entwerfen, die diese Risiken beherrschen[1].
- Tests sollten entworfen werden, um die Schnittstellen zwischen den verschiedenen Testobjekten zu testen.

1.5.1 Logische und konkrete Testfälle

Logische Testfälle liefern eine Richtlinie und spezifizieren, was getestet werden soll; sie stellen dem Test Analyst jedoch frei, die tatsächlichen Daten und sogar den Ablauf für die Durchführung des Tests zu variieren. Logische Testfälle können eine bessere Abdeckung als konkrete Testfälle erreichen, weil bei jeder Durchführung ein wenig variiert wird. Dies führt allerdings auch zu einer geringeren Reproduzierbarkeit. Logische Testfälle werden am besten dann eingesetzt, wenn die Anforderungen nicht gut spezifiziert sind, wenn der Test Analyst, der die Tests durchführen wird, Erfahrung sowohl im Testen als auch mit dem Produkt hat, und wenn keine formale Dokumentation benötigt wird (wenn z.B. kein Audit durchgeführt wird). Logische Testfälle können schon früh im Anforderungsprozess spezifiziert werden, wenn die Anforderungen noch nicht gut definiert sind. Diese Testfälle können zur Erstellung konkreter Testfälle verwendet werden, wenn die Anforderungen genauer definiert und stabiler sind. In diesem Fall erfolgt die Erstellung der Testfälle sequenziell von den logischen zu den konkreten Testfällen, wobei lediglich die konkreten Testfälle durchgeführt werden.

Eine Aufgabe des Test Analysten ist es, die besten Arten von Testfällen für eine vorgegebene Situation zu bestimmen. Konkrete Testfälle liefern sämtliche spezifische Informationen und Prozeduren, die der Tester zur Durchführung des Testfalls und zur Verifizierung der Ergebnisse benötigt (ggf. einschließlich der benötigten Daten). Konkrete Testfälle sind nützlich, wenn die Anforderungen gut spezifiziert sind, wenn das Testpersonal nicht viel Erfahrung hat, und wenn die Tests extern verifiziert werden sollen, z.B. durch ein Audit. Konkrete Testfälle sind ausgezeichnet reproduzierbar (d.h. andere Tester erzielen die gleichen Ergebnisse), sie können jedoch auch einen erheblichen Wartungsaufwand bedeuten, und sie engen die Fantasie der Tester bei der Durchführung der Tests eher ein.

1.5.2 Testfälle entwerfen

Testfälle werden in einer schrittweisen Ausarbeitung und Verfeinerung der identifizierten Testbedingungen entworfen. Dabei nutzen die Tester Testverfahren, die in der Teststrategie und/oder im Testkonzept festgelegt wurden. Testfälle sollten wiederholbar, nachprüfbar und zur Testbasis (z.B. Anforderungen) rückverfolgbar sein.

Zum Entwurf von Testfällen gehört die Identifizierung von:

- Zielsetzung
- Vorbedingungen, wie z.B. projektbezogene oder lokale Testumgebungen (Testvorrichtungen) und deren geplante Bereitstellung, Zustand des Systems usw.
- Anforderungen für Testdaten (sowohl Eingabedaten für den Testfall als auch Daten, die im System vorhanden sein müssen, damit der Testfall durchgeführt werden kann)

[1] Gebräuchliche Synonyme in diesem Kontext: Risikovermeidung, Risikosteuerung und –bewältigung, Risikoüberwachung

- erwarteten Ergebnissen
- Nachbedingungen, wie z.B. beeinflusste Daten, Zustand des Systems, Auslöser für nachfolgende Prozessabläufe usw.

Der Detaillierungsgrad für die Testfälle wirkt sich sowohl auf die Kosten für die Entwicklung als auch auf die Wiederholbarkeit bei der Testfalldurchführung aus, und sollte daher festgelegt sein, bevor die Testfälle tatsächlich erstellt werden. Weniger detaillierte Testfälle geben dem Test Analyst mehr Flexibilität bei der Durchführung der Tests und die Möglichkeit, potenziell interessante Bereiche genauer zu untersuchen. Weniger Detail bedeutet jedoch auch eine geringere Reproduzierbarkeit.

Oft stellt die Definition der erwarteten Testergebnisse eine besondere Herausforderung dar. Die manuelle Berechnung ist häufig mühsam und fehleranfällig. Es gilt, wenn möglich ein automatisiertes Testorakel zu finden oder zu erstellen. Bei der Suche nach erwarteten Ergebnissen achten die Tester nicht nur auf Bildschirmausgaben, sondern auch auf Daten und Nachbedingungen in der Testumgebung. Wenn die Testbasis klar definiert ist, ist die Identifizierung des korrekten Ergebnisses 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 muss der Test Analyst 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 schwierig gestalten. Ein Testorakel ist daher unverzichtbar. Die Durchführung von Testfällen ohne Möglichkeit, die Richtigkeit der Ergebnisse zu überprüfen, hat nur sehr begrenzten Wert, denn daraus können sich ungültige Fehlerberichte und unbegründetes Vertrauen in das System ergeben.

Die beschriebenen Aktivitäten lassen sich in 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, User Stories oder dem Programmcode selbst basieren. Es ist zu beachten, dass diese Aktivitäten in allen Teststufen stattfinden, auch wenn das Ziel des Tests variieren kann. Zum Beispiel soll funktionales Testen in der Komponenten-Teststufe sicherstellen, dass eine bestimmte Komponente die Funktionalität liefert, die im detaillierten Entwurf dieser Komponente spezifiziert wurde. Funktionales Testen in der Integrationsteststufe soll verifizieren, dass Komponenten zusammenwirken und dadurch die benötigte Funktionalität liefern. Im Systemtest sollte die End-to-End-Funktionalität Ziel des Testens sein. Bei der Analyse und beim Entwurf von Tests ist es wichtig, die Stufe, auf die der Test abzielt, sowie das Objekt des Tests zu berücksichtigen. Dies hilft dabei, den benötigten Detaillierungsgrad und Werkzeuge zu bestimmen, die möglicherweise benötigt werden (z.B. Treiber und Platzhalter in der Komponententest).

Bei der Entwicklung der Testbedingungen und Testfälle entstehen in der Regel verschiedene Dokumente, welche in unterschiedliche Arbeitsergebnisse resultieren. In der Praxis gibt es jedoch große Unterschiede in 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
- Standards und/oder Vorschriften, die eingehalten werden müssen
- das Lebenszyklusmodell (bei agilen Methoden soll der Umfang der Dokumentation gerade ausreichend sein)
- die Anforderung an Rückverfolgbarkeit von der Testbasis über die Testanalyse bis hin zum Testentwurf

Je nach Testumfang können sich Testanalyse- und Testentwurfsphase auch mit den Qualitätsmerkmale des Testobjekts bzw. der Testobjekte befassen. ISO 25000 [ISO25000], der ISO 9126 ersetzt, bietet dafür eine nützliche Referenzgrundlage. Beim Testen von Hardware-/Softwaresystemen können weitere Merkmale relevant sein.

Der Testanalyse- und Testentwurfsprozess lässt sich durch eine Verknüpfung mit Reviews und statischer Analyse qualitativ verbessern. Die Durchführung von Testanalyse und Testentwurf ist oft eigentlich eine Form des statischen Testens, da dabei Probleme in den Basisdokumenten gefunden werden können. Die Durchführung des Testanalyse- und des Testentwurfsprozesses basierend auf der Anforderungsspezifikation ist beispielsweise eine ausgezeichnete Vorbereitung auf die Review-Sitzung für die Anforderungsspezifikation. Das Lesen der Anforderungen, um diese für den Entwurf von Tests zu verwenden, setzt voraus, dass die Anforderung verstanden wird und dass bewertet werden kann, ob die Anforderung erfüllt ist. Häufig werden bei dieser Aktivität Anforderungen entdeckt, die unklar sind, oder die nicht testbar sind, oder für die keine Abnahmekriterien definiert sind. Auch Arbeitsergebnisse wie Testfälle, Risikoanalysen und Testkonzepte sollten Reviews und statischen Analysen unterzogen werden.

Bei manchen Projekten, z.B. bei Projekten mit einem agilen Lebenszyklusmodell, sind die Anforderungen möglicherweise nur minimal dokumentiert. Diese liegen manchmal in Form von User Stories vor, in denen kleine, nachweisbare Bestandteile der Funktionalität beschrieben werden. In den User Stories sollte auch eine Definition der Abnahmekriterien enthalten sein. Wenn nachgewiesen werden kann, dass die Software die Abnahmekriterien erfüllt, dann ist sie gewöhnlich soweit, dass sie mit anderen fertiggestellten Funktionalitäten integriert werden kann, oder sie wurde bereits integriert, um ihre Funktionalität nachzuweisen.

In der Testentwurfsphase lassen sich Anforderungsdetails an die Testinfrastruktur genau 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 Tests nötig ist).

Die Endekriterien von Testanalyse und Testentwurf variieren je nach den Projektparametern; es sollte jedoch überlegt werden, ob die in diesen beiden Abschnitten behandelten Punkte in den definierten Endekriterien enthalten sein sollten. Es ist zu beachten, dass die Kriterien messbar sein sollten, und es muss sichergestellt werden, dass alle Informationen und Vorbereitungen für die nachfolgenden Schritte zur Verfügung gestellt werden.

1.6 Testrealisierung

Die Testrealisierung ist die Umsetzung des Testentwurfs. Dazu gehört die Erstellung automatisierter Tests, die Festlegung der Reihenfolge für die Ausführung der (manuellen und automatisierten) Tests, die endgültige Festlegung der Testdaten und Testumgebungen, und die Erstellung eines Testausführungsplans einschließlich Ressourcenzuweisung. Wenn dies vorliegt, kann die Ausführung der Testfälle beginnen. Dazu gehört auch die Überprüfung anhand von expliziten und impliziten Eingangskriterien für die betroffene Teststufe. Außerdem muss sichergestellt werden, dass die Endekriterien der vorangegangenen Prozessschritte erfüllt wurden. Wenn die Endekriterien für die Teststufe oder für einen Schritt im Testprozess übersprungen wurden, dann ist es möglich, dass sich dies in Form von Verzögerungen, mangelnder Qualität und unerwartetem Mehraufwand auf den Testrealisierungsaufwand auswirkt. Es ist daher wichtig, dass alle Endekriterien erfüllt sind, bevor mit der Testrealisierung begonnen wird.

Beim Festlegen der Reihenfolge der Durchführung sind einige Faktoren zu berücksichtigen. Manchmal kann es Sinn machen, die Testfälle in Testsuiten (d.h. Gruppen von Testfällen) zusammenzustellen. Dies kann bei der Organisation des Testens hilfreich sein, so dass die zusammengehörigen Testfälle miteinander ausgeführt werden. Falls eine risikoorientierte Teststrategie verwendet wird, ist die Reihenfolge, in der die Testfälle ausgeführt werden, möglicherweise durch die Risikopriorität vorgeschrieben. Außerdem gibt es noch viele weitere Faktoren, die die Reihenfolge bestimmen, z.B. Verfügbarkeit der richtigen Personen, Ausrüstungen, Daten und die zu testende Funktionalität. Es ist nicht ungewöhnlich, dass Code schrittweise freigegeben wird und der Testaufwand mit der Reihenfolge koordiniert werden muss, in der die Software für den Test bereitgestellt wird. Insbesondere bei inkrementellen Lebenszyklusmodellen

muss der Test Analyst den Ablauf mit dem Entwicklungsteam koordinieren und sicherstellen, dass die Software in einer Reihenfolge freigegeben wird, die das Testen ermöglicht. Während der Testrealisierung sollte der Test Analyst die Reihenfolge der manuellen und automatisierten Tests endgültig festlegen und bestätigen. Dabei ist sorgfältig zu prüfen, ob es Faktoren gibt, die eine bestimmte Reihenfolge der Tests zwingend vorgeben. Abhängigkeiten müssen dokumentiert und überprüft werden.

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 beispielsweise die definierte Norm DO-178B/ED 12B [RTCA DO-178B/ED-12B] der United States Federal Aviation Administration.

Wie oben erwähnt, sind Testdaten für das Testen nötig, und die Datenmengen können ziemlich umfangreich sein. Während der Testrealisierungsphase erzeugt der Test Analyst Eingabe- und Umgebungsdaten, die in Datenbanken oder anderen Repositories abgelegt werden. Außerdem erzeugt der Test Analyst Daten für datengetriebene automatisierte Tests sowie für manuelle Tests.

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. Je nach unvorhergesehenen Änderungen, Testergebnissen oder aus sonstigen Erwägungen können während der Testdurchführung Änderungen an der Testumgebung nötig werden. Falls derartige Änderungen während der Durchführung anfallen, muss geprüft werden, welche Auswirkungen diese auf die bereits durchgeführten Tests haben.

In der Testrealisierungsphase müssen die Tester 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, Fehlermanagement, Testprotokollierung und Testmanagement. Außerdem müssen Test Analysten die Verfahren verifizieren, mit denen die Daten für Testendekriterien und für Berichte über die Testergebnisse gesammelt werden.

Für die Testrealisierung empfiehlt sich ein ausgewogener Ansatz, der bei der Testplanung festgelegt wurde. Risikoorientierte, analytische Teststrategien werden beispielsweise oft mit dynamischen Teststrategien kombiniert. In diesem Fall wird ein Teil des Testrealisierungsaufwands für das Testen ohne vorgegebene Testskripte verwendet.

Testen ohne Testskripte sollte aber nicht ad hoc oder ziellos sein, da der Zeitaufwand und die erzielte Überdeckung schwer einzuschätzen sind, falls keine zeitliche Eingrenzung definiert ist. Im Laufe der Zeit haben Tester verschiedene erfahrungsbasierte Verfahren entwickelt, wie Fehlerangriffe, intuitive Testfallermittlung 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 „leichtgewichtig" und oft sehr effektiv beim Auffinden von Fehlern, sie haben aber auch Nachteile. Sie benötigen die Erfahrung des Test Analysten, ihre Dauer kann schwer abzuschätzen sein, sie liefern oft nicht die notwendigen Informationen zur Testüberdeckung, und ohne gute Dokumentation oder ein spezifisches Werkzeug für Regressionstests können sie schwierig zu wiederholen sein.

1.7 Testdurchführung

Die Testdurchführung beginnt, sobald das Testobjekt zur Verfügung steht und die Eingangskriterien für die Testdurchführung erfüllt sind oder auf sie verzichtet wird. Die Tests sollten gemäß dem bei der Testrealisierung festgelegten Plan durchgeführt werden. Allerdings sollte den Test Analysten 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. Die Einbindung von Testverfahren mit und ohne Testablaufbeschreibung (z.B. explorative Verfahren) hilft zu verhindern, dass Bereiche im Test nicht abgedeckt werden, die sich aus Lücken in der vorgeschriebenen Überdeckung ergeben; außerdem lässt sich damit das Problem umgehen, dass Testwiederholungen zunehmend ihre Wirksamkeit verlieren.

Kernstück der Testdurchführung ist der Vergleich zwischen tatsächlichen und erwarteten Testergebnissen. Test Analysten müssen diese Aufgabe mit großer Aufmerksamkeit und Sorgfalt erledigen, denn alle Arbeit bei Entwurf und Realisierung des Tests kann umsonst gewesen sein, wenn Fehlerwirkungen übersehen werden (falsch negatives Ergebnis), oder wenn korrektes Verhalten irrtümlicherweise als inkorrekt gedeutet wird (falsch positives 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 Fehlermanagement enthält Kapitel 6.

Wird eine Fehlerwirkung entdeckt, sollte zunächst die Testdokumentation (Testspezifikation, Testfall usw.) einer genauen Bewertung unterzogen werden, um deren Richtigkeit sicherzustellen. Testspezifikationen können aus verschiedenen Gründen inkorrekt sein. Stellt sich heraus, dass die Testspezifikation tatsächlich 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 Testverfahren, wie z.B. beim explorativen Testen, entkräften.) Da sich Testobjekt, Testmittel und Testumgebungen weiterentwickeln können, muss aus der Protokollierung hervorgehen, welche Version und welche Umgebungskonfigurationen getestet wurden. Die Testprotokollierung liefert eine chronologische Aufzeichnung aller relevanten Details der Testdurchführung.

Die Testergebnisse sind sowohl bei einzelnen Tests als auch bei Aktivitäten und 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 Tests protokolliert der Test Analyst die Informationen über die Testdurchführung; häufig erfolgt dies direkt in ein Testmanagementwerkzeug, das die Informationen über die Testdurchführung verfolgt. In manchen Fällen, wie bei der Testrealisierung, beeinflussen Vorschriften oder Audit-Anforderungen den genauen Umfang der Testprotokollierung.

In manchen Fällen können Nutzer oder Kunden am Durchführen des Tests beteiligt werden, 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.

Die folgenden Punkte sollten bei der Testdurchführung berücksichtigt werden:

- „Irrelevante" Merkwürdigkeiten (Nebeneffekte) müssen wahrgenommen und untersucht werden. Scheinbar irrelevante Beobachtungen oder Ergebnisse sind oft Hinweise auf mögliche Fehler, die sich (wie Eisberge) unter der Oberfläche verstecken.
- Es muss sichergestellt werden, dass das Produkt nichts tut, was es nicht tun soll. Zu den normalen Testaufgaben gehört es zu prüfen, ob das Produkt tut, was es tun soll. Der Test Analyst muss aber auch sicher sein, dass das Produkt nicht tut, was es nicht tun soll (beispielsweise zusätzliche, unerwünschte Funktionen).
- Testsuiten müssen entwickelt werden, die erwartungsgemäß im Laufe der Zeit größer werden und sich verändern. 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. Lücken in der Testabdeckung werden häufig während der Testdurchführung entdeckt. Die Entwicklung der Testsuite ist ein kontinuierlicher Prozess.
- Es sollten Notizen für kommende Testaufgaben gemacht werden. 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.
- Man sollte weder davon ausgehen, dass alle manuellen Tests wiederholt werden, noch ist es realistisch, dies zu erwarten. Wenn ein Problem vermutet wird, sollte der Test Analyst das Problem untersuchen und notieren. Keinesfalls sollte der Test Analyst davon ausgehen, dass es später entdeckt wird, wenn der Testfall wieder ausgeführt wird.
- Die Daten des Fehlerverfolgungswerkzeugs sollten für zusätzliche Testfälle genutzt werden. Es sollte überlegt werden, ob Testfälle für Fehlerzustände, die beim Testen ohne Testablaufspezifikation oder beim explorativen Testen gefunden wurden, zur Regressions-Testsuite hinzugefügt werden sollten.
- Fehlerzustände sollten vor dem Regressionstest aufgedeckt werden. Die Zeit für den Regressionstest ist häufig beschränkt, und die Aufdeckung von Fehlerzuständen im Regressionstest kann zu Verzögerungen führen. 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 also bereits aufgedeckt worden sein. Das bedeutet allerdings nicht, dass Regressionstests ganz entfallen sollten. Ihre Effektivität bei der Auffindung neuer Fehlerzustände ist aber geringer als bei anderen Testarten.

1.8 Bewertung von Endekriterien und Bericht

Für den Testprozess geht es bei der Überwachung des Testfortschritts vor allem darum, Informationen gemäß den Anforderungen für die Berichte zu sammeln. Dazu gehört auch, den Testfortschritt mit Bezug auf die Endekriterien zu messen. Wenn die Endekriterien bei der Testplanung festgelegt

wurden, ist möglicherweise eine Aufteilung der Kriterien in solche, die erfüllt werden müssen, und solche, die erfüllt werden sollten, bereits erfolgt. Möglicherweise ist vorgeschrieben, dass es keine offenen Fehlerzustände der Prioritätsklassen 1 und 2 geben darf und dass mindestens 95% aller Testfälle bestanden werden sollten. In einem solchen Fall wären die Endekriterien nicht erfüllt, wenn das verpflichtende nicht erfüllt ist; wenn aber 93% der Testfälle bestanden wurden, könnte das Projekt trotzdem für die nächste Stufe freigegeben werden. Die Endekriterien müssen klar spezifiziert sein, damit eine objektive Bewertung möglich ist.

Der Test Analyst liefert die Informationen, die der Testmanager zur Bewertung des Testfortschritts gegen die Endekriterien verwendet, und um sicherzustellen, dass die Daten richtig sind. Wenn das Testmanagementsystem für den Abschluss von Testfällen beispielsweise die folgenden Statuscodes liefert:

- Bestanden
- Nicht bestanden
- Bestanden mit Ausnahme

dann muss dem Test Analyst sehr klar sein, was diese bedeuten, und der jeweilige Status muss konsistent angewendet werden. Bedeutet “Bestanden mit Ausnahme”, dass ein Fehlerzustand gefunden wurde, dieser aber die Funktionalität des Systems nicht beeinträchtigt? Wie verhält es sich dann mit einem Benutzbarkeitsfehlerzustand, der die Benutzer verwirrt? Wenn die Bestehensquote (der Anteil der bestandenen Tests) ein verpflichtendes Endekriterium ist, dann kann es kritisch sein, ob ein Testfall als “Nicht bestanden” oder als “Bestanden mit Ausnahme“ gezählt wird. Außerdem muss berücksichtigt werden, wie Testfälle behandelt werden, die als “Nicht bestanden” eingestuft sind, bei denen die Ursache der Fehlerwirkung aber kein Fehlerzustand ist, sondern z.B. eine falsch konfigurierte Testumgebung. Falls es in Zusammenhang mit den verfolgten Metriken oder mit der Verwendung der Statusdaten Unstimmigkeiten gibt, muss der Test Analyst mit dem Testmanager eine Klärung herbeiführen, damit die Informationen im Projektverlauf richtig und genau verfolgt werden können.

Es ist durchaus üblich, dass der Test Analyst während der Testzyklen um einen Statusbericht gebeten wird, sowie um einen Beitrag zum Abschlussbericht am Ende des Testens. Für diese Aufgaben werden Metriken aus den Fehler- und Testmanagmentsystemen gesammelt, und die Gesamtüberdeckung und Fortschritt werden bewertet. Der Test Analyst sollte die Berichtswerkzeuge benutzen können und in der Lage sein, dem Testmanager die angeforderten Informationen zu liefern, damit dieser daraus die benötigten Informationen herausfiltern kann.

1.9 Abschluss der Testaktivitäten

Nachdem die Testdurchführung als abgeschlossen erklärt wurde, sollten die wichtigsten Ergebnisse des Testens festgehalten und entweder an die zuständige Person weitergeleitet oder archiviert werden. Man bezeichnet das als den Abschluss der Testaktivitäten. Der Test Analyst kann damit rechnen, dass er bei der Weiterleitung der Arbeitsergebnisse an die Personen oder Stellen, die diese benötigen, involviert ist. 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. Informationen über Testarbeitsergebnisse müssen klar dokumentiert sein, einschließlich der entsprechenden Links und einschließlich der entsprechenden Zugriffsberechtigungen.

Der Test Analyst sollte außerdem davon ausgehen, an Besprechungen über die Erfahrungen aus dem Testen (Bewertungssitzungen, “lessons learned“) teilzunehmen. In diesen Besprechungen können wichtige Erkenntnisse sowohl aus dem eigentlichen Testprozess als auch aus dem gesamten Softwarelebenszyklus dokumentiert und daraus Maßnahmen abgeleitet werden, damit die guten

Aspekte verstärkt und schlechte Aspekte eliminiert oder zumindest kontrolliert werden. Der Test Analyst ist für diese Besprechungen eine sachkundige Informationsquelle und muss mitwirken, wenn es darum geht, Informationen für Prozessverbesserungen zusammenzutragen. Falls nur der Testmanager an der Sitzung teilnimmt, dann muss der Test Analyst diesem die relevanten Informationen zur Verfügung stellen, damit das Projekt zutreffend dargestellt wird.

Ergebnisse, Protokolle, Berichte und andere Dokumente und Arbeitsergebnisse müssen im Konfigurationsmanagement-System archiviert werden. Hierfür ist häufig der Test Analyst zuständig. Dies ist eine wichtige Testabschlussaktivität, besonders wenn die Informationen für ein zukünftiges Projekt verwendet werden sollen.

Während der Testmanager in der Regel bestimmt, welche Informationen archiviert werden sollen, sollte der Test Analyst auch überdenken, welche Informationen benötigt würden, falls das Projekt in der Zukunft erneut gestartet wird. Diese Überlegungen am Ende eines Projektes können Monate an Aufwand einsparen, wenn das Projekt tatsächlich zu einem späteren Zeitpunkt oder mit einem anderen Team neu gestartet wird.

2. Testmanagement: Zuständigkeiten des Test Analysten - 90 Minuten

Begriffe

Produktrisiko, Risikoanalyse, Risikoidentifizierung/-identifikation, Risikomanagement, risikoorientierter Test, Risikostufe, Risikoüberwachung/-beherrschung, Teststrategie, Testüberwachung

Lernziele zum Thema Testmanagement: Zuständigkeiten des Test Analysten

2.2 Testfortschritt überwachen und steuern

TA-2.2.1 (K2) Sie können die verschiedenen Arten von Informationen erklären, die beim Testen verfolgt werden müssen, um eine geeignete Überwachung und Steuerung des Projekts zu ermöglichen

2.3 Verteiltes Testen, Outsourcing und Insourcing

TA-2.3.1 (K2) Sie können Beispiele für gute Kommunikationspraktiken anführen, die für eine Tätigkeit in einer Testumgebung geeignet sind, in der 24 Stunden rund um die Uhr getestet wird

2.4 Aufgaben des Test Analysten beim risikoorientierten Testen

TA-2.4.1 (K3) Sie können für eine vorgegebene Projektsituation an der Risikoidentifizierung mitwirken, eine Risikobewertung durchführen und geeignete Maßnahmen zur Risikobeherrschung vorschlagen.

2.1 Einführung

Es gibt viele Bereiche, in denen der Test Analyst mit dem Testmanager interagiert und dem Testmanager Daten liefert. Dieser Abschnitt befasst sich jedoch schwerpunktmäßig mit bestimmten Bereichen des Testprozesses, in denen der Test Analyst einen sehr wesentlichen Beitrag leistet. Es wird erwartet, dass der Testmanager die benötigten Informationen vom Test Analyst anfordert.

2.2 Testfortschritt überwachen und steuern

Die fünf wesentlichen Aspekte für die Überwachung des Testfortschritts sind:

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

Während eines Projekts oder während des Betriebs werden Produktrisiken, Fehlerzustände, Testfälle und Überdeckungsgrad auf spezielle Art und Weise durch den Test Analyst gemessen und berichtet. Das Vertrauen in die Softwarequalität wird gewöhnlich subjektiv berichtet, auch wenn es durch Umfragen messbar ist. Das Zusammentragen der Informationen, auf die sich die Metriken stützen, gehört zu den täglichen Aufgaben des Test Analysten. Es ist zu beachten, dass die Richtigkeit der gesammelten Daten kritisch ist, da aus ungenauen Daten falsche Trenddaten erstellt werden, was zu falschen Schlussfolgerungen führen kann. Im schlimmsten Fall resultieren ungenaue Daten in falschen Managemententscheidungen und schädigen die Glaubwürdigkeit des Testteams.

Bei einer risikoorientierten Testvorgehensweise sollte der Test Analyst folgende Metriken verfolgen:

- Welche Risiken wurden durch Testen reduziert?
- Welche Risiken gelten als offene Risiken?

Die Risikobeherrschung erfolgt häufig mit Hilfe eines Werkzeugs das auch die Testdurchführung verfolgt (z.B. mit einem Testmanagementwerkzeug). Dies bedingt, dass die identifizierten Risiken mit den Testbedingungen in Bezug gesetzt werden, die sich wiederum auf die Testfälle beziehen, die die Risiken reduzieren, wenn sie ausgeführt werden und bestehen. Auf diese Weise werden die Informationen zur Risikoüberwachung automatisch mit aktualisiert, wenn die Testfälle aktualisiert werden. Dies kann für manuelle und automatische Tests erfolgen.

Die Fehlerverfolgung erfolgt in der Regel mit Hilfe eines Fehlermanagementwerkzeugs. Es werden nicht nur Fehlerzustände aufgezeichnet, sondern auch bestimmte Informationen über die Fehlerklasse jedes einzelnen Fehlerzustands. Mit diesen Daten werden Trenddiagramme und Grafiken erstellt, aus denen der Testfortschritt und Softwarequalität hervorgehen. Informationen zur Fehlerklasse werden im Kapitel Fehlermanagement ausführlicher behandelt. Der Lebenszyklus kann den Umfang der aufzuzeichnenden Fehlerdokumentation sowie die Methoden zur Aufzeichnung der Informationen beeinflussen.

Im Zuge der Testdurchführung sollten die Statusinformationen zu den Testfällen aufgezeichnet werden. Gewöhnlich erfolgt die Aufzeichnung durch ein Testmanagementwerkzeug, kann bei Bedarf aber auch manuell erfolgen. Zu den aufgezeichneten Testfallinformationen können gehören:

- Status der erstellten Testfälle (z.B. entworfen, geprüft)
- Status der durchgeführten Testfälle (z.B. bestanden, nicht bestanden, blockiert, ausgelassen)
- Informationen über die durchgeführten Testfälle (z.B. Datum und Uhrzeit, Tester, verwendete Daten)

- Artefakte der durchgeführten Testfälle (z.B. Screenshots, testbegleitende Protokolle)

Wie die identifizierten Risiken sollten auch die Testfälle mit den jeweiligen Anforderungen in Bezug gebracht werden, die sie abdecken. Der Test Analyst sollte beachten, dass wenn Testfall A, der sich auf die Anforderung A bezieht, und wenn dies der einzige Testfall ist, der diese Anforderung abdeckt, dann gilt die Anforderung als erfüllt, sobald Testfall A ausgeführt und bestanden wird. Das muss nicht immer richtig sein. In vielen Fällen wird mehr als nur ein Testfall benötigt, um eine Anforderung gründlich zu testen, aber aus Zeitmangel wird nur ein Teil dieser Tests tatsächlich erstellt. Beispiel: Wenn 20 Testfälle nötig sind, um die Implementierung einer Anforderung gründlich zu testen, aber nur 10 Testfälle entworfen und ausgeführt werden, dann wird als Anforderungsüberdeckung eine Überdeckung von 100 % angegeben, auch wenn tatsächlich nur eine Überdeckung von 50% erzielt wurde. Die genaue Verfolgung der Überdeckung sowie der Anforderungen mit dem Status „Geprüft“ können als eine vertrauensbildende Maßnahme eingesetzt werden .

Umfang (und Detaillierungsgrad) der aufgezeichneten Informationen hängen von mehreren Faktoren ab, nicht zuletzt vom Softwarelebenszyklus. In agilen Projekten werden typischerweise weniger Statusinformationen aufgezeichnet, weil das Team enger zusammenarbeit und mehr persönliche Kommunikation stattfindet.

2.3 Verteiltes Testen, Outsourcing und Insourcing

Manchmal wird nicht der gesamte Testaufwand von einem einzigen Testteam erbracht, das sich aus Kollegen des Projektteams zusammensetzt, und die am selben Ort wie das Projektteam arbeiten. Wird der Testaufwand von Personen an mehreren Standorten erbracht, dann spricht man von verteiltem Testen. Wenn das Testen an einem einzigen Standort erfolgt, kann es als zentralisiertes Testen bezeichnet werden. Wenn Personen an einem oder mehreren Standorten testen, die nicht Mitarbeiter des Unternehmens sind und nicht am Standort des Projektteams arbeiten, spricht man von Outsourcing. Wenn Personen testen, die nicht Mitarbeiter des Projektteams sind, aber am selben Standort arbeiten, dann spricht man von Insourcing.

Bei Projekten, bei denen Teile des Testteams an unterschiedlichen Standorten oder sogar für unterschiedliche Unternehmen arbeiten, muss der Test Analyst einer effektiven Kommunikation und Informationsaustausch besondere Aufmerksamkeit widmen. In manchen Organisationen läuft das Testen rund um die Uhr; hier übergibt das Team in einer Zeitzone die Arbeit an das Team in einer anderen Zeitzone, das mit dem Testen fortfährt. Hier ist spezielle Planung seitens der Test Analysten notwendig, die die Arbeit übergeben bzw. entgegen nehmen. Gute Planung ist zwar wichtig, um die Zuständigkeiten zu verstehen, doch es ist unerlässlich, dass der Test Analyst sicherstellt, dass die richtigen Informationen verfügbar sind.

Wenn die Kommunikation nicht mündlich erfolgen kann, dann muss schriftliche Kommunikation ausreichen. Es müssen E-Mails, Statusberichte und die effektive Nutzung von Testmanagement- und Fehlerverfolgungswerkzeugen eingesetzt werden. Wenn im Testmanagementwerkzeug Tests einzelnen Personen zugeordnet werden können, dann kann dies auch als Planungswerkzeug eingesetzt werden und bietet eine einfache Möglichkeit, Arbeit zwischen Personen zu übergeben. Fehlerzustände, die genau aufgezeichnet werden, können bei Bedarf an Kollegen zur Nachverfolgung weitergeleitet werden. Die effektive Nutzung dieser Kommunikationsmittel ist für Organisationen, die sich nicht auf die tägliche persönliche Interaktion verlassen können, von entscheidender Bedeutung.

2.4 Aufgaben des Test Analysten beim risikoorientierten Testen

2.4.1 Übersicht

Der Testmanager trägt die Gesamtverantwortung für das Aufsetzen und Management einer risikoorientierten Teststrategie. In der Regel wird der Testmanager die Unterstützung des Test Analysten anfordern, um sicherzustellen, dass die risikoorientierte Vorgehensweise fachgerecht implementiert wird.

Der Test Analyst sollte an den folgenden risikoorientierten Testaufgaben aktiv mitwirken:

- Risikoidentifizerung
- Risikobewertung
- Risikobeherrschung

Diese Aufgaben erfolgen iterativ über den gesamten Projektlebenszyklus, und befassen sich mit auftretenden Risiken, Änderung von Prioritäten und mit der regelmäßigen Bewertung und Kommunikation des Risikostatus.

Test Analysten sollten innerhalb des vom Testmanager für das Projekt festgelegten risikoorientierten Testrahmens tätig sein. Sie sollten ihr Wissen über die Risiken des Geschäftsbereichs beisteuern, wie z.B. Risiken in Zusammenhang mit der Betriebssicherheit, geschäftlichen und wirtschaftlichen Aspekten sowie politischen Faktoren.

2.4.2 Risikoidentifizierung

Wenn man im Risikoidentifizierungs-Prozess eine möglichst breite Basis der Betroffenen involviert, desto höher ist die Wahrscheinlichkeit, dass dabei die größtmögliche Menge an wichtigen Risiken aufgedeckt wird. Da die Test Analysten häufig über spezifisches Wissen über den jeweiligen Geschäftsbereich des zu testenden Systems verfügen, sind sie in besonderer Weise dazu geeignet, Experten-Interviews mit Experten und Benutzern des Geschäftsbereichs zu führen, unabhängige Bewertungen zu erstellen, Risiko-Vorlagen zu verwenden und deren Einsatz zu ermöglichen, Risiko-Workshops zu leiten, Brainstorming mit potenziellen und aktuellen Nutzern durchzuführen, Checklisten zu spezifizieren, sowie auf Erfahrungen aus der Vergangenheit mit ähnlichen Systemen oder Projekten zurückzugreifen. Der Test Analyst sollte insbesondere mit den Benutzern und anderen Experten des Geschäftsbereichs eng zusammenarbeiten, um die Bereiche mit Geschäftsrisiken zu bestimmen, die beim Testen berücksichtigt werden sollten. Der Test Analyst kann auch besonders hilfreich sein, wenn es darum geht, potenzielle Auswirkungen von Risiken auf Benutzer und Betroffene zu identifizieren.

Zu den Risiken, die bei einem Projekt identifiziert werden könnten, gehören beispielsweise:

- Probleme mit der Genauigkeit der Softwarefunktionalität, z.B. falsche Berechnungen
- Probleme mit der Benutzbarkeit, z.B. zu wenig Tastaturkürzel
- Probleme mit der Erlernbarkeit, z.B. keine Anweisungen für Benutzer an Hauptentscheidungspunkten

Das Testen spezifischer Qualitätsmerkmale wird in Kapitel 4 dieses Lehrplans behandelt.

2.4.3 Risikobewertung

Während es bei der Risikoidentifizierung darum geht, möglichst viele vorhandene Risiken zu identifizieren, befasst sich die Risikoanalyse mit der Untersuchung der identifizierten Risiken. Sie kategorisiert jedes Risiko und legt dafür Eintrittswahrscheinlichkeit und Auswirkungen fest.

Für die Festlegung der Risikostufe werden für jedes einzelne Risiko die Eintrittswahrscheinlichkeit und die Auswirkung eingeschätzt. Die Wahrscheinlichkeit des Auftretens bezeichnet oft die Wahrscheinlichkeit, dass das potenzielle Problem im zu testenden System vorhanden ist. Es geht also um ein technisches Risiko. Der Technical Test Analyst sollte mitwirken, wenn es darum geht, die

Risikostufe der einzelnen potenziellen technischen Risiken festzulegen und zu verstehen. Der Test Analyst sollte mitwirken, wenn es darum geht, die potenziellen geschäftlichen Auswirkungen des Problems zu verstehen, falls dieses auftritt.

Die Auswirkung bei Eintreten des Risikos bezeichnet oft das Ausmaß der Wirkung auf Benutzer, Kunden und/oder andere Betroffene. Bei der Auswirkung geht es also um ein Geschäftsrisiko. Der Test Analyst sollte mitwirken, wenn es darum geht, die potenziellen Auswirkungen auf Geschäft oder Benutzer für die einzelnen Risiken zu identifizieren und zu bewerten. Folgende Faktoren beeinflussen die Geschäftsrisiken:

- Häufigkeit der Nutzung der betroffenen Funktion
- entgangene Geschäfte
- mögliche finanzielle, ökologische oder soziale Verluste oder Haftungsansprüche
- zivil- oder strafrechtliche Maßnahmen
- Sicherheitsbedenken
- Geldstrafen, Aberkennung der Lizenz
- keine vernünftigen Lösungsalternativen
- Sichtbarkeit des Funktionsmerkmals
- das negative Erscheinungsbild, wenn Mängel bekannt werden, Negativschlagzeilen, Schaden für die Reputation
- Verlust von Kunden

Anhand der vorhandenen Informationen über das Risiko, muss der Test Analyst die Risikostufen für die Geschäftsrisiken basierend auf den vom Testmanager vorgegebenen Richtlinien festlegen. Diese können mit Begriffen (z.B. hoch, mittel, gering) oder quantitativ mit Zahlen angegeben werden. Wenn es nicht möglich ist, das Risiko objektiv anhand einer definierten Skala zu messen, kann die Messung nicht als eine echte quantitative Messung gelten. Auch die Messung von Wahrscheinlichkeit und finanzieller Kosten/Konsequenzen gestaltet sich in der Regel sehr schwierig. Die Risikostufe wird daher meist qualitativ bestimmt. Den qualitativen Werten können zwar Zahlen zugeordnet werden, aber auch dies macht die Messung nicht zu einer echten quantitativen Messung. Beispiel: Der Testmanager gibt vor, das Geschäftsrisiko mit einem Wert zwischen 1 und 10 zu kategorisieren (wobei 1 für das Risiko mit der höchsten Auswirkung auf das Geschäft steht). Sobald die Eintrittswahrscheinlichkeit (Bewertung des technischen Risikos) und die Auswirkung (Bewertung des Geschäftsrisikos) zugewiesen wurden, können diese beiden Werte miteinander multipliziert werden. So lässt sich die Kennzahl für das Gesamtrisiko für jedes einzelne Risiko bestimmen. Dieses Rating wird dann für die Priorisierung der Risikoreduzierungsaktivitäten benutzt. Bei manchen risikoorienten Testvorgehensweisen (wie z.B. PRISMA® [vanVeenendaal12]) werden die Risikowerte nicht kombiniert; dadurch können die technischen und geschäftlichen Risiken in der Testvorgehensweise getrennt behandelt werden.

2.4.4 Risikobeherrschung

Im Projekt sollten die Test Analysten folgende Aufgaben übernehmen:

- Produktrisiken reduzieren und zu diesem Zweck geeignete Testfälle einsetzen, die einwandfrei nachweisen, ob die Tests bestanden oder nicht bestanden wurden, sowie an Reviews der Software-Artefakte (wie z.B. Anforderungen, Entwürfe und Benutzerdokumentation) teilnehmen
- geeignete Maßnahmen zur Risikobeherrschung umsetzen, die in der Teststrategie und im Testkonzept festgelegt sind
- bekannte Risiken neu bewerten, basierend auf zusätzlichen Informationen, die im Verlauf des Projektes verfügbar werden, und anhand dieser Informationen die Eintrittswahrscheinlichkeit oder -Auswirkung, oder ggf. beide Informationen anzupassen
- neue Risiken erkennen, die durch Informationen, die bei Testen gewonnen werden, identifiziert werden

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 der Systemfreigabe 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. Test Analysten wirken bei der Bestimmung der möglichen Maßnahmen zur Risikobeherrschung mit: sie untersuchen Möglichkeiten, genaue Testdaten zu sammeln, sie erstellen und testen realistische Szenarien, und sie leiten oder überwachen Benutzbarkeitsstudien.

2.4.4.1 Tests priorisieren

Die Risikostufe wird auch für die Priorisierung der Tests verwendet. Beispiel: Ein Test Analyst stellt fest, dass es im Bereich der Transaktionsgenauigkeit eines Buchhaltungssystems ein hohes Risiko gibt. Um dieses Risiko zu beherrschen, arbeitet der Tester mit anderen Experten des Geschäftsbereichs zusammen, um eine solide Datenmenge zusammenzutragen, die verarbeitet werden kann und anhand derer sich die Genauigkeit verifizieren lässt. Ein weiteres Beispiel: Ein Test Analyst stellt fest, dass Probleme mit der Benutzung ein bedeutendes Risiko für ein neues Produkt sind. Anstatt den Benutzer-Abnahmetest abzuwarten, bei dem die Probleme entdeckt würden, erteilt der Test Analyst einem frühen Benutzbarkeitstest, der im Integrationstest stattfinden soll, eine hohe Priorität. So sollen Benutzbarkeitsprobleme frühzeitig im Testprozess identifziert und behoben werden. Eine solche Priorisierung sollte so früh wie möglich in der Planung berücksichtigt werden, damit die Zeit für die notwendigen Tests zum benötigten Zeitpunkt eingeplant werden kann.

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 allerdings dafür sorgen, dass jedes Risiko mindestens einmal abgedeckt wird (Breadth-first, d.h. Testen in die Breite).

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 weiter getestet werden soll, oder ob das Restrisiko an die Benutzer, Kunden, Helpdesks oder technische Unterstützung und/oder an das Bedienpersonal weitergegeben wird.

2.4.4.2 Anpassung des Testens an weitere Testzyklen

Die Risikobewertung ist keine einmalige Aktivität, die vor Beginn der Testrealisierung erfolgt, sondern ein fortlaufender Prozess. Alle weiteren geplanten Testzyklen sollten auf Basis einer neuen Risikoanalyse erfolgen, bei der verschiedene Faktoren berücksichtigt werden, wie z.B.:

- mögliche neue oder deutlich veränderte Produktrisiken
- instabile oder fehleranfällige Bereiche des Systems, die im Laufe des Testens identifiziert wurden
- Risiken als Folge behobener Fehler
- Fehler, die sich beim Testen als typische Fehler herausgestellt haben
- nicht ausreichend getestete Bereiche des Systems (Bereiche mit geringer Testüberdeckung)

Wenn noch Zeit für weitere Tests zur Verfügung steht, können eventuell Risiken mit niedrigerer Risikostufe abgedeckt werden.

3. Testverfahren - 825 Minuten

Begriffe

anforderungsbasierter Test, anwendungsfallbasierter Test, Äquivalenzklassenbildung, Checklisten-basiertes Testen, , Entscheidungstabellentest, erfahrungsbasiertes Verfahren, exploratives Testen, fehlerbasiertes Verfahren, Fehlertaxonomie, Grenzwertanalyse, intuitive Testfallermittlung, Klassifikationsbaumverfahren, kombinatorisches Testen, orthogonale Arrays, Testen mit orthogonalen Arrays, paarweises Testen, spezifikationsorientiertes Verfahren, Test-Charta, Ursache-Wirkungs-Graph-Analyse, User-Story-basiertes Testen, Wertebereichsanalyse, zustandsbasierter Test

Lernziele zum Thema Testverfahren

3.2 Spezifikationsorientierte Testverfahren

TA-3.2.1 (K2) Sie können den Einsatz von Ursache-Wirkungs-Graphen erklären

TA-3.2.2 (K3) Sie können Testfälle aus vorgegebenen Spezifikationselementen unter Verwendung der Äquivalenzklassenbildung erstellen, um einen vorgegebenen Überdeckungsgrad zu erzielen

TA-3.2.3 (K3) Sie können Testfälle aus vorgegebenen Spezifikationselementen unter Verwendung der Grenzwertanalyse erstellen, um einen vorgegebenen Überdeckungsgrad zu erzielen

TA-3.2.4 (K3) Sie können Testfälle aus vorgegebenen Spezifikationselementen unter Verwendung des Entscheidungstabellentests erstellen, um einen vorgegebenen Überdeckungsgrad zu erzielen

TA-3.2.5 (K3) Sie können Testfälle aus vorgegebenen Spezifikationselementen unter Verwendung des zustandsbasierten Tests erstellen, um einen vorgegebenen Überdeckungsgrad zu erzielen

TA-3.2.6 (K3) Sie können Testfälle aus vorgegebenen Spezifikationselementen unter Verwendung des paarweisen Testens erstellen, um einen vorgegebenen Überdeckungsgrad zu erzielen

TA-3.2.7 (K3) Sie können Testfälle aus vorgegebenen Spezifikationselementen unter Verwendung des Klassifikationsbaumverfahrens erstellen, um einen vorgegebenen Überdeckungsgrad zu erzielen

TA-3.2.8 (K3) Sie können Testfälle aus vorgegebenen Spezifikationselementen unter Verwendung des anwendungsfallbasierten Testens erstellen, um einen vorgegebenen Überdeckungsgrad zu erzielen

TA-3.2.9 (K2) Sie können erklären, wie in einem agilen Projekt User Stories eingesetzt werden, um das Testen zu unterstützen

TA-3.2.10 (K3) Sie können Testfälle aus vorgegebenen Spezifikationselementen unter Verwendung der Wertebereichsanalyse erstellen, um einen vorgegebenen Überdeckungsgrad zu erzielen

TA-3.2.11 (K4) Sie können ein System oder dessen Anforderungsspezifikation analysieren, um zu bestimmen, welche Fehlerarten wahrscheinlich gefunden werden und die geeigneten spezifikationsorientierten Testverfahren auswählen

3.3 Fehlerbasierte Verfahren

TA-3.3.1 (K2) Sie können die Anwendung fehlerbasierter Testverfahren beschreiben und deren Einsatz gegen den Einsatz spezifikationsorientierter Testverfahren abgrenzen

TA-3.3.2 (K4) Sie können eine vorgegebene Fehlertaxonomie bezüglich ihrer Anwendbarkeit in einer vorgegebenen Situation analysieren und verwenden dafür Kriterien für eine gute Taxonomie

3.4 Erfahrungsbasierte Verfahren

TA-3.4.1 (K2) Sie können das Prinzip der erfahrungsbasierten Verfahren erklären, sowie deren Vor- und Nachteile mit spezifikationsorientierten und fehlerbasierten Testverfahren vergleichen

TA-3.4.2 (K3) Sie können für ein vorgegebenes Szenario explorative Tests spezifizieren und erklären, wie die Ergebnisse berichtet werden können

TA-3.4.3 (K4) Sie können für eine vorgegebene Projektsituation festlegen, welche spezifikationsorientierten, fehlerbasierten oder erfahrungsbasierten Verfahren für bestimmte Ziele einzusetzen sind

3.1 Einführung

Die in diesem Kapitel behandelten Testentwurfsverfahren lassen sich in die folgenden Kategorien einteilen:

- spezifikationsorientiert (auch verhaltensbasiert oder Black-Box)
- fehlerbasiert
- erfahrungsbasiert

Die Verfahren ergänzen sich gegenseitig und können soweit anwendbar für jede Testaktivität unabhängig von der Teststufe eingesetzt werden.

Es ist zu beachten, dass alle drei Kategorien von Testverfahren sowohl für das Testen funktionaler als auch nicht-funktionaler Qualitätsmerkmale verwendet werden können. Das Testen nicht-funktionaler Qualitätsmerkmale wird im nächsten Kapitel behandelt.

Die in den nachfolgenden Abschnitten behandelten Testentwurfsverfahren konzentrieren sich primär auf die Festlegung optimaler Testdaten (z.B. Äquivalenzklassen) oder auf die Ableitung von Testabläufen (z.B. Zustandsmodelle). Es ist üblich, die Verfahren zur Erstellung kompletter Testfälle zu kombinieren.

3.2 Spezifikationsorientierte Testverfahren

Spezifikationsorientierte Testverfahren werden auf Testbedingungen angewendet, um Testfälle basierend auf einer Analyse der Testbasis für eine der Komponente oder System abzuleiten ohne dabei deren interne Struktur zu berücksichtigen.

Gemeinsame Merkmale der spezifikationsorientierten Testverfahren sind:

- Entsprechend dem Testverfahren werden während des Testentwurfs Modelle erstellt (z.B. Zustandsübergangsdiagramme und Entscheidungstabellen) Aus diesen Modellen werden die Testbedingungen systematisch abgeleitet

Einige Verfahren liefern Kriterien für den Überdeckungsgrad als Maß für die Testentwurfs- und Testdurchführungsaufgaben. Wenn die Kriterien für den Überdeckungsgrad vollständig erfüllt sind, heißt das nicht, dass die Menge von Tests vollständig ist. Es bedeutet vielmehr, dass das Modell keine weiteren Tests vorschlägt, um die Überdeckung gemäß dem vorliegenden Testverfahren zu erhöhen.

Spezifikationsorientierte Tests basieren meist auf der Anforderungsdokumentation für das System. Da die Anforderungsspezifikation das Systemverhalten beschreiben sollte, vor allem hinsichtlich seiner Funktionalität, lassen sich aus den spezifizierten Anforderungen Tests ableiten, die das Verhalten des Systems prüfen. Es kann vorkommen, dass die Anforderungen nicht dokumentiert sind, sondern implizit vorliegen, beispielsweise wenn die Funktionalität eines vorhandenen Systems ersetzt werden soll.

Es gibt mehrere spezifikationsorientierte Testverfahren, die für unterschiedliche Arten von Software und unterschiedliche Szenarien eingesetzt werden. In den nachfolgenden Abschnitten wird die Anwendbarkeit der einzelnen Verfahren aufgezeigt, sowie einige Einschränkungen und Schwierigkeiten, auf die der Test Analyst stoßen könnte. Des Weiteren wird die Methode zur Messung der Testüberdeckung beschrieben, sowie die Fehlerarten, die mit diesen Verfahren aufgedeckt werden.

3.2.1 Äquivalenzklassenbildung

Die Äquivalenzklassenbildung wird eingesetzt, um die Anzahl der Testfälle zu reduzieren, die erforderlich sind, um die Verarbeitung von Eingabe- und Ausgabewerten, internen Werten und zeitbezogenen Werten effektiv zu testen. Die durch Aufteilung (engl. partitioning) erstellten Äquivalenzklassen (auch Äquivalenzpartitionen genannt) enthalten Wertemengen, die auf dieselbe Weise verarbeitet werden. Indem aus einer Äquivalenzklasse ein repräsentativer Wert ausgewählt wird, wird die Überdeckung aller Werte in derselben Äquivalenzklasse angenommen.

Anwendbarkeit
Dieses Verfahren ist in allen Teststufen anwendbar und ist dann geeignet, wenn es wahrscheinlich ist, dass die in Gruppen aufgeteilten Werte für die Software auf dieselbe Weise verarbeitet werden und bei der Verwendung durch die Anwendung nicht interagieren. Die Auswahl von Wertemengen (Äquivalenzklassen) können sowohl für gültige Daten als auch für ungültige Daten gebildet werden (d.h. Klassen mit Werten, die für die zu testende Software als ungültig zu behandeln sind). Dieses Verfahren ist dann am nützlichsten, wenn es mit der Grenzwertanalyse kombiniert wird und die getesteten Daten auch die Werte im Bereich der Grenzen der einzelnen Äquivalenzklassen mit einschließen. Das Verfahren ist sehr gebräuchlich für den Smoke-Test eines neuen Builds oder eines neuen Releases, da damit sehr schnell festgestellt wird, ob die grundlegende Funktionalität erfüllt ist.

Einschränkungen/Schwierigkeiten
Falls die Annahme, dass die Werte in der Äquivalenzklasse auf genau dieselbe Art und Weise verarbeitet werden, falsch ist, dann ist es möglich, dass mit diesem Verfahren Fehler übersehen werden. Auch müssen die Äquivalenzklassen sorgfältig ausgewählt werden. Beispiel: Für ein Eingabefeld für positive und negative Zahlen empfiehlt es sich, zwei gültige Äquivalenzklassen zu bilden, eine für die positiven und eine für die negativen Zahlen, da diese wahrscheinlich unterschiedlich verarbeitet werden. Je nachdem, ob die Null zulässig ist oder nicht, würde dafür eine weitere Äquivalenzklasse gebildet. Um die bestmöglichen Äquivalenzklassen zu bestimmen ist es wichtig, dass der Test Analyst die zugrunde liegenden Verarbeitungsprozesse versteht.

Überdeckung
Die Überdeckung wird bestimmt, indem man die Anzahl der überdeckten Äquivalenzklassen durch die Anzahl der insgesamt identifizierten Äquivalenzklassen teilt. Wenn für eine Äquivalenzklasse mehrere Werte verwendet werden, erhöht dies die prozentuale Überdeckung nicht.

Fehlerarten
Mit diesem Verfahren werden funktionale Fehlerzustände beim Verarbeiten verschiedener Datenwerte gefunden.

3.2.2 Grenzwertanalyse

Die Grenzwertanalyse wird eingesetzt, um die Werte an den Grenzen der eingeteilten Äquivalenzklassen zu testen. Es gibt zwei unterschiedliche Vorgehensweisen: das Testen mit zwei und das Testen mit drei Werten. Beim Testen mit zwei Werten, wird der Grenzwert (direkt auf der Grenze) und der Wert, der so knapp wie möglich unmittelbar über der Grenze liegt (durch das kleinstmögliche Inkrement), getestet. Beispiel: Wenn die Äquivalenzklasse die Werte 1 bis 10 in Schritten von 0,5 enthält, dann würden zum Testen der oberen Grenze die Werte 10 und 10,5 verwendet. Für die untere Grenze würden die Werte 1 und 0,5 verwendet. Die Grenzen definieren sich aus den maximalen und minimalen Werten in der festgelegten Äquivalenzklasse.

Für den Grenzwerttest mit drei Werten, werden die Werte unterhalb, auf und oberhalb der Grenze verwendet. Im geschilderten Beispiel wären das die Werte 9,5, 10 und 10,5 für die obere Grenze, bzw. 1,5, 1 und 0,5 für die untere Grenze. Die Entscheidung, ob mit zwei oder mit drei Werten getestet werden soll, orientiert sich am identifizierten Risiko des zu testenden Elements, wobei Elemente mit höheren Risiken mit drei Werten getestet werden sollten.

Anwendbarkeit
Dieses Verfahren ist in allen Teststufen anwendbar und ist dann geeignet, wenn die Elemente Äquivalenzklassen durch eine Ordnungsrelation miteinander vergleichbar sind. Die Ordnungsrelation in den Äquivalenzklassen ist notwendig, da das Konzept vorsieht, dass Werte an oder über der Grenze liegen. Zum Beispiel bildet ein Zahlenbereich eine geordnete Äquivalenzklasse, wohingegen eine Klasse, die ausschließlich aus rechteckigen Objekten besteht, nichtgeordnet ist und keine Grenzwerte hat. Außer für Zahlenbereiche kann die Grenzwertanalyse auch angewendet werden für:

- Numerische Attribute nicht-numerischer Variablen (z.B. Länge)
- Schleifen, einschließlich von Schleifen in Anwendungsfällen
- Gespeicherte Datenstrukturen
- Physikalische Objekte (einschließlich Speicher)
- Zeitbestimmte Aktivitäten

Einschränkungen/Schwierigkeiten
Da die Genauigkeit dieses Verfahrens von der genauen Festlegung der Äquivalenzklassen abhängt, gelten auch dieselben Einschränkungen und Schwierigkeiten. Der Test Analyst muss die Inkremente bei den gültigen und ungültigen Werten genau berücksichtigen, da sonst eine korrekte Auswahl der zu testenden Werte nicht möglich ist. Durch Grenzwertanalyse können nur geordnete Äquivalenzklassen getestet werden, diese beschränken sich jedoch nicht auf eine Gruppe gültiger Eingaben. Beispiel: Wenn eine Anzahl von Zellen einer Tabellenkalkulationsanwendung zu testen ist, dann enthält eine Äquivalenzklasse alle Zellen bis zur maximalen Anzahl, einschließlich der Zelle an der Grenze; die nächste Äquivalenzklasse beginnt dagegen mit der Zelle genau über der Grenze.

Überdeckung
Die Überdeckung wird bestimmt, indem man die Anzahl der getesteten Grenzwerte durch die Anzahl der insgesamt identifizierten Grenzwerte teilt (unter Nutzung entweder der 2-Wert oder der 3-Wert-Methode). Mit dieser Berechnung wird die prozentuale Überdeckung für die Grenzwertanalyse ermittelt.

Fehlerarten
Mit der Grenzwertanalyse werden verschobene oder fehlende Grenzen zuverlässig aufgedeckt, und es können zusätzliche Grenzen gefunden werden. Mit diesem Verfahren werden Fehlerzustände in Zusammenhang mit der Verarbeitung von Grenzwerten aufgedeckt, insbesondere bei logischen Bedingungen mit „kleiner-als“ und „grösser-als“ (d.h. wenn Grenzverschiebungen vorliegen). Es können mit diesem Verfahren auch nicht-funktionale Fehlerzustände aufgedeckt werden, beispielsweise Abweichungen bei den Lastgrenzen (z.B. das System unterstützt 10.000 gleichzeitige Benutzer).

3.2.3 Entscheidungstabellen

Entscheidungstabellen werden verwendet, um das Zusammenwirken zwischen Kombinationen von Bedingungen zu testen. Entscheidungstabellen liefern eine geeignete Methode, um zu verifizieren, dass alle vorhandenen Kombinationen von Bedingungen getestet werden und dass alle möglichen Kombinationen von Bedingungen in der zu testenden Software berücksichtigt wurden. Ziel des Entscheidungstabellentests ist, dass jede mögliche Kombination von Bedingungen, Beziehungen und Beschränkungen getestet wird. Wenn jede mögliche Kombination getestet wird, können die Entscheidungstabellen sehr groß werden. Wenn nicht alle möglichen Kombinationen getestet werden, sondern auf intelligente Art und Weise die „interessanten“ Kombinationen für den Test ausgewählt werden, wird dies als reduzierter Entscheidungstabellentest bezeichnet. Bei dieser Methode wird die Anzahl der Kombinationen auf jene mit unterschiedlichen Ausgaben reduziert; redundante und unrealistische Tests werden weggelassen. Die Entscheidung, ob ein vollständiger oder ein reduzierter Entscheidungstabellentest durchgeführt wird, orientiert sich meist am Risiko. [Copeland03]

Anwendbarkeit
Dieses Verfahren wird meist in den Teststufen Integrations-, System- und Abnahmetest angewendet. Abhängig vom Programmcode ist es auch beim Modultest anwendbar, falls die Komponente für Entscheidungslogik zuständig ist. Dieses Verfahren ist besonders nützlich, wenn Anforderungen in Form von Ablaufdiagrammen oder Tabellen von Geschäftsregeln vorliegen. Entscheidungstabellen können auch zur Spezifikation von Anforderungen verwendet werden; daher liegen die Anforderungsspezifikationen manchmal bereits in dieser Form vor. Auch wenn die Anforderungen nicht als Tabelle oder Ablaufdiagramm vorliegen, können die Kombinationen von Bedingungen meist aus dem Text abgeleitet werden. Beim Entwurf der Entscheidungstabellen müssen nicht nur die spezifizierten Kombinationen von Bedingungen abgedeckt werden, sondern auch jene, die nicht ausdrücklich spezifiziert (aber trotzdem vorhanden) sind. Nur wenn wirklich alle möglichen Kombinationen berücksichtigt werden, sind Entscheidungstabellen ein gutes Testentwurfswerkzeug.

Einschränkungen/Schwierigkeiten
Das Identifizieren sämtlicher Kombinationen von Bedingungen kann eine Herausforderung darstellen, insbesondere dann, wenn die Anforderungen an das System nicht gut spezifiziert sind oder wenn keine existieren. Es ist nicht ungewöhnlich, dass eine Menge von Bedingungen erstellt wird, nur um dann festzustellen, dass das erwartete Ergebnis nicht bekannt ist.

Überdeckung
Die minimale Testüberdeckung bei einer Entscheidungstabelle ist ein Test pro Spalte. Dies setzt voraus, dass es keine Mehrfachbedingungen gibt, und dass in der Spalte alle möglichen Kombinationen von Bedingungen berücksichtigt wurden. Bei der Auswahl der Tests aus einer Entscheidungstabelle sollten auch etwaige Grenzbedingungen berücksichtigt werden, die getestet werden sollten. Hierdurch kann sich die Anzahl der Testfälle erhöhen, die zum hinreichenden Testen der Software nötig ist. Grenzwertanalyse und Äquivalenzklassenbildung sind Testverfahren, die den Entscheidungstabellentest ergänzen.

Fehlerarten
Zu den typischen Fehlerzuständen, die mit diesem Verfahren aufgedeckt werden, gehören die inkorrekte Verarbeitung, die auf bestimmten Kombinationen von Bedingungen basiert und zu unerwarteten Ergebnissen führt. Beim Erstellen der Entscheidungstabellen können Fehlerzustände in den Spezifikationen aufgedeckt werden. Am häufigsten handelt es sich um Lücken (z.B. wenn keine Informationen vorliegen, die beschreiben, was in einer bestimmten Situation passieren soll) oder um Widersprüche. Beim Testen können auch Probleme mit Kombinationen von Bedingungen aufgedeckt werden, die gar nicht oder nicht gut gehandhabt werden.

3.2.4 Ursache-Wirkungs-Graph-Analyse

Ursache-Wirkungs-Graphen können aus jeder Quelle abgeleitet werden, in der die Funktionslogik (d.h. die „Regeln“) eines Programms beschrieben ist, wie z.B. in User Stories oder Ablaufdiagrammen. Sie geben einen graphischen Überblick über die logische Struktur eines Programms und dienen meist als Basis für die Erstellung der Entscheidungstabellen. Durch die Ermittlung der Entscheidungen in Form eines Ursache-Wirkungs-Graphs und/oder in Form von Entscheidungstabellen lässt sich eine systematische Testüberdeckung der Logik des zu testenden Programms erzielen.

Anwendbarkeit
Die Ursache-Wirkungs-Analyse gilt in den gleichen Situationen wie Entscheidungstabellen, und sie wird in den gleichen Teststufen eingesetzt. Insbesondere zeigt ein Ursache-Wirkungs-Graph Kombinationen von auslösenden Bedingungen, die zu Ergebnissen führen (Kausalität); Kombinationen von Bedingungen, die Ergebnisse ausschließen („nicht“); Mehrfachbedingungen, die „wahr“ sein müssen, um zu einem Ergebnis zu führen („und“), sowie alternative Bedingungen, die „wahr“ sein können, um zu einem bestimmten Ergebnis zu führen („oder“). Es ist leichter, diese Beziehungen aus einem Ursache-Wirkungs-Graph abzulesen als aus einer Beschreibung in Textform.

Einschränkungen/Schwierigkeiten
Verglichen mit anderen Testentwurfsverfahren ist für das Erlernen der Ursache-Wirkungs-Analyse zusätzlich Zeit und Aufwand erforderlich. Außerdem ist Werkzeugunterstützung notwendig. Ursache-Wirkungs-Graphen haben eine spezifische Notation, die vom Ersteller und vom Leser der Graphen verstanden werden muss.

Überdeckung
Um eine Mindestüberdeckung zu erzielen, muss jede mögliche Ursache, einschließlich der Kombinationen von auslösenden Bedingungen, die mit einer Wirkung in Verbindung gesetzt ist, getestet werden. Im Ursache-Wirkungs-Graph können auch Beschränkungen in Zusammenhang mit den Daten und in Zusammenhang mit der Ablauflogik spezifiziert werden.

Fehlerarten
Mit Hilfe der Ursache-Wirkungs-Analyse werden dieselben Fehlerarten gefunden wie mit Entscheidungstabellen. Des Weiteren kann durch die Erstellung des Ursache-Wirkungs-Graphen der geforderte Detaillierungsgrad in der Testbasis erreicht werden. Dadurch werden Granularität und Qualität der Testbasis verbessert. Dies hilft den Testern bei der Identifizierung fehlender Anforderungen.

3.2.5 Zustandsbasierter Test

Der zustandsbasierte Test wird eingesetzt, um die Software in Bezug auf ihre definierten Zustände und die Übergänge zwischen diesen Zuständen zu testen; die Zustandsübergänge können gültig oder ungültig sein. Ereignisse sind die Auslöser für die Software, von einem Zustand in einen anderen zu wechseln und Aktionen auszuführen. Ereignisse sind von Bedingungen beeinflusst (diese werden auch als Schutzbedingungen oder Übergangsschutz bezeichnet), was sich wiederum auf den Zustandsübergangspfad auswirkt. Beispiel: Das Einloggen mit einer gültigen Kombination aus Benutzernamen und Passwort resultiert in einem anderen Zustandsübergang als das Einloggen mit einem ungültigen Passwort.

Zustandsübergänge werden entweder in einem Zustandsdiagramm festgehalten, das alle gültigen Zustandsübergänge grafisch darstellt, oder in einer Zustandstabelle, die alle möglichen Zustandsübergänge (sowohl gültige als auch ungültige) auflistet.

Anwendbarkeit
Der zustandsbasierte Test kann für jede Software eingesetzt werden, die definierte Zustände hat, und bei der die Übergänge zwischen diesen Zuständen durch Ereignisse ausgelöst werden (z.B. Wechsel des Bildschirms). Der zustandsbasierte Test kann in jeder Teststufe eingesetzt werden. Eingebettete Software, browserbasierte Software, sowie jede Art von Transaktionssystem bieten sich für dieses Testverfahren an. Außerdem ist der zustandsbasierte Test gut geeignet für Steuerungssysteme (z.B. Systeme zur Steuerung von Verkehrsampeln).

Einschränkungen/Schwierigkeiten
Die schwierigste Aufgabe beim Erstellen von Zustandsdiagrammen oder –übergangstabellen ist die Bestimmung der Zustände. Wenn Software eine Benutzerschnittstelle hat, dann werden häufig die einzelnen Bildschirmmasken, die für den Benutzer angezeigt werden, zur Definition der verschiedenen Zustände verwendet. Bei eingebetteter Software hängen die Zustände oft von den Zuständen der Hardware ab.

Abgesehen von den Zuständen ist der einzelne Zustandsübergang (auch als 0-Switch bezeichnet) die Basisgröße beim zustandsbasierten Test. Wenn einfach nur alle Zustandsübergänge getestet werden, dann werden sicherlich einige Defekte bei den Zustandsübergängen gefunden; es können jedoch mehr gefunden werden, wenn Folgen von Transaktionen getestet werden. Eine Sequenz mit zwei aufeinander folgenden Übergängen wird 1-Switch bezeichnet; eine Sequenz mit drei aufeinander

folgenden Übergangen wird 2-Switch bezeichnet; usw. Diese “Switches” werden manchmal auch als N-1-Switches bezeichnet, wobei N für die Anzahl der zu durchlaufenden Zustandsübergänge steht. In diesem Fall wird ein einziger Übergang (ein 0-Switch) als 1-1-Switch bezeichnet. [Bath08]

Überdeckung
Wie bei den anderen Testverfahren gibt es auch beim zustandsbasierten Test eine Hierarchie hinsichtlich der Überdeckungsgrade. Die minimale Überdeckung, die akzeptabel ist, ist wenn jeder Zustand und jeder Zustandsübergang während des Tests einmal ausgeführt wurde. 100% Zustandsübergangsüberdeckung (auch bekannt als 100% 0-Switch-Überdeckung oder 100% logische Zweigüberdeckung) garantiert, dass jeder Zustand getestet wurde, es sei denn der Systementwurf oder das Zustandsübergangsmodell (Zustandsdiagramm oder -tabelle) ist fehlerhaft. Je nach den Beziehungen zwischen den Zuständen und den Zustandsübergängen kann es erforderlich sein, dass manche Zustandsübergänge mehr als einmal ausgeführt werden müssen, damit andere Zustandsübergänge wenigstens einmal ausgeführt werden können.

Der Begriff "N-Switch-Überdeckung" bezieht sich auf Sequenzen, die mehr als einen einzigen Zustandsübergang enthalten. Beispiel: Für 100% 1-Switch-Überdeckung muss jede gültige Sequenz von zwei aufeinander folgenden Zustandsübergängen mindestens einmal getestet werden. Dieses Testen kann einige Fehlerwirkungen hervorrufen, die mit 100% 0-Switch-Überdeckung unentdeckt geblieben wären.

"Rundreise-Überdeckung" betrifft Situationen, in denen Folgen von Zustandsübergängen Schleifen formen. 100% "Rundreise-Überdeckung" ist erzielt, wenn alle Schleifen ab einem beliebigen Zustand und zurück zu diesem Zustand getestet wurden. Das muss für alle Zustände getestet werden, die zur Schleife gehören.

"Vollständige Überdeckung" ist erreicht, wenn alle möglichen Sequenzen mit einer Länge von n-1 getestet wurden (wobei n die Anzahl der Zustände im Testobjekt ist).

Bei all diesen Vorgehensweisen kann eine noch höhere Überdeckung erzielt werden, wenn man auch versucht alle ungültigen Zustandsübergänge einzubeziehen. Aus den Überdeckungsanforderungen und abzudeckenden Zustandsübergängen muss hervorgehen, ob ungültige Zustandsübergänge beinhaltet sind.

Fehlerarten
Zu den typischen Defekten gehören inkorrekte Verarbeitungen im aktuellen Zustand als Folge der Verarbeitung in einem vorangegangenen Zustand, falsche oder nicht unterstützte Zustands-übergänge, Zustände ohne Ausgang, sowie ein Bedarf an Zuständen oder Zustandsübergängen, die nicht existieren. Bei der Erstellung des Zustandsmodells können Defekte im Spezifikationsdokument aufgedeckt werden. Am häufigsten handelt es sich um Lücken (z.B. es gibt keine Informationen darüber, was in einer bestimmten Situation passieren soll) oder um Widersprüche.

3.2.6 Kombinatorische Testverfahren

Kombinatorisches Testen wird eingesetzt, wenn Software mit mehreren Parametern getestet wird, von denen jeder mehrere Werte hat, was in mehr Kombinationen resultiert als in der verfügbaren Zeit getestet werden können. Die Parameter müssen unabhängig und insoweit kompatibel sein, dass jede Option für jeden beliebigen Faktor mit jeder Option eines beliebigen anderen Faktors kombiniert werden kann. Bei Klassifikationsbäumen ist es zulässig, einige Kombinationen auszuschließen, falls bestimmte Optionen nicht kompatibel sind. Daraus kann jedoch nicht abgeleitet werden, dass sich die miteinander kombinierten Faktoren nicht gegenseitig beeinflussen; das ist durchaus möglich, allerdings sollte dies in einer akzeptablen Art und Weise erfolgen.

Durch kombinatorisches Testen lässt sich eine geeignete Teilmenge dieser Kombinationen identifizieren, um ein vorgegebenes Überdeckungsmaß. zu erzielen. Die Anzahl der Elemente für die

Kombinationen wird vom Test Analyst bestimmt und ausgewählt. Sie umfasst einzelne Elemente, Paare von Elementen, oder drei oder mehr Elemente [Copeland03]. Es gibt verschiedene Werkzeuge, die den Test Analyst bei dieser Aufgabe unterstützen (siehe www.pairwise.org für Beispiele). Beim Einsatz dieser Werkzeuge müssen entweder die Parameter und deren Werte aufgelistet werden (bei paarweisem Testen und bei orthogonalen Arrays oder sie müssen grafisch dargestellt werden (Klassifikationsbäume) [Grochtmann94]. Paarweises Testen ist ein Verfahren, bei dem Paare von Variablen in Kombination getestet werden. Bei orthogonalen Arrays handelt es sich um vordefinierte, mathematisch genaue Tabellen, die es dem Test Analysten ermöglichen, die zu testenden Elemente durch Variable aus dem Array zu ersetzen; dies resultiert in einer Menge von Kombinationen, die ein Überdeckungsmaß erzielen, wenn sie getestet werden [Koomen06]. Klassifikationsbaumwerkzeuge ermöglichen es dem Test Analyst, die Größe der zu testenden Kombinationen zu definieren (d.h. Kombinationen von zwei Werten, von drei Werten, usw).

Anwendbarkeit
Das Problem mit zu vielen Kombinationen von Parameterwerten wird in mindestens zwei unterschiedlichen Situationen beim Testen deutlich. Einige Testfälle enthalten mehrere Parameter, jeweils mit mehreren möglichen Werten, beispielsweise eine Bildschirmmaske mit mehreren Eingabefeldern. In diesem Fall sind die Kombinationen von Parameterwerten die Eingabedaten für die Testfälle. Des Weiteren können einige Systeme in einer Reihe von Dimensionen konfigurierbar sein, was zu einem potenziell großen Konfigurationsraum führt. In beiden Fällen kann kombinatorisches Testen eingesetzt werden, um eine Teilmenge von Kombinationen in akzeptablem Umfang zu identifizieren.

Für Parameter mit sehr vielen Werten kann Äquivalenzklassenbildung oder irgendein anderer Auswahlmechanismus zunächst auf jeden Parameter einzeln angewendet werden, um die Anzahl der Werte für die einzelnen Parameter zu reduzieren. Das kombinatorische Testen wird dann erst im Anschluss daran eingesetzt, um die Menge der resultierenden Kombinationen zu reduzieren.

Dieses Verfahren wird meist beim Integrations-, System- und Systemintegrationstest angewendet.

Einschränkungen/Schwierigkeiten
Die größte Einschränkung bei diesen Verfahren ist die Annahme, dass die Ergebnisse einiger weniger Tests repräsentativ sind für alle Tests, und dass diese wenigen Tests die erwartete Nutzung abbilden. Falls es eine unerwartete Interaktion zwischen bestimmten Variablen gibt, wird diese mit dieser Testart möglicherweise nicht entdeckt, wenn diese bestimmte Kombination nicht getestet wird. Diese Verfahren können Personen ohne technischen Hintergrund nur schwer erklärt werden, da sie die Logik bei der Reduzierung der Tests möglicherweise nicht verstehen.

Die Identifizierung der Parameter und der jeweiligen Werte gestaltet sich manchmal schwierig. Auch ist es schwierig, die minimale Anzahl von Kombinationen für ein bestimmtes Überdeckungsmaß manuell zu bestimmen. Zur Bestimmung der minimalen Menge von Kombinationen werden meist Werkzeuge eingesetzt. Einige Werkzeuge bieten die Möglichkeit, einige (Teil-) Kombinationen bei der endgültigen Auswahl der Kombinationen zwingend einzuschließen oder auszuschließen. Diese Fähigkeit der Werkzeuge kann der Test Analyst nutzen, um basierend auf der Kenntnis des Geschäftsbereichs oder den Informationen über die Produktverwendung einzelne Faktoren mehr oder weniger zu gewichten.

Überdeckung
Es gibt mehrere Überdeckungsmaße. Das niedrigste ist die sogenannte 1-fache oder Singleton-Überdeckung, bei der jeder Wert eines jeden Parameters in mindestens einer der ausgewählten Kombinationen vorhanden sein muss. Das nächste Überdeckungsmaß wird als 2-fache oder paarweise Überdeckung bezeichnet. Dabei muss jedes Wertepaar von zwei beliebigen Parametern in mindestens einer Kombination vorhanden sein. Diese Idee lässt sich verallgemeinern zur n-fachen Überdeckung, bei der jede Teilkombination von Werten einer jeden Menge von „n“ Parametern in der

Menge der ausgewählten Kombinationen enthalten sein muss. Je höher „n“, desto mehr Kombinationen sind erforderlich, um 100% Überdeckung zu erzielen. Die Mindestüberdeckung bei diesen Verfahren ist ein Testfall pro Kombination, die vom Werkzeug erstellt wird.

Fehlerarten
Die häufigsten Fehlerzustände, die mit diesen Testverfahren aufgedeckt werden, sind Fehlerzustände in Zusammenhang mit den kombinierten Werten mehrerer Parameter.

3.2.7 Anwendungsfallbasierter Test

Der anwendungsfallbasierte Test liefert transaktionale, auf Szenarien basierende Tests, die die Benutzung des Systems nachahmen. Ein Anwendungsfall beschreibt die Interaktionen zwischen den Aktoren und dem System, die ein Ergebnis erzielen. Aktoren können Benutzer oder externe Systeme sein.

Anwendbarkeit
Der anwendungsfallbasierte Test wird meist beim System- und Abnahmetest angewendet. Je nach Integrationsebene kann er in der Integrationsteststufe eingesetzt werden oder, je nach Verhalten der zu testenden Komponente, sogar auch beim Komponententest. Anwendungsfälle dienen häufig auch als Basis für den Performanztest, da sie die Anwendung des Systems realistisch abbilden. Die in den Anwendungsfällen beschriebenen Szenarien können virtuellen Benutzern zugewiesen werden, um eine realistische Systemlast zu erzeugen.

Einschränkungen/Schwierigkeiten
Anwendungsfälle müssen, um gültig zu sein, realistische Benutzertransaktionen abbilden. Die entsprechenden Informationen stammen von einem Benutzer oder von einem Benutzervertreter. Der Wert eines Anwendungsfalles verringert sich, wenn dieser die Aktivitäten der echten Benutzer nicht genau wiedergibt. Eine genaue Spezifikation der verschiedenen alternativen Pfade (Abläufe) ist für eine gründliche Testüberdeckung unerlässlich. Anwendungsfälle sollten als Richtlinien aufgefasst werden, und nicht als vollständige Definition dessen, was zu testen ist; sie liefern nämlich unter Umständen keine eindeutige Definition der gesamten Menge von Anforderungen. Es empfiehlt sich möglicherweise, aus der Schilderung des Anwendungsfalls auch andere Modelle zu erstellen, wie z.B. Ablaufdiagramme. Dadurch lässt sich die Genauigkeit des Testens verbessern und der Anwendungsfall verifizieren.

Überdeckung
Zur Mindestabdeckung eines Anwendungsfall ist ein Testfall für den (positiven) Hauptpfad erforderlich, und ein Testfall für jeden alternativen Pfad oder Prozessablauf. Bei den alternativen Pfaden kann es sich auch um Ausnahme- oder Ausfallpfade handeln. Alternative Pfade sind manchmal als Verlängerung des Hauptpfades dargestellt. Die prozentuale Überdeckung wird bestimmt, indem man die Anzahl der getesteten Pfade durch die Gesamtzahl der Haupt- und Alternativpfade teilt.

Fehlerarten
Zu den mit diesem Testverfahren aufgedeckten Fehlerzuständen gehören die fehlerhafte Verarbeitung spezifizierter Szenarien, die versäumte Verarbeitung alternativer Pfade, die inkorrekte Verarbeitung der vorliegenden Bedingungen, sowie das umständliche oder inkorrekte Berichten von Fehlern.

3.2.8 User-Story-basiertes Testen

Bei manchen agilen Methodologien, wie beispielsweise Scrum, werden die Anforderungen in Form von User Stories erstellt, die kleine Funktionseinheiten beschreiben und in einer einzigen Iteration entworfen, entwickelt, getestet und vorgeführt werden können [Cohn04]. In diesen User Stories ist eine Beschreibung der zu implementierenden Funktionalität enthalten, alle nicht-funktionale Kriterien, sowie die Abnahmekriterien, die erfüllt sein müssen, damit die User Story als vollständig betrachtet werden kann.

Anwendbarkeit
User Stories werden überwiegend im agilen, sowie in ähnlich ausgeprägtem, iterativem und inkrementellem Umfeld eingesetzt. Sie werden sowohl für funktionales als auch für nicht-funktionales Testen verwendet. User Stories werden in allen Teststufen eingesetzt. Es wird erwartet, dass der Entwickler die für die User Story implementierte Funktionalität demonstriert, bevor der Code an das Team zur Durchführung der nächsten Stufe der Testaufgaben (z.B. Integrationstest, Performanztest) übergeben wird.

Einschränkungen/Schwierigkeiten
Da es sich bei User Stories um kleine Inkremente der Funktionalität handelt, ist es manchmal notwendig, Treiber und Platzhalter zu erstellen, damit die gelieferte Teilfunktionalität überhaupt getestet werden kann. Hierfür sind Programmierfähigkeiten und die Benutzung von Werkzeugen notwendig, die das Testen unterstützen wie z.B. API-Testwerkzeuge. Die Erstellung der Treiber und Platzhalter ist meist Aufgabe des Entwicklers, obwohl auch ein Technical Test Analyst bei der Erstellung des Codes und der Benutzung des API-Testwerkzeugs involviert sein kann. Falls, wie bei den meisten agilen Projekten, eine kontinuierliche Integration erfolgt, wird der Bedarf von Treibern und Platzhaltern minimiert.

Überdeckung
Die Mindestüberdeckung einer User Story ist dann gegeben, wenn alle spezifizierten Abnahmekriterien erfüllt wurden.

Fehlerarten
Es werden in der Regel funktionale Fehlerzustände aufgedeckt, bei denen die Software die spezifizierte Funktionalität nicht liefert. Außerdem werden Fehlerzustände in Zusammenhang mit der Integration der Funktionalität der neuen User Story mit der bereits vorhandenen Funktionalität gefunden. Da User Stories unabhängig entwickelt werden können, können Probleme mit Performanz, Schnittstellen und Fehlerbehandlungen gefunden werden. Es ist wichtig, dass der Test Analyst nicht nur die einzelnen gelieferten Funktionalitäten testet, sondern auch jedes Mal, wenn eine neue User Story für den Test freigegeben wird, einen Integrationstest durchführt.

3.2.9 Wertebereichsanalyse

Eine Domain oder ein Wertebereich ist eine definierte Menge von Werten. Dabei kann es sich um einen Wertebereich einer einzelnen Variablen handeln (ein eindimensionaler Wertebereich, z.B. “Männer über 24 und unter 66 Jahren”), oder um Wertebereiche interagierender Variablen (ein multidimensionaler Wertebereich, z.B. “Männer über 24 und unter 66 Jahren UND mit einem Gewicht über 69 kg und unter 90kg”). Jeder Testfall für einen multi-dimensionalen Wertebereich muss geeignete Werte für jede der betroffenen Variablen enthalten.

Für die Wertebereichsanalyse eines eindimensionalen Wertebereichs werden normalerweise Äquivalenzklassenbildung und Grenzwertanalyse verwendet. Wenn die Äquivalenzklassen definiert sind, wählt der Test Analyst Werte für jede der Äquivalenzklassen, und zwar einen Wert, der innerhalb des Wertebereichs (IN) ist, und einen, der außerhalb ist (OUT), sowie einen Wert auf der Grenze (ON) und einen knapp neben der Grenze (OFF). Durch die Bestimmung dieser Werte wird jeder Teilbereich sowie seine jeweiligen Grenzbedingungen getestet. [Black07]

Bei multidimensionalen Wertebereichen nimmt die Anzahl der mit diesen Verfahren erstellten Testfälle exponentiell mit der Anzahl der betroffenen Variablen zu. Eine Vorgehensweise, die auf der Domain - Theorie basiert, führt dagegen zu einem linearen Zuwachs. Da die formale Vorgehensweise eine Theorie der Fehlerzustände (ein Fehlermodell) beinhaltet, das bei Äquivalenzklassenbildung und Grenzwertanalyse fehlt, werden mit einer kleineren Testmenge Fehlerzustände in multidimensionalen Wertebereichen aufgedeckt, die bei der größeren, heuristischen Testmenge wahrscheinlich

unentdeckt bleiben würden. Beim Testen multidimensionaler Wertebereiche wird das Testmodell in Form einer Entscheidungstabelle (oder Wertebereichs-Matrix) erstellt. Zur Identifizierung von Testfalldaten für multidimensionale Wertebereiche, die mehr als dreidimensional sind, ist wahrscheinlich Computerunterstützung nötig.

Anwendbarkeit
Die Wertebereichsanalyse kombiniert die Testverfahren Entscheidungstabellen, Äquivalenzklassenbildung und Grenzwertanalyse, um mit einer kleineren Testmenge die Bereiche abzudecken, die wichtig sind und in denen Fehlerwirkungen wahrscheinlich sind. Die Wertebereichsanalyse wird oft bei sehr vielen potenziell miteinander interagierenden Variablen angewendet, wenn Entscheidungstabellen nicht handhabbar wären. Die Wertebereichsanalyse kann in jeder beliebigen Teststufe eingesetzt werden, wird aber am häufigsten in den Integrations- und Systemteststufen eingesetzt.

Einschränkungen/Schwierigkeiten
Für eine gründliche Wertebereichsanalyse ist ein gutes Verständnis der Software unerlässlich, damit die verschiedenen Wertebereiche und die möglichen Interaktionen zwischen den Wertebereichen identifiziert werden können. Wird ein Wertebereich übersehen und nicht identifiziert, dann kann das Testen unzureichend sein. Es ist jedoch wahrscheinlich, dass der Wertebereich entdeckt wird, da die OFF- und OUT-Variablen möglicherweise in den unentdeckten Bereich fallen. Die Wertebereichsanalyse ist ein starkes Verfahren, besonders wenn Tester mit Entwicklern zusammenarbeiten, um die Testbereiche festzulegen.

Überdeckung
Die Mindestüberdeckung bei der Wertebereichsanalyse ist gegeben, wenn in jedem Wertebereich jeder IN-, OUT-, ON- und OFF-Wert im Test abgedeckt wurde. Wenn sich Werte überschneiden (z.B. wenn der OUT-Wert eines Wertebereichs identisch ist mit dem IN-Wert eines anderen Wertebereichs), müssen nicht zwei Tests durchgeführt werden. In der Praxis sind daher oft weniger als vier Tests pro Wertebereich erforderlich.

Fehlerarten
Zu den gefundenen Fehlerzuständen gehören funktionale Probleme innerhalb des Wertebereichs, Behandlung von Grenzwerten, Probleme bei der Interaktion von Variablen und bei der Fehlerbehandlung (insbesondere für Werte, die nicht zu einem gültigen Wertebereich gehören).

3.2.10 Kombination von Verfahren

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. Bei der Auswahl des geeigneten Testverfahrens sollte der Test Analyst die Anwendbarkeit des Verfahrens, die Einschränkungen und Schwierigkeiten, sowie die Testziele bezüglich der Überdeckung und der aufdeckbaren Fehlerzustände berücksichtigen. Möglicherweise gibt es für eine Situation nicht ein einziges “bestes” Verfahren. Oft liefert eine Kombination verschiedener Verfahren die größtmögliche Überdeckung, vorausgesetzt dass für die korrekte Anwendung der Verfahren ausreichend Zeit und fachliche Kompetenz vorhanden sind.

3.3 Fehlerbasierte Verfahren

3.3.1 Verwendung von fehlerbasierten Verfahren

Bei fehlerbasierten Testentwurfsverfahren dient die Art des gesuchten Fehlers als Basis für den Testentwurf, wobei die Tests systematisch davon abgeleitet werden, was über den Fehlerart bekannt ist. Anders als beim spezifikationsorientierten Test, bei dem Tests aus der Spezifikation abgeleitet werden, werden beim fehlerbasierten Test die Tests aus Fehlertaxonomien (d.h. aus kategorisierten Listen) abgeleitet, die völlig unabhängig von der getesteten Software sein können. Die Taxonomien können Listen möglicher Fehlerarten, Grundursachen, Symptome von Fehlerwirkungen und sonstige fehlerbezogenen Daten enthalten. Beim fehlerbasierten Testen können auch Listen mit den identifizierten Risiken und Risikoszenarien als Grundlage für zielgerichtetes Testen verwendet werden. Dieses Verfahren ermöglicht es dem Tester, auf besondere Arten von Fehlern abzuzielen, oder eine Fehlertaxonomie mit bekannten und häufigen Fehlerzuständen bestimmter Arten systematisch durchzuarbeiten. Der Test Analyst nutzt die Taxonomiedaten, um das Testziel zu bestimmen, d.h. eine bestimmte Fehlerart zu finden. Basierend auf diesen Informationen erstellt der Test Analyst Testfälle und Testbedingungen, die den Fehlerzustand aufdecken werden, sofern vorhanden.

Anwendbarkeit
Fehlerbasiertes Testen kann in jeder beliebigen Teststufe eingesetzt werden, wird aber am häufigsten in der Systemteststufe eingesetzt. Es gibt Standardtaxonomien, die für mehrere Arten von Software passen. Diese Art des Testens, die nicht produktspezifisch ist, unterstützt auch branchenspezifisches Standardwissen, um bestimmte Tests abzuleiten. Wenn branchenspezifische Taxonomien verwendet werden, dann können Metriken zum Fehlerauftreten projektübergreifend und sogar organisationsübergreifend verfolgt werden.

Einschränkungen/Schwierigkeiten
Es gibt verschiedene Fehlertaxonomien, mit spezifischen Schwerpunkt auf jeweils bestimmten Testarten z.B. Benutzbarkeitstest. Es ist wichtig, eine solche Taxonomie zu finden und auszuswählen, die für die zu testende Software anwendbar ist. Für innovative Software gibt es möglicherweise gar keine Taxonomien. Manche Unternehmen haben eigene Taxonomien mit wahrscheinlichen oder häufig auftretenden Fehlern erstellt. Unabhängig davon, welche Taxonomie verwendet wird, muss vor Testbeginn auf jeden Fall die erwartete Überdeckung festgelegt werden.

Überdeckung
Das Verfahren liefert auch Kriterien für den Überdeckungsgrad, der bestimmt, wann alle sinnvollen Tests identifiziert wurden. In der Praxis sind die Kriterien für Überdeckungsgrade bei fehlerbasierten Testentwurfsverfahren weniger systematisch als bei spezifikationsorientierten Verfahren. Nur allgemeine Regeln zur Überdeckung werden vorgegeben, es liegt aber im eigenen Ermessen, wo genau die Grenze einer sinnvollen Überdeckung im Testentwurf liegt. Wie bei anderen Verfahren bedeutet das Kriterium für den Überdeckungsgrad nicht, dass die Menge von Tests vollständig ist. Es gibt vielmehr an, dass dieses Testverfahren für durch die betrachteten Fehlerarten keine weiteren sinnvollen Tests mehr vorschlägt.

Fehlerarten
Die Fehlerarten, die gefunden werden, hängen gewöhnlich von der benutzten Taxonomie ab. Wenn eine Benutzerschnittstellen-Taxonomie verwendet wird, wird der Großteil der entdeckten Fehler Benutzerschnittstellenspezifisch sein, andere Fehler können jedoch als Nebenprodukt der spezifischen Tests gefunden werden.

3.3.2 Fehlertaxonomien

Fehlertaxonomien sind kategorisierte Listen von Fehlerarten. Diese Listen können sehr allgemein gehalten sein (als abstrakte Richtlinien), oder sie können sehr spezifisch sein. Beispiel: Eine

Taxonomie für mögliche Fehler von Benutzerschnittstellen könnte allgemeine Punkte beinhalten, wie z.B. Funktionalität, Fehlerbehandlung, grafische Darstellung und Performanz. Eine detaillierte Taxonomie könnte eine Liste aller möglichen Objekte von Benutzerschnittstellen enthalten (insbesondere bei grafischen Benutzeroberflächen) und die unsachgemäße Behandlung dieser Objekte benennen, zum Beispiel für:

- Textfelder
  - Gültige Daten werden nicht übernommen
  - Ungültige Daten werden übernommen
  - Länge der Eingabe wird nicht verifiziert
  - Sonderzeichen werden nicht erkannt
  - Fehlermeldungen sind nicht informativ
  - Benutzer kann fehlerhafte Daten nicht korrigieren
  - Regeln werden nicht angewendet
- Datumsfeld
  - Gültige Datumsangaben werden nicht übernommen
  - Ungültige Datumsangaben werden nicht abgewiesen
  - Datumsbereiche werden nicht verifiziert
  - Präzisionsdaten werden nicht korrekt behandelt (zum Beispiel hh:mm:ss)
  - Benutzer kann fehlerhafte Daten nicht korrigieren
  - Regeln werden nicht angewendet (z.B. Enddatum muss größer sein als Startdatum)

Es gibt viele Fehlertaxonomien, von formalen Taxonomien, die gekauft werden können, bis hin zu Taxonomien, die von verschiedenen Unternehmen für bestimmte Zwecke entworfen wurden. Intern entwickelte Fehlertaxonomien können auch eingesetzt werden, um spezifische Fehlerzustände aufzudecken, die häufig innerhalb des Unternehmens gefunden werden. Bevor man eine neue Fehlertaxonomie erstellt oder eine vorhandene individuell anpasst, müssen die Ziele oder Zielsetzungen der Taxonomie festgelegt werden. Beispiel: Mögliche Ziele könnten sein, Probleme mit Benutzerschnittstellen zu finden, die in Produktivsystemen entdeckt wurden, oder Probleme mit der Behandlung von Eingabefeldern zu identifizieren.

Die Schritte zur Erstellung einer Taxonomie sind:

1. Ein Ziel festsetzen und den gewünschten Detaillierungsgrad definieren
2. Eine vorhandene Taxonomie auswählen, die als Basis dienen kann
3. Werte und häufige Fehlerzustände spezifizieren, die im Unternehmen und/oder außerhalb in der Praxis vorgekommen sind.

Je detaillierter eine Taxonomie ist, desto höher der Zeitaufwand für Entwicklung und Wartung der Taxonomie. Ein hoher Detaillierungsgrad resultiert jedoch in einer höheren Reproduzierbarkeit der Testergebnisse. Detaillierte Taxonomien können redundant sein; sie ermöglichen dem Testteam jedoch die Aufteilung des Testens, ohne Verlust an Informationen oder Überdeckung.

Nachdem eine geeignete Taxonomie ausgewählt wurde, kann sie für den Entwurf von Testbedingungen und Testfällen verwendet werden. Eine risikoorientierte Taxonomie kann die Aufmerksamkeit beim Testen auf einen bestimmten Risikobereich lenken. Taxonomien können auch für nicht-funktionale Bereiche, wie z.B. Benutzbarkeit, Performanz, usw., verwendet werden. Taxonomie-Listen sind in verschiedenen Veröffentlichungen erhältlich, vom IEEE oder im Internet.

3.4 Erfahrungsbasierte Verfahren

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 Testabläufe zu produzieren. Wenn die Systemdokumentation schlecht ist, wenn die verfügbare Zeit zum Testen extrem knapp ist, oder wenn das Testteam ein ausgeprägtes Fachwissen über das System hat, dann kann erfahrungsbasiertes Testen eine gute Alternative zu strukturierten Vorgehensweisen sein. Erfahrungsbasiertes Testen ist weniger geeignet, wenn Systeme eine detaillierte Testdokumentation oder sehr gute Wiederholbarkeit benötigen, oder wenn der Testüberdeckungsgrad genau bewertbar sein muss.

Beim Einsatz dynamischer oder heuristischer Ansätze verwenden Tester normalerweise erfahrungsbasierte Tests. Das Testen reagiert dann eher auf Ereignisse als eine geplante Vorgehensweise zu befolgen. 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.

Auch wenn für die behandelten Testverfahren einige Aspekte der Überdeckung erwähnt werden, ist zu beachten, dass erfahrungsbasierte Verfahren keine formalen Kriterien für Überdeckungsgrade liefern.

3.4.1 Intuitive Testfallermittlung (Error Guessing)

Bei der intuitiven Testfallermittlung nutzen Test Analysten ihre Erfahrung, um Fehler zu erraten, die vielleicht gemacht wurden, als der Code entworfen und entwickelt wurde. Wenn die erwarteten Fehlhandlungen identifiziert wurden, bestimmt der Test Analyst die Methoden zur Aufdeckung der daraus resultierenden Fehlerzustände. Beispiel: Wenn der Test Analyst erwartet, dass die Software Fehlerwirkungen zeigen wird, wenn ein ungültiges Passwort eingegeben wird, dann werden Tests zur Eingabe vieler unterschiedlicher Werte im Passwortfeld entworfen, um zu verifizieren, dass der Fehler tatsächlich gemacht wurde zu dem Fehlerzustand geführt hat, und als Fehlerwirkung in Erscheinung tritt, wenn die Tests ausgeführt werden.

Außer als Testverfahren ist das Error Guessing auch bei der Risikoanalyse nützlich, um potenzielle Fehlerauswirkungen zu identifizieren. [Myers97]

Anwendbarkeit
Intuitive Testfallermittlung wird überwiegend beim Integrations- und Systemtest eingesetzt, kann aber grundsätzlich in jeder Teststufe angewendet werden. Dieses Verfahren wird häufig zusammen mit anderen Testverfahren eingesetzt, um die Bandbreite der bestehenden Testfälle zu erweitern. Die intuitive Testfallermittlung kann auch effektiv beim Testen neuer Softwarereleases eingesetzt werden, um die Software auf häufige Fehler und Fehlhandlungen zu testen, noch bevor gründlicheres und skriptbasiertes Testen erfolgt. Checklisten und Taxonomien können hilfreich sein, um das Testen zu unterstützen.

Einschränkungen/Schwierigkeiten
Die Überdeckung ist schwierig zu bestimmen und variiert stark je nach Fachkompetenz und Erfahrung des Test Analysten. Es ist am besten, wenn das Verfahren von einem erfahrenen Tester eingesetzt wird, der sich gut auskennt mit den Fehlerarten, die häufig bei der Art des zu testenden Codes vorkommen. Die intuitive Testfallermittlung wird häufig eingesetzt, wird aber selten dokumentiert, und sie ist daher wohl weniger reproduzierbar als andere Arten des Testens.

Überdeckung
Bei Verwendung einer Taxonomie wird die Überdeckung durch die zutreffenden Datenfehler und Fehlerarten bestimmt. Ohne Taxonomie ist die Überdeckung durch Erfahrung und Wissen der Tester sowie die verfügbare Zeit beschränkt. Der Erfolg bei diesem Verfahren hängt stark davon ab, wie gut der Tester die Problembereiche treffen kann.

Fehlerarten
Die Fehlerarten, die mit diesem Verfahren aufgedeckt werden, sind normalerweise die Fehlerzustände, die durch die verwendete Taxonomie vorgegeben sind, oder Fehlerzustände, die der Test Analyst intuitiv „errät“, und die durch spezifikationsorientierte Testverfahren vielleicht nicht gefunden worden wären.

3.4.2 Checklisten-basiertes Testen

Beim Checklisten-basierten Testverfahren benutzen erfahrene Test Analysten eine abstrakte, verallgemeinerte Liste mit Punkten, die beachtet und geprüft werden müssen, oder die nicht vergessen werden dürfen. Es kann auch ein Satz von Vorschriften und Kriterien sein, gegen die das Produkt verifiziert werden muss. Diese Checklisten werden aus einer Reihe von Standards, Erfahrungen und anderen relevanten Überlegungen zusammengestellt. Beispiel für einen Checklisten-basierten Test ist eine Checkliste mit Standards für Benutzerschnittstellen, die als Grundlage der Tests einer Anwendung dient.

Anwendbarkeit
Checklisten-basiertes Testen ist am effektivsten in Projekten mit einem erfahrenen Testteam, das die zu testende Software oder den Themenbereich der Checkliste gut kennt (Beispiel: Um eine Checkliste über Benutzerschnittstellen erfolgreich verwenden zu können, kennt sich ein Test Analyst mit dem Testen von Benutzerschnittstellen gut aus, kennt aber möglicherweise die zu testende Software nicht). Da Checklisten abstrakt sind und die einzelnen Schritte, die normalerweise in Testfällen und Testablaufspezifikationen vorgegeben sind, nicht im Detail enthalten, nutzt der Tester sein Wissen, um diese Lücken zu füllen. Dadurch, dass die detaillierten Schritte weggelassen werden, ist der Wartungsaufwand gering und die Checklisten können für mehrere ähnliche Releases benutzt werden. Checklisten können in jeder Teststufe verwendet werden, auch für den Regressionstest und den Smoke-Test.

Einschränkungen/Schwierigkeiten
Die abstrakt gehaltenen Checklisten können die Reproduzierbarkeit der Testergebnisse beeinträchtigen. Es ist gut möglich, dass mehrere Tester die Checklisten unterschiedlich interpretieren und unterschiedliche Ansätze befolgen, um die Punkte der Checkliste zu erfüllen. Dies kann zu unterschiedlichen Ergebnissen führen, selbst wenn dieselbe Checkliste benutzt wurde. Die Folge ist eine breitere Überdeckung, aber die Reproduzierbarkeit der Tests wird manchmal dafür geopfert. Checklisten können manchmal auch zu übermäßig unbegründetem Vertrauen hinsichtlich des erzielten Überdeckungsgrades führen, da der eigentliche Test von der Einschätzung des Testers abhängt. Checklisten können von detaillierteren Testfällen oder Listen abgeleitet werden und neigen dazu, im Laufe der Zeit immer umfangreicher zu werden. Eine Wartung der Checklisten ist nötig, um sicherzustellen, dass sie die wichtigen Aspekte der zu testenden Software abdecken.

Überdeckung
Die Überdeckung ist so gut wie die Checkliste, die verwendet wird. Da die Checkliste jedoch abstrakt gehalten ist, variieren die Ergebnisse je nach Test Analyst, der die Checkliste verwendet.

Fehlerarten
Zu den typischen Fehlerzuständen, die mit diesem Verfahren aufgedeckt werden, gehören Fehlerwirkungen, die als Folge variierender Daten, oder in Zusammenhang mit der Ablaufsequenz

oder dem allgemeinen Workflow beim Testen auftreten. Checklisten können dabei helfen, das Testen aktuell zu halten, da auch neue Kombinationen von Daten und Prozessen während des Testens erlaubt sind.

3.4.3 Exploratives Testen

Beim explorativen Testen lernen die Tester gleichzeitig das Produkt und dessen Fehler kennen, planen vorgesehene Testaufgaben, entwerfen die Tests und führen sie durch, und berichten über die Testergebnisse. Die Tester passen die Testziele während der Testdurchführung dynamisch an und dokumentieren dabei nur leichtgewichtig. [Whittaker09]

Anwendbarkeit
Gutes exploratives Testen ist geplant, interaktiv und kreativ. Es ist wenig Dokumentation über das zu testende System erforderlich. Daher wird exploratives Testen häufig angewendet, wenn keine Dokumentation vorhanden ist oder wenn die Dokumentation für andere Testverfahren nicht ausreichend ist. Exploratives Testen wird oft als Erweiterung anderer Testverfahren eingesetzt, und als Basis zur Erstellung zusätzlicher Testfälle.

Einschränkungen/Schwierigkeiten
Management und Zeitplanung können sich beim explorativen Testen schwierig gestalten. Die Überdeckung kann sporadisch sein, und die Reproduzierbarkeit problematisch. Als eine Methode des Managements beim explorativen Testen können Test-Chartas erstellt werden, die die Aufgaben spezifizieren, die in einer Testsitzung abgedeckt werden sollen, und den dafür verfügbaren Zeitrahmen eingrenzen. Am Ende jeder Testsitzung bzw. einer Menge von Testsitzungen hält der Testmanager eine Abschlussbesprechung, um die Ergebnisse zu sammeln und die Chartas für die nächsten Testsitzungen zu bestimmen. Solche Abschlussbesprechungen lassen sich für große Testteams oder große Projekte nur schwer skalieren.

Eine weitere Schwierigkeit bei explorativen Testsitzungen ist, wie diese in einem Testmanagementsystem genau verfolgt werden können. Dazu werden manchmal explorative Sitzungen als Testfall-Artefakte im Testmanagementsystem angelegt. Dadurch lassen sich die für das explorative Testen zur Verfügung gestellte Zeit und die geplante Überdeckung mit den anderen Testaufgaben verfolgen.

Da die Reproduzierbarkeit beim explorativen Testen schwierig sein kann, kann es zu Problemen kommen, wenn die Schritte zur Reproduktion einer Fehlerwirkung wieder benötigt werden. Manche Unternehmen verwenden die Mitschnittfunktion (Capture/Playback) eines Testautomatisierungswerkzeugs, um die beim explorativen Testen durchgeführten Schritte aufzuzeichnen. Dies liefert eine komplette Aufzeichnung aller Aktivitäten während einer explorativen Testsitzung (bzw. während jeder anderen erfahrungsbasierten Testsitzung). Die Suche nach der tatsächlichen Ursache der beobachteten Fehlerwirkung kann sich zwar mühsam gestalten, aber wenigstens sind die durchgeführten Schritte allesamt aufgezeichnet.

Überdeckung
Es lassen sich Test-Chartas erstellen, die Aufgaben, Ziele und Arbeitsergebnisse spezifizieren. Für explorative Testsitzungen lässt sich damit spezifizieren, was erreicht werden soll. In der Test-Charta kann auch identifiziert werden, worauf man sich konzentrieren sollte, was innerhalb und außerhalb des Leistungsumfangs liegt und welche Ressourcen zur Durchführung der geplanten Tests vorzusehen sind. Eine Testsitzung kann sich auf ganz bestimmte Fehlerarten und andere potenzielle Problembereiche fokussieren, die ohne die Formalität von skriptbasiertem Testen adressiert werden können.

Fehlerarten
Zu den typischen Fehlerzuständen, die mit explorativem Testen aufgedeckt werden, gehören Probleme mit Szenarien, die beim skriptbasierten funktionalen Test übersehen wurden, Probleme in funktionalen Grenzbereichen, sowie Probleme in Zusammenhang mit dem Workflow. Auch Performanz- und Sicherheitsprobleme werden manchmal beim explorativen Testen aufgedeckt.

3.4.4 Anwendung des am besten geeigneten Testverfahrens

Fehler- und erfahrungsbasierte Testverfahren nutzen Kenntnisse über Fehler und andere Erfahrungen im Testen, um die Fehleraufdeckung zielgerichtet 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 skriptbasierten Testen gesucht, um die Überdeckung zu maximieren.
- Betriebsausfälle sollen analysiert werden.

Fehler- und erfahrungsbasierte Testverfahren sind außerdem nützlich, wenn sie mit spezifikationsorientierten Verfahren kombiniert werden, weil sie die Lücken in der Testüberdeckung schließen, die aus den systematischen Schwächen dieser Verfahren resultieren. Wie auch bei den spezifikationsorientierten Verfahren gibt es nicht ein einziges, perfektes Verfahren, das für alle Situationen passt. Der Test Analyst muss die Vor- und Nachteile der einzelnen Verfahren kennen, um das beste Verfahren oder die beste Kombination von Verfahren für eine bestimmte Situation auszuwählen. Dabei sind verschiedene Faktoren (z. B. Art des Projekts, Zeitplan, Zugang zu Informationen, Fachkompetenz der Tester, usw.) zu berücksichtigen, die diese Auswahl beeinflussen können.

4. Softwarequalitätsmerkmale - 120 Minuten

Begriffe

Angemessenheitstest, Attraktivität, Benutzbarkeitstest, Erlernbarkeit, heuristische Evaluation, Interoperabilitätstest, Operabilität, Korrektheitstest, Softwarebenutzbarkeits-Messinventar (SUMI), Verständlichkeit, WAMMI (Website Analysis and MeasureMent Inventory), Zugänglichkeitstest

Lernziele zum Thema Test der Softwarequalitätsmerkmale

4.2 Qualitätsmerkmale bei fachlichen Tests

TA-4.2.1 (K2) Sie können anhand von Beispielen erläutern, welche Testverfahren geeignet sind, um Richtigkeit, Angemessenheit, Interoperabilität und Konformität zu testen

TA-4.2.2 (K2) Sie können die typischen Fehler erläutern, die beim Testen der Richtigkeits-, Angemessenheits- und Interoperabilitätseigenschaften aufgedeckt werden

TA-4.2.3 (K2) Sie können erläutern, in welcher Stufe des Lebenszyklus die Richtigkeits-, Angemessenheits- und Interoperabilitätseigenschaften getestet werden sollten

TA-4.2.4 (K4) Sie können für einen vorgegebenen Projektkontext die Ansätze erläutern, die geeignet wären, um sowohl die Implementierung der Benutzbarkeitsanforderungen als auch die Erfüllung der Benutzererwartungen zu verifizieren und zu validieren

4.1 Einführung

Während das vorige Kapitel die verschiedenen Testverfahren beschreibt, die Testern zur Verfügung stehen, behandelt dieses Kapitel wie Tester die Verfahren anwenden, um die wichtigsten Qualitätsmerkmale für Softwareanwendungen oder -systeme zu bewerten.

Im vorliegenden Lehrplan werden die Qualitätsmerkmale behandelt, die von Test Analysten zu bewerten sind. Die Qualitätsmerkmale, die von Technical Test Analysten zu bewerten sind, werden im Technical Test Analyst Lehrplan (CTAL-TTA) behandelt. Die Beschreibung der Qualitätsmerkmale orientiert sich am ISO Standard 9126, der als Richtschnur verwendet wurde. Weitere Standards, wie z.B. ISO Standard 25000 [ISO25000] (der ISO 9126 abgelöst hat) gelten ebenfalls als nützlich. Die ISO Qualitätsmerkmale sind unterteilt in Produktqualitätsmerkmale (oder -merkmale), die jeweils in weitere Teileigenschaften (oder Teilmerkmale) unterteilt werden können. Diese sind in der folgenden Tabelle zusammengefasst, aus der ebenfalls hervorgeht, in welchem Lehrplan (Test Analyst oder Technical Test Analyst) die jeweiligen Merkmale/Untermerkmale behandelt werden:

Qualitätsmerkmal	Untermerkmale	Test Analyst	Technical Test Analyst
Funktionalität	Richtigkeit, Angemessenheit, Interoperabilität, Einhaltung von Standards (Konformität)	X	
	Sicherheit		X
Zuverlässigkeit	Softwarereife (Robustheit), Fehlertoleranz, Wiederherstellbarkeit, Einhaltung von Standards (Konformität)		X
Benutzbarkeit	Verständlichkeit, Erlernbarkeit, Operabilität, Attraktivität, Einhaltung von Standards (Konformität)	X	
Effizienz	Performanz (Zeitverhalten), Ressourcen-nutzung, Einhaltung von Standards (Konformität)		X
Wartbarkeit	Analysierbarkeit, Änderbarkeit, Stabilität, Testbarkeit, Einhaltung von Standards (Konformität)		X
Portabilität	Anpassbarkeit, Installierbarkeit, Koexistenz, Austauschbarkeit, Einhaltung von Standards (Konformität)		X

Der Test Analyst sollte sich auf das Testen der Qualitätsmerkmale Funktionalität und Benutzbarkeit konzentrieren. Auch für den Zugänglichkeitstest sollte der Test Analyst zuständig sein. Auch wenn die Zugänglichkeit nicht als Teilmerkmal aufgelistet ist, so wird sie gewöhnlich als ein Teil des Benutzbarkeitstests angesehen. Für das Testen der anderen Qualitätsmerkmale ist in der Regel der Technical Test Analyst zuständig. Auch wenn die Verteilung der Zuständigkeiten in unterschiedlichen Organisationen möglicherweise variiert, ist sie in den vorliegenden Lehrplänen des ISTQB so erfolgt.

Das Teilmerkmal “Einhaltung von Standards (Konformität)“ ist bei jedem der Qualitätsmerkmale aufgeführt. In besonderem sicherheitskritischen oder regulatorischen Umfeld müssen einzelne Qualitätsmerkmale bestimmte Standards und Vorschriften erfüllen (Beispiel: Die Einhaltung von Funktionalitätskonformität kann darauf hindeuten, dass die Funktionalität einem bestimmten Standard entsprechen muss, dass z.B. ein bestimmtes Kommunikationsprotokoll verwendet werden muss, damit der Datenaustausch mit einem Chip möglich ist). Da diese Standards je nach Branche stark variieren, werden sie an dieser Stelle nicht eingehend behandelt. Wenn ein Test Analyst in einem Umfeld arbeitet, in dem Anforderungen für die Einhaltung von Standards (Konformität) gelten, dann

muss der Test Analyst diese Anforderungen verstehen und sicherstellen, dass sowohl das Testen als auch die Testdokumentation diese Anforderungen erfüllen.

Für alle in diesem Abschnitt behandelten Qualitäts- und Teilmerkmale müssen die typischen Risiken erkannt werden, damit eine geeignete Teststrategie ausgearbeitet und dokumentiert werden kann. Für das Testen von Qualitätsmerkmale müssen Lebenszyklusabläufe, benötigte Werkzeuge, Verfügbarkeit von Software und Dokumentation sowie technisches Fachwissen besondere Beachtung finden. Wenn es keine geplante Vorgehensweise für jedes einzelne Merkmal und dessen spezifische Testerfordernisse gibt, dann ist es möglich, dass dem Tester nicht genug Zeit für Planung, Vorbereitung und Durchführung der Tests bleibt. Bei einigen der Tests, z.B. Benutzbarkeitstests, müssen Spezialisten, umfangreiche Planung, spezielle Labors, bestimmte Werkzeuge, spezielle Testfähigkeiten, und in den meisten Fällen ein erheblicher Zeitaufwand eingeplant und zugewiesen werden. In manchen Fällen kann der Benutzbarkeitstest auch von einer separaten Gruppe von Benutzbarkeits-Experten (oder User-Experience-Spezialisten) durchgeführt werden.

Das Testen der Qualitätsmerkmale und Teilmerkmale muss in den übergreifenden Testzeitplan integriert werden, und es müssen ausreichend Ressourcen für den Aufwand zugewiesen werden. Jeder der Bereiche hat bestimmte Erfordernisse, befasst sich mit bestimmten Problematiken und kann zu unterschiedlichen Zeitpunkten im Softwarelebenszyklus vorkommen. Dies wird in den nachfolgenden Abschnitten behandelt.

Auch wenn Test Analysten nicht für die Qualitätsmerkmale zuständig sind, die einen mehr technisch ausgerichteten Ansatz erfordern, müssen sie doch diese anderen Qualitätsmerkmale kennen und wissen, welche Bereiche sich beim Testen überschneiden. Beispiel 1: Ein Produkt, das einen Performanztest nicht besteht, wird wahrscheinlich auch den Benutzbarkeitstest nicht bestehen, wenn es zu langsam ist, und der Benutzer es nicht effektiv nutzen kann. Beispiel 2: Wenn es bei einem Produkt Probleme mit der Interoperabilität bei manchen Komponenten gibt, ist es wahrscheinlich noch nicht bereit für den Portabilitätstest, da die zugrunde liegenden Probleme in einer neuen Umgebung weniger erkennbar sind.

4.2 Qualitätsmerkmale bei fachlichen Tests

Funktionale Tests sind ein Aufgabenschwerpunkt von Test Analysten. Die funktionalen Tests sind in erster Linie darauf fokussiert, was das Produkt leistet. Sie basieren im Allgemeinen auf einer Spezifikation oder einem Anforderungsdokument, spezifischer Expertise in einem Fachgebiet oder einem erwarteten Bedarf. Funktionale Tests unterscheiden sich je nach der Teststufe, in der sie durchgeführt werden, und können auch vom Softwarelebenszyklus beeinflusst 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. In einem agilen Umfeld beschränkt sich das funktionale Testen meist auf die Funktionalität, die in einer bestimmten Iteration oder in einem Sprint zur Verfügung gestellt wird, obwohl der Regressionstest für eine Iteration sehr wohl die gesamte freigegebene Funktionalität abdecken kann.

Funktionale Tests setzen viele unterschiedliche Testverfahren ein (siehe Kapitel 3). Dabei kann entweder ein dedizierter Tester, ein Fachexperte oder (wenn es um die Komponententests geht) ein Entwickler die funktionalen Tests durchführen.

Zusätzlich zu den funktionalen Tests, die in diesem Abschnitt beschrieben sind, gibt es noch zwei weitere Qualitätsmerkmale, für die der Test Analyst zuständig ist, und die als nicht-funktional gelten (d.h. sie konzentrieren darauf, wie ein Produkt die benötigte Funktionalität liefert). Es handelt sich dabei um die nicht-funktionalen Qualitätsmerkmale Benutzbarkeit und Zugänglichkeit.

In diesem Abschnitt werden die folgenden Qualitätsmerkmale behandelt:

- Funktionale Qualitätsmerkmale
  - Korrektheit
  - Angemessenheit
  - Interoperabilität
- Nicht-funktionale Qualitätsmerkmale
  - Benutzbarkeit
  - Zugänglichkeit

4.2.1 Test der funktionalen Korrektheit

Funktionale Korrektheitstests prüfen die Einhaltung von spezifizierten oder impliziten Anforderungen an eine Anwendung; sie können auch die Richtigkeit der Berechnungen prüfen. Korrektheitstest nutzen viele der Testverfahren aus Kapitel 3. Oft dient die Spezifikation oder ein vorhandenes System als Testorakel. Korrektheitstest können in jeder Phase des Lebenszyklus durchgeführt werden; mit diesen Tests soll inkorrekte Handhabung von Daten oder Situationen aufgedeckt werden.

4.2.2 Angemessenheitstest

Tests auf Angemessenheit bewerten und validieren, ob sich eine Menge von Funktionen für die vorgesehenen spezifizierten Aufgaben eignen. Die Tests können auf Anwendungsfällen (use cases) basieren. Tests auf Angemessenheit werden meist beim Systemtest durchgeführt, können aber auch in den späteren Stufen des Integrationstests erfolgen. Die Fehlerzustände, die mit dieser Testart aufgedeckt werden, sind Hinweise darauf, dass das System die Erfordernisse der Benutzer nicht in akzeptabler Weise erfüllen wird.

4.2.3 Interoperabilitätstests

Interoperabilitätstests prüfen, inwieweit zwei oder mehr Systeme oder Komponenten Informationen austauschen und die ausgetauschten Informationen verwenden können. Die Tests müssen alle vorgesehenen Umgebungen abdecken (einschließlich der Varianten von Hardware, Software, Middleware, Betriebssystem usw.), um sicherzustellen, dass der Datenaustausch korrekt funktioniert. In der Praxis ist dies möglicherweise nur für eine relativ kleine Anzahl von Umgebungen machbar. In diesem Fall können die Interoperabilitätstests auf eine repräsentative Gruppe von Umgebungen begrenzt werden. 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 Stellen prüfen, die Daten austauschen.

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

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

- die Verwendung von industrieüblichen Kommunikationsstandards, beispielsweise XML
- die Fähigkeit der Software, die Kommunikationsanforderungen anderer Systeme, mit denen sie zusammenwirkt, automatisch zu erkennen und entsprechend anzupassen

Interoperabilitätstests sind besonders wichtig für Unternehmen, die kommerzielle Standardsoftware und -Werkzeuge entwickeln, oder für Unternehmen, die Multisysteme entwickeln.

Während der Komponentenintegrations- und Systemtests ist diese Testart auf das Zusammenwirken des Systems mit seiner Umgebung fokussiert. Beim Systemintegrationstest wird mit diesen Tests

geprüft, wie gut das fertig entwickelte System mit anderen Systemen zusammenwirkt. Da Systeme möglicherweise auf mehreren Ebenen interoperieren, muss der Test Analyst diese Interaktionen verstehen, um in der Lage zu sein, die Bedingungen zu schaffen, mit denen die verschiedenen Interaktionen ausgeführt werden. Beispiel: Wenn zwei Systeme Daten austauschen, muss der Test Analyst in der Lage sein, die notwendigen Daten und Transaktionen zu erzeugen, damit der Datenaustausch stattfinden kann. Es ist zu beachten, dass eventuell nicht alle Interkationen in den Anforderungsdokumenten klar spezifiziert sind. Viele der Interaktionen sind stattdessen in der Dokumentation von Systemarchitektur und Systementwurf spezifiziert. Daher muss der Test Analyst in der Lage und bereit sein, diese Dokumente zu untersuchen, damit die Punkte des Datenaustausches zwischen dem System und seiner Umgebung bestimmt werden können; nur so ist sichergestellt, dass auch all diese Punkte getestet werden. Testverfahren, wie Entscheidungstabellen, Zustandsübergangsdiagramme, Anwendungsfälle und kombinatorisches Testen, sind allesamt im Interoperabilitätstest anwendbar. Zu den typischen Fehlerzuständen, die aufgedeckt werden, gehört der inkorrekte Datenaustausch zwischen interagierenden Komponenten.

4.2.4 Benutzbarkeitstest

Es ist wichtig zu verstehen, warum Benutzer mit dem System Schwierigkeiten haben könnten. Die Benutzer eines Systems, die es zu verstehen gilt, können sehr unterschiedlichen Personengruppen angehören, angefangen von IT-Experten bis hin zu Kindern oder Menschen mit besonderen Einschränkungen.

Einige nationale Verbände (beispielsweise das British Royal National Institute for the Blind) empfehlen, dass Webseiten auch für behinderte, blinde, sehbehinderte oder taube Nutzer und für Menschen mit Bewegungs- oder Wahrnehmungseinschränkungen zugänglich sein sollten. Die Prüfung, ob eine Webseite für diese Personengruppen nutzbar ist, verbessert unter Umständen auch die Benutzbarkeit für alle anderen. Mehr zum Thema Zugänglichkeit, siehe weiter unten.

Benutzbarkeitstests prüfen, wie einfach es für die Nutzer ist, das System zu nutzen bzw. zu erlernen, um damit spezifizierte Ziele in bestimmten Anwendungskontexten zu erreichen. Beim Benutzbarkeitstest wird folgendes gemessen:

- Effektivität: Eignung der Softwareanwendung 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

Im Benutzbarkeitstest 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

Benutzbarkeitstests werden normalerweise in zwei Stufen durchgeführt:

- Formativer Benutzbarkeitstest – Diese Tests werden iterativ während der Entwurfs- und Prototyping-Stufen durchgeführt und sollen durch die Identifizierung von Entwurfsfehlern, die für die Benutzbarkeit relevant sind, dabei helfen, den Entwurf zu „formen“.

- Summativer Benutzbarkeitstest – Diese Tests werden nach der Realisierung durchgeführt, um die Benutzbarkeit zu messen, und um Probleme mit einer fertigen Komponente oder mit dem System zu identifizieren.

Die Tester sollten für den Benutzbarkeitstest Wissen oder Expertise in folgenden Wissensbereichen haben:

- Soziologie
- Psychologie
- Einhaltung nationaler Standards oder Vorschriften (einschließlich Zugänglichkeitsstandards)
- Ergonomie

4.2.4.1 Benutzbarkeitstests durchführen

Das implementierte System sollte unter möglichst 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. Formale Benutzbarkeitstests können mit echten Benutzern oder mit Benutzervertretern durchgeführt werden, die oft für ihre Aufgabe vorbereitet werden müssen, indem sie Drehbücher oder Anweisungen erhalten, die sie befolgen müssen. In anderen freien Tests dürfen die Benutzer mit der Software experimentieren, damit die Beobachter bestimmen können, wie einfach oder wie schwierig es für die Benutzer ist herauszufinden, wie sie ihre Aufgaben erfüllen können.

Viele Benutzbarkeitstests können vom Test Analyst als Teil anderer Tests durchgeführt werden, beispielsweise beim funktionalen Systemtest. Um ein konsistentes Herangehen an Aufdeckung und Berichterstattung von Benutzbarkeitsfehlern in allen Phasen des Softwarelebenszyklus zu unterstützen, ist eine Ergonomierichtlinie hilfreich. Ohne eine solche Richtlinie kann es schwierig sein zu bestimmen, was „unakzeptable“ Benutzbarkeit ist. Beispiel: Ist es unzumutbar, wenn ein Benutzer 10 Mausklicks benötigt, um sich in eine Anwendung einzuloggen? Ohne spezifische Richtlinien kann der Test Analyst in die schwierige Lage kommen, dass er einen Fehlerbericht verteidigen muss, den der Entwickler schließen möchte, weil die Software dem Design entspricht. Es ist sehr wichtig, dass die verifizierbaren Benutzbarkeitsspezifikationen in den Anforderungen enthalten sind, und dass außerdem ein Satz Ergonomierichtlinien existiert, die für alle ähnlichen Projekte gelten.

Diese Richtlinien sollten folgende Punkte abdecken: Zugänglichkeit von Anleitungen, Klarheit von Eingabeaufforderungen, Anzahl der Klicks für die Durchführung einer Aufgabe, Fehlermeldungen, Verarbeitungsindikatoren (die dem Benutzer anzeigen, dass das System momentan mit der Verarbeitung beschäftigt ist und keine weiteren Eingaben annehmen kann), Richtlinien über Bildschirmlayout, Verwendung von Farben und Geräuschen sowie weitere Faktoren, die Einfluss auf die Benutzererfahrung haben.

4.2.4.2 Benutzbarkeitstestspezifikation

Die wichtigsten Verfahren für Benutzbarkeitstests sind

- Inspektionen, Bewertungen oder Reviews
- Dynamische Interaktion mit Prototypen
- Verifizierung und Validierung der tatsächlichen Implementierung
- Durchführung von Befragungen

Inspektionen, Bewertungen oder Reviews

Inspektionen oder Reviews von Anforderungsspezifikation und Entwürfen unter dem Gesichtspunkt der Benutzbarkeit erhöhen die Nutzerbeteiligung und können kosteneffektiv sein, weil sie Probleme früh entdecken. 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 Gutachterteam die Schnittstelle und ihre Konformität mit anerkannten Grundsätzen der Benutzbarkeit (den heuristischen Grundsätzen der Ergonomie). Reviews sind effektiver, wenn die Benutzerschnittstelle sichtbar ist. Beispiel: Screenshots sind leichter zu verstehen und zu interpretieren als eine Beschreibung einer bestimmten Bildschirmfunktionalität in Textform. Visualisierung ist wichtig, damit ein angemessenes Review der Dokumentation auf Benutzbarkeit möglich ist.

Dynamische Interaktion mit Prototypen
Wenn Prototypen entwickelt werden, sollte der Test Analyst sich mit diesen beschäftigen und den Entwicklern bei der Weiterentwicklung behilflich sein, indem sie Benutzerfeedback in den Entwurf einfließen lassen. Auf diese Weise lassen sich Prototypen verfeinern, und der Nutzer kann einen realistischen Einblick zu Look and Feel des fertigen Produkts bekommen und wie es zu benutzen sein wird.

Verifizierung und Validierung der tatsächlichen Implementierung
Wenn Benutzbarkeitsmerkmale der Software in den Anforderungen spezifiziert sind (z.B. die Anzahl der Mausklicks für eine bestimmte Aufgabe), dann sollten Testfälle entworfen werden, die verifizieren, dass die spezifizierten Eigenschaften in der Implementierung berücksichtigt wurden.

Zur Validierung der tatsächlichen Implementierung 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 Erlernbarkeit oder Operabilität.

Es lassen sich spezifische Benutzbarkeitstestszenarien für das Testen von Syntax und Semantik entwickeln. Syntax betrifft Aufbau oder Grammatik der Schnittstelle (z.B. was in ein Eingabefeld eingegeben werden kann). Semantik beschreibt dagegen Bedeutung und Zweck der Schnittstelle (z.B. sinnvolle und aussagekräftige Systemmeldungen und -ausgaben an den Nutzer).

Black-Box-Verfahren (wie beispielsweise in Abschnitt 3.2 beschrieben), insbesondere Anwendungsfälle, die entweder in Textform oder in UML (Unified Modeling Language) definiert werden können, werden manchmal im Benutzbarkeitstest angewendet.

Testszenarien für Benutzbarkeitstests sollten auch Anleitungen für die Nutzer enthalten, Zeiträume für Interviews vor und nach den Tests vorsehen, um die Nutzer einzuweisen bzw. um von ihnen Rückmeldungen einzusammeln, sowie ein vereinbartes Protokoll für die Durchführung der Testsitzungen. Das Protokoll beschreibt, wie der Test ausgeführt wird, den zeitlichen Ablauf, Protokollierung und Aufzeichnung der Testsitzung sowie die vorgesehenen Methoden für Befragung und Beaufsichtigung.

Durchführung von Umfragen und Fragebögen
Mit Umfragen und Fragebögen lassen sich Erkenntnisse und Feedback über das Verhalten der Anwender bei der Systemnutzung sammeln. Standardisierte und öffentlich zugängliche Umfragen, 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 auch konkrete Benutzbarkeitsmaße, die sich als Testende- oder Abnahmekriterien verwenden lassen.

4.2.5 Zugänglichkeitstests

Es ist wichtig, die Zugänglichkeit der Software für Menschen mit besonderen Bedürfnissen oder eingeschränkten Nutzungsmöglichkeiten zu prüfen. Dazu gehören auch behinderte Personen. Beim Benutzbarkeitstest sollte die Einhaltung der relevanten Standards berücksichtigt werden, wie etwa Richtlinien über Barrierefreiheit (Web Content Accessibility Guidelines) oder gesetzliche Vorschriften, wie z.B. Anti-Diskriminierungsgesetze (Disability Discrimination Acts, Großbritannien, Australien) und Section 508 (USA). Ähnlich wie bei der Benutzbarkeit müssen auch die Aspekte der Zugänglichkeit in der Entwurfsphase berücksichtigt werden. Testen auf Zugänglichkeit erfolgt meist in den Integrationsteststufen und setzt sich im Systemtest und im Abnahmetest fort. Die Fehler ergeben sich meist aus der Tatsache, dass die Software die spezifizierten Vorschriften oder Standards nicht erfüllt.

5. Reviews - 165 Minuten

Begriffe

keine

Lernziele zum Thema Reviews

5.1 Einführung

TA-5.1.1 (K2) Sie können erklären, warum die Vorbereitung des Reviews für den Test Analysten wichtig ist

5.2 Checklisten in Reviews verwenden

TA-5.2.1 (K4) Sie können einen Anwendungsfall oder eine Benutzerschnittstelle analysieren und Probleme anhand der im Lehrplan enthaltenen Checklisten identifizieren

TA-5.2.2 (K4) Sie können eine Anforderungsspezifikation oder eine User Story analysieren und Probleme anhand der im Lehrplan enthaltenen Checklisten identifizieren

5.1 Einführung

Zu einem erfolgreichen Review-Prozess gehören das Planen von Reviews, die Durchführung und die Nachbereitung. Test Analysten 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 Review-Prozessen besser verstehen. Alle Reviewteilnehmer 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.

Unabhängig von der Art des Reviews, das durchgeführt wird, muss der Test Analyst genug Zeit für die Vorbereitung einplanen. Diese Zeit wird benötigt, um das Arbeitsergebnis zu prüfen, um die Dokumente zu prüfen, auf die verwiesen wird, um zu verifizieren, dass das Arbeitsergebnis mit diesen Dokumenten konsistent ist, und um zu bestimmen, was im Arbeitsergebnis fehlt. Ohne angemessene Vorbereitungszeit könnte der Beitrag des Test Analysten darauf beschränkt sein, den bereits vorhandenen Inhalt des Dokuments zu überarbeiten, anstatt an einem effizienten Review teilzunehmen, bei dem die Zeit des Review-Teams so effizient wie möglich genutzt wird, um das bestmögliche Feedback zu liefern. Zu einem guten Review gehört es, die Inhalte zu verstehen, zu bestimmen, ob etwas fehlt und wenn ja was, und zu verifizieren, dass das beschriebene Produkt mit anderen, bereits entwickelten Produkten konsistent ist (bzw. mit anderen Produkten, die derzeit entwickelt werden). Beispiel: Beim Review eines Stufentestkonzepts für den Integrationstest muss der Test Analyst auch die Objekte berücksichtigen, die integriert werden sollen. Welche Bedingungen müssen erfüllt sein, damit diese integriert werden können? Gibt es Abhängigkeiten, die dokumentiert werden müssen? Sind Daten verfügbar, um die Integrationsstellen zu testen? Ein Review konzentriert sich nicht allein auf das Arbeitsergebnis, das geprüft wird; sondern es muss auch die Interaktion des Review-Gegenstandes mit anderen Objekten des Systems berücksichtigt werden.

Es passiert leicht, dass der Autor eines im Review geprüften Arbeitsergebnisses sich kritisiert fühlt. Der Test Analyst sollte bei Reviewbefunden immer vor Augen halten, dass die Zusammenarbeit mit dem Autor letztlich dazu dient, das bestmögliche Arbeitsergebnis zu erzielen. Mit einer solchen Grundhaltung werden die Kommentare konstruktiv formuliert sein, und sie werden sich am Review-Gegenstand orientieren und nicht am Autor. Beispiel: Wenn eine Aussage zweideutig ist, dann ist es besser zu sagen “Ich verstehe nicht, was ich testen soll, um zu verifizieren, ob diese Anforderung korrekt implementiert wurde. Können Sie mir helfen, das besser zu verstehen?“ anstatt „Die Anforderung ist zweideutig, die wird keiner verstehen können“. Aufgabe des Test Analysten beim Review ist es sicherzustellen, dass die im Arbeitsergebnis gelieferten Informationen ausreichend sind, um das Testen zu unterstützen. Wenn Informationen nicht vorhanden, nicht klar und eindeutig sind, oder wenn sie nicht detailliert genug sind, dann ist dies möglicherweise ein Fehler, der vom Autor korrigiert werden muss. Eine positive statt eine kritische Grundhaltung trägt dazu bei, dass Kommentare besser entgegengenommen und die Sitzungen produktiver werden.

5.2 Checklisten in Reviews verwenden

Checklisten werden bei Reviews verwendet, damit die Teilnehmer angehalten sind, bestimmte Punkte im Laufe des Reviews zu prüfen. Sie können auch dazu beitragen, Reviews zu entpersonalisieren, z.B. mit der Aussage, “Dies ist dieselbe Checkliste, die für alle Reviews verwendet wird, nicht nur für das vorliegende Arbeitsergebnis.“ Checklisten können allgemein gehalten sein und für alle Reviews verwendet werden, oder sie können sich speziell mit bestimmten Qualitätsmerkmale, Themenbereichen oder Dokumenten befassen. Beispiel: Mit einer allgemeinen Checkliste können die allgemeinen Eigenschaften des Dokuments verifiziert werden, wie beispielsweise, ob das Dokument eine eindeutige Kennung hat, dass es keine Verweise auf Punkte gibt, die noch offen sind, dass die Formatierung und ähnliche Elemente korrekt bzw. konform sind. Eine spezifische Checkliste für ein

Anforderungsdokument könnte vorgeben, dass überprüft wird, ob die Begriffe “soll” und “sollte” richtig verwendet sind, oder ob jede Anforderung testbar ist. Auch das Format der Anforderungen kann darauf hinweisen, welche Art von Checkliste zu verwenden ist. Für ein Anforderungsdokument in Textform werden andere Review-Kriterien verwendet als für eines, das auf Diagrammen basiert.

Checklisten können auch auf die spezifischen Kompetenzen der Programmierer / Systemarchitekten oder der Tester ausgerichtet sein. Diese Checklisten können die unten aufgeführten Punkte enthalten.

Checklisten, die für Anforderungen, Anwendungsfälle oder User Stories verwendet werden, haben generell einen anderen Fokus als Checklisten, die für den Programmcode oder die Systemarchitektur verwendet werden. Diese an den Anforderungen orientierten Checklisten könnten die folgenden Punkte beinhalten:

- Testbarkeit der einzelnen Anforderungen
- Abnahmekriterien für die einzelnen Anforderungen
- Verfügbarkeit einer Anwendungsfall-Aufrufstruktur, falls zutreffend
- Eindeutige Identifikation bzw. Kennung jeder Anforderung / Anwendungsfall / User Story
- Versionierung der einzelnen Anforderungen / Anwendungsfälle / User Stories
- Rückverfolgbarkeit jeder einzelnen Anforderung von den Anforderungen des Fachbereichs/Marketings
- Verfolgbarkeit zwischen Anforderungen und Anwendungsfällen

Obige Punkte sind lediglich als Beispiele gedacht. Es ist zu beachten, dass wenn eine Anforderung nicht testbar ist, ein Fehlerzustand in der Anforderung vorliegt. Nicht testbar bedeutet, dass die Anforderung so spezifiziert ist, dass der Test Analyst nicht bestimmen kann, wie sie getestet werden soll. Beispiel: Die Anforderung „Die Software soll sehr benutzerfreundlich sein“ ist nicht testbar. Wie soll ein Test Analyst bestimmen, ob eine Software benutzerfreundlich oder gar sehr benutzerfreundlich ist? Wenn die Anforderung stattdessen festlegt „Die Software muss den Benutzbarkeitsstandards entsprechen, die im Benutzbarkeitsstandard-Dokument spezifiziert sind“, dann ist die Anforderung testbar (vorausgesetzt das Benutzbarkeitsstandard-Dokument existiert). Es handelt sich außerdem um eine übergreifende Anforderung, die jedes Element der Benutzerschnittstelle betrifft. In diesem Fall könnte diese eine Anforderung bei einer komplexen Anwendung viele einzelne Testfälle hervorbringen. Auch ist die Verfolgbarkeit von dieser Anforderung, bzw. vom Benutzbarkeitsstandard-Dokument, zu den Testfällen kritisch; sollte es Änderungen der Benutzbarkeitsspezifikation geben, auf die in der Anforderung verwiesen wurde, dann müssten alle Testfälle geprüft und bei Bedarf aktualisiert werden.

Eine Anforderung gilt auch dann als untestbar, wenn der Tester nicht bestimmen kann, ob der Test bestanden wurde oder nicht, oder wenn kein Test entwickelt werden kann, der bestanden oder nicht bestanden werden kann. Beispiel für eine nicht testbare Anforderung: „Das System soll 100% der Zeit zur Verfügung stehen, 24 Stunden pro Tag, 7 Tage pro Woche, 365 (bzw. 366) Tage pro Jahr”.

In einer einfachen Review-Checkliste für Anwendungsfälle könnten folgende Fragen enthalten sein:

- Ist der Hauptpfad (Szenario) genau spezifiziert?
- Sind alle alternativen Pfade (Szenarien) identifiziert, einschließlich der Fehlerbehandlung?
- Sind die Meldungen der Benutzerschnittstelle spezifiziert?
- Gibt es nur einen Hauptpfad (es sollte nur einen geben, denn sonst sind mehrere Anwendungsfälle nötig)?
- Ist jeder Pfad testbar?

In einer einfachen Checkliste zum Benutzbarkeits-Review der Benutzerschnittstelle einer Anwendung könnten folgende Fragen enthalten sein:

- Ist jedes Feld und dessen Funktion spezifiziert?
- Sind alle Fehlermeldungen spezifiziert?

- Sind alle Benutzeraufforderungen spezifiziert und konsistent?
- Ist die Tabfolge der Felder spezifiziert?
- Gibt es alternativ zu Mausaktionen auch Keyboardeingaben?
- Sind Tastenkombinationen (“Shortcuts”) für den Benutzer spezifiziert (z.B. ausschneiden & einfügen)?
- Gibt es Abhängigkeiten zwischen den Feldern (z.B. ein Datum muss später sein als ein anderes)?
- Ist ein Bildschirmlayout vorgegeben?
- Entspricht das Bildschirmlayout den spezifizierten Anforderungen?
- Gibt es eine Anzeige für den Benutzer, die zeigt, dass das System gerade verarbeitet?
- Erfüllt der Bildschirm die Mindestanzahl von Mausklicks (falls spezifiziert)?
- Ist der Navigationsfluss für den Benutzer logisch und basierend auf den Anwendungsfall?
- Erfüllt der Bildschirm etwaige Anforderungen bezüglich der Erlernbarkeit?
- Gibt es Hilfetexte für den Benutzer?
- Gibt es schwebende Textanzeigen?
- Wird dies für den Benutzer “attraktiv” sein (subjektive Bewertung)?
- Sind die verwendeten Farben konsistent mit anderen Anwendungen und mit Unternehmensstandards?
- Werden die Soundeffekte richtig verwendet und sind sie konfigurierbar?
- Erfüllt der Bildschirm die Anforderungen bezüglich Lokalisierung?
- Ist für den Benutzer ersichtlich, was zu tun ist (Verständlichkeit) (subjektive Bewertung)?
- Kann sich der Benutzer daran erinnern, was zu tun ist (Erlernbarkeit) (subjektive Bewertung)?

In agilen Projekten liegen die Anforderungen meist als User Stories vor. Es handelt sich dabei um kleine Inkremente demonstrierbarer Funktionalitäten. Während es sich bei einem Anwendungsfall um eine Benutzertransaktion handelt, die mehrere Bereiche von Funktionalität durchläuft, ist eine User Story eine isoliertere Einheit, deren Umfang im Allgemeinen durch die Zeit bestimmt wird, die für ihre Entwicklung notwendig ist. In einer Checkliste für eine User Story könnte enthalten sein:

- Ist die Story angemessen für die vorgesehene Iteration/Sprint?
- Sind die Abnahmekriterien spezifiziert und sind diese testbar?
- Ist die Funktionalität genau definiert?
- Gibt es Abhängigkeiten zwischen dieser User Story und anderen User Stories?
- Ist die User Story priorisiert?
- Umfasst die User Story ein Inkrement von Funktionalität?

Falls eine User Story eine neue Schnittstelle definiert, dann ist es natürlich besser, eine allgemeine Checkliste (siehe Beispiel oben) und eine detaillierte Benutzerschnittstellen-Checkliste zu verwenden. Checklisten können an die folgenden Grundlagen angepasst werden:

- Unternehmen / Organisation (z.B. Berücksichtigung der Unternehmenspolitik, -standards, -konventionen)
- Projekt / Entwicklungsaufwand (z.B. Schwerpunkt, technische Standards, Risiken)
- Review-Objekt (Code-Reviews können auf eine bestimmte Programmiersprache ausgerichtet sein)

Gute Checklisten decken Probleme auf, und sie initiieren Diskussionen über andere Punkte, auf die die Checkliste möglicherweise nicht verweist. Wenn beim Review verschiedene Checklisten kombiniert werden, dann ist das ein probates Mittel, dass das Review die beste Qualität für das Arbeitsergebnis erzielt. Durch die Verwendung von Standard-Checklisten, wie diejenigen, auf die im Foundation-Level-Lehrplan verwiesen wird, sowie durch die Entwicklung eigener, unternehmensinterner Checklisten, wie diejenigen, die oben beschrieben wurden, kann der Test Analyst einen effektiven Beitrag bei Reviews leisten.

Mehr Informationen über Reviews und Inspektionen, siehe [Gilb93] und [Wiegers03].

6. Fehlermanagement – 120 Minuten

Begriffe

Fehlertaxonomie, Fehlereindämmung innerhalb der Phase, Grundursachenanalyse

Lernziele zum Thema Fehlermanagement

6.2 Wann lässt sich ein Fehlerzustand aufdecken?

TA-6.2.1 (K2) Sie können erklären, wie eine Fehlereindämmung innerhalb einer Phase Kosten reduzieren kann

6.3 Die Pflichtfelder in Fehlerberichten

TA-6.3.1 (K2) Sie können erklären, welche Informationen zur Dokumentation eines nicht-funktionalen Fehlerzustands notwendig sein können

6.4 Fehlerklassifizierung

TA-6.4.1 (K4) Sie können für einen vorgegebenen Fehlerzustand die Informationen zur Fehlerklassifizierung identifizieren, sammeln und aufzeichnen

6.5 Grundursachenanalyse

TA-6.5.1 (K2) Sie können Sinn und Zweck der Grundursachenanalyse erklären

6.1 Einführung

Test Analysten bewerten das Verhalten des Systems aus der geschäftlichen und Benutzerperspektive, beispielsweise: Würden Nutzer wissen, was zu tun ist, wenn diese Meldung erscheint, oder wenn das System sich so verhält? Der Test Analyst bestimmt, ob sich das System korrekt verhält, indem er tatsächliche und erwartete Ergebnisse vergleicht. Eine Anomalie (auch Abweichung genannt) ist ein unerwartetes Ereignis, das näher untersucht werden muss. Es kann sich bei einer Anomalie um eine Fehlerwirkung handeln, die durch einen Fehlerzustand verursacht wird. Eine Anomalie muss kein Fehlerzustand sein und kann zur Erstellung eines Fehlerberichts führen oder nicht. Ein Fehlerzustand ist ein tatsächliches Problem, das festgestellt wurde und behoben werden muss.

6.2 Wann lässt sich ein Fehlerzustand aufdecken?

Ein Fehlerzustand lässt sich durch statische Tests finden; die Symptome eines Fehlerzustands, d.h. die Fehlerwirkung, können dagegen nur durch dynamisches Testen aufgedeckt werden. Für jede Phase des Softwarelebenszyklus sollte es Methoden 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 ein Fehlerzustand erkannt und korrigiert wird, desto geringer sind die Kosten der Qualität für das Gesamtsystem. Statische Tests können Fehlerzustände aufdecken, bevor dynamisches Testen möglich ist. Dies ist einer der Gründe, weshalb der statische Test eine kosteneffektive Methode für die Entwicklung qualitativ hochwertiger Software ist.

Mit Hilfe des Fehlerverfolgungssystems sollte der Test Analyst die Lebenszyklusphase aufzeichnen können, in der der Fehlerzustand entstanden ist, und die Phase, in der der Fehlerzustand aufgedeckt wurde. Falls die beiden Phasen identisch sind, dann ist eine Fehlereindämmung innerhalb der Phase gelungen. Mit anderen Worten ist der Fehler wurde in ein- und derselben Phase entstanden und behoben, und ist nicht in eine nachfolgende Phase entkommen. Ein Beispiel dafür ist eine inkorrekte Anforderung, die im Anforderungs-Review gefunden und an Ort und Stelle korrigiert wird. Dies ist nicht nur ein effizienter Einsatz eines Anforderungs-Reviews, sondern verhindert auch, dass der Fehlerzustand später zusätzlichen Aufwand verursacht und für das Unternehmen so teurer würde. Wenn eine inkorrekte Anforderung übersehen wird (“entkommt”), und anschließend vom Entwickler implementiert, vom Test Analyst getestet, und schließlich vom Benutzer beim Abnahmetest gefunden wird, dann war die ganze Arbeit an dieser Anforderung eine Zeitverschwendung (von einem möglichen Vertrauensverlust des Nutzers in das System ganz zu schweigen).

Fehlereinsämmung innerhalb der Phase ist eine effektive Methode, die Kosten von Fehlern zu reduzieren.

6.3 Die Pflichtfelder in Fehlerberichten

Die Felder (Parameter) eines Fehlerberichts sollen ausreichend Informationen liefern, damit entsprechende Maßnahmen zur Fehlerbehebung ergriffen werden können. Ein für die Festlegung der Maßnahmen geeigneter Fehlerbericht ist:

- Vollständig – der Bericht enthält alle benötigten Informationen
- Kurz und prägnant – der Bericht enthält keine unnötigen und irrelevanten Informationen

- Richtig – die Informationen im Bericht sind richtig, die erwarteten und die tatsächlichen Ergebnisse sind genau spezifiziert, die Schritte zum Reproduzieren sind enthalten
- Objektiv – der Bericht ist professionell verfasst und bezieht sich auf Tatsachen

Die im Fehlerbericht enthaltenen Informationen sollten in Datenfelder unterteilt sein. Je besser diese Felder definiert sind, desto leichter ist es, Fehler zu berichten sowie Berichte über Fehlertrends und sonstige zusammenfassende Berichte zu erstellen. Wenn für ein Feld mehrere Optionen möglich sind, dann bieten „Dropdown-Listen“ zur Auswahl der Eingabewerte eine Möglichkeit, den Zeitaufwand für die Aufzeichnung eines Fehlerzustands zu verringern. Solche „Dropdown-Listen“ sind jedoch nur dann effektiv, wenn die Anzahl möglicher Optionen begrenzt ist und der Benutzer nicht durch eine lange Liste von Auswahlmöglichkeiten scrollen muss, um die zutreffende Option zu finden. Unterschiedliche Fehlerarten benötigen unterschiedliche Fehlerberichtsinformationen. Daher sollte das Fehlermanagementwerkzeug flexibel sein und je nach Fehlerart die richtigen Felder abfragen. Daten sollten in spezifischen Feldern aufgezeichnet werden, die im Idealfall mit einer Datenvalidierungsfunktion ausgestattet sind, um Fehler bei der Dateneingabe zu vermeiden und um eine effektive Berichterstattung sicherzustellen.

Fehlerberichte werden für aufgedeckte Fehlerwirkungen während des funktionalen und nicht-funktionalen Testens erstellt. Die Informationen im Fehlerbericht sollten immer zum Ziel haben, das Szenario, in dem das Problem entdeckt wurde, genau zu identifizieren, einschließlich der benötigten Schritte und Daten, um das Szenario zu reproduzieren, sowie der erwarteten und der tatsächlichen Ergebnisse. Für nicht-funktionale Fehlerberichte sind mehr Details über die Umgebung, sonstige Performanz-Parameter (z.B. Höhe der Last), Reihenfolge der Schritte und erwartete Ergebnisse nötig. In einem Fehlerbericht über einen Benutzbarkeitsfehlerzustand ist es wichtig, dass beschrieben wird, welche Softwareaktion der Nutzer erwartet hat. Beispiel: Wenn der Benutzbarkeits-Standard festlegt, dass eine Aktion weniger als vier Mausklicks benötigt, dann sollte im Fehlerbericht enthalten sein, wieviele Mausklicks tatsächlich benötigt wurden im Gegensatz zum erwähnten Standard. Wenn kein Standard zur Verfügung steht und die nicht-funktionalen Qualitätsmerkmale der Software in den Anforderungen nicht abgedeckt sind, kann der Tester einen Test mit einer „vernünftigen Person“ durchführen, um zu bestimmen, ob die Benutzbarkeit akzeptabel ist. In diesem Fall müssen die Erwartungen dieser Testperson im Fehlerbericht klar dargelegt werden. Da die nicht-funktionalen Anforderungen in der Anforderungsdokumentation manchmal fehlen, ist das Dokumentieren des erwarteten und tatsächlichen Systemverhaltens in Zusammenhang mit nicht-funktionalen Fehlerwirkungen für Tester oft eine besondere Herausforderung.

Während Fehlerberichte normalerweise zum Ziel haben, die Behebung eines Problems herbeizuführen, müssen die im Fehlerbericht gelieferten Informationen auch die richtige Fehlerklassifizierung, die Risikoanalyse und Prozessverbesserungen unterstützen.

6.4 Fehlerklassifizierung

Ein Fehlerbericht kann während seines Lebenszyklus mehrere verschiedene Klassifizierungsstufen haben. Die richtige Klassifizierung von Fehlern ist integraler Bestandteil einer korrekten Fehlerberichterstattung. Die Klassifizierung wird genutzt, um Fehlerzustände zu gruppieren, um die Effektivität des Testens und des Entwicklungslebenszyklus zu bewerten und um interessante Trends zu erkennen.

Die Klassifizierung neu identifizierter Fehlerzustände beinhaltet häufig die folgenden Informationen:

- Projektaktivität, die durchgeführt wurde, als der Fehlerzustand entdeckt wurde (z.B. Review, Audit, Inspektion, Programmieren, Testen)
- Projektphase, in der der Fehlerzustand entstanden ist (falls bekannt) (z.B. Anforderungen, Entwurf, Feinentwurf, Implementierung)

- Projektphase, in der der Fehlerzustand entdeckt wurde (z.B. Anforderungen, Entwurf, Feinentwurf, Implementierung, Code-Review, Modultest, Integrationstest, Systemtest, Abnahmetest)
- Vermutete Ursache des Fehlerzustands – (z.B. Anforderungen, Entwurf, Schnittstelle, Programmcode, Daten)
- Wiederholbarkeit – (z.B. einmal, sporadisch, reproduzierbar)
- Symptom(e) – (z.B. Systemabsturz, Systemstillstand, Benutzerschittstellenfehler, System-fehler, Performanz)

Nachdem der Fehlerzustand analysiert wurde, kann eine weitere Klassifizierung erfolgen:

- Grundursache – der Fehler, der gemacht wurde, und aus dem der Fehlerzustand resultierte (z.B. Prozess, Programmierfehler, Benutzerfehler, Testfehler, Konfigurationsproblem, Datenproblem, Fremdsoftware, externes Softwareproblem, Dokumentationsproblem)
- Quelle – das Arbeitsprodukt, in dem der Fehler gemacht wurde (z.B. Anforderungen, Entwurf, Feinentwurf, Architektur, Datenbankdesign, Benutzerdokumentation, Testdokumentation)
- Art – (z.B. Logikproblem, Berechnungsproblem, zeitliches Problem, Datenhandling, Verbesserung)

Nachdem der Fehlerzustand behoben ist (oder zurückgestellt wurde oder nicht bestätigt werden konnte), sind noch weitere Klassifizierungsinformationen zum Fehlerzustand verfügbar, wie z.B.:

- Lösung – (z.B. Programmänderung, Dokumentationsänderung, zurückgestellt, kein Problem, Duplikat)
- Korrekturmaßnahme(n) – (z.B. Anforderungs-Review, Code-Review, Modultest, Konfigurationsdokumentation, Datenvorbereitung, keine Änderung erfolgt)

Zusätzlich zu diesen Klassifizierungen werden Fehlerzustände häufig auch nach Schweregrad und Priorität klassifiziert. Je nach Projektkontext kann es außerdem Sinn machen, die Fehlerzustände nach deren sicherheitsrelevanten Auswirkung, auf Projektzeitplan, auf Projektkosten, auf Projektrisiken und auf Projektqualität zu klassifizieren. Diese Klassifizierungen können Beachtung finden, wenn vereinbart werden soll, wie schnell ein Fehler zu beheben ist.

Die letztendliche Fehlerklassifizierung entsteht bei der Schließung des Fehlerzustands. Fehlerzustände werden häufig entsprechend des Abschlusses klassifiziert und gruppiert z.B. behoben/verifiziert, geschlossen/kein Problem, zurückgestellt, offen/nicht behoben. Diese Fehlerklassifzierung wird häufig während des Projekts verwendet, um die Fehlerzustände während ihres Lebenszyklus zu verfolgen.

Die Klassifizierungsdaten, die in Unternehmen zum Einsatz kommen, werden häufig individuell angepasst. Bei den oben beschriebenen Beispielen handelt es sich lediglich um einige übliche Werte, die in der Industrie verwendet werden. Es ist jedoch wichtig, dass die Klassifizierungswerte konsistent verwendet werden, damit sie nützlich sind. Wenn es zu viele Klassifizierungsfelder gibt, ist das Öffnen und Bearbeiten von Fehlermeldungen relativ zeitintensiv. Es sollte daher im Einzelfall beurteilt werden, ob der Wert der gesammelten Daten die gestiegenen Kosten für jeden bearbeiteten Fehler rechtfertigt. Ein wichtiger Faktor bei der Werkzeugauswahl ist die individuelle Anpassbarkeit der Klassifizierungswerte.

6.5 Grundursachenanalyse

Zweck einer Grundursachenanalyse ist es zu bestimmen, was den Fehlerzustand verursacht hat und Daten zur Unterstützung von Prozessänderungen zu liefern. Diese Prozessänderungen sollen Grundursachen eliminieren, die für einen bedeutenden Teil der Fehlerzustände verantwortlich sind. Die Grundursachenanalyse wird meist von der Person durchgeführt, die einen Fehlerzustand untersucht und entweder das Problem behebt, oder aber bestimmt, dass das Problem nicht behoben

werden kann oder nicht behoben werden soll. In der Regel ist diese Person der Entwickler. Die vorläufige Grundursachen erfolgt häufig durch den Test Analyst, der eine auf Erfahrung basierende Vermutung über die wahrscheinliche Ursache des Problems äußert. Nachdem die Behebung des Problems bestätigt wurde, verifiziert der Test Analyst die vom Entwickler eingetragene Grundursache. In Zusammenhang mit der Bestimmung der Grundursache wird häufig auch die Phase bestimmt oder bestätigt, in der der Fehlerzustand entstanden ist.

Zu den typischen Grundursachen von Fehlerzuständen gehören:

- Unklare Anforderung
- Fehlende Anforderung
- Falsche Anforderung
- Inkorrekte Implementierung des Entwurfs
- Inkorrekte Implementierung der Schnittstelle
- Logikfehler im Code
- Berechnungsfehler
- Hardwarefehler
- Schnittstellenfehler
- Ungültige Daten

Die Informationen über Grundursachen werden zusammengeführt, um die Themen zu bestimmen, die zu Fehlerzuständen führen. Beispiel: Wenn viele Fehlerzustände durch unklare Anforderungen verursacht werden, dann würde es Sinn machen, mehr Aufwand in die Durchführung effektiver Anforderungs-Reviews zu investieren. Ein weiteres Beispiel: Falls mehrere Entwicklungsteams Probleme bei der Implementierung von Schnittstellen haben, dann wären vielleicht gemeinsame Entwurfssitzungen angebracht.

Wenn die Informationen über Grundursachen zu Prozessverbesserungen genutzt werden, dann kann ein Unternehmen den Nutzen effektiver Prozessänderungen überwachen, denn damit lassen sich die Kosten von Fehlerzuständen quantifizieren, die auf eine bestimmte Grundursache zurückzuführen sind. Dies kann hilfreich sein, wenn es darum geht, finanzielle Mittel für Prozessänderungen zu bekommen, für die zusätzliche Werkzeuge und Ausrüstung beschafft werden müssen, oder die sich auf den Zeitplan auswirken. Das Thema Grundursachenanalyse wird im ISTQB Expert Level Lehrplan „Improving the Test Process“ [ISTQB_EL_ITP] detaillierter behandelt.

7. Testwerkzeuge - 45 Minuten

Begriffe

schlüsselwortgetriebener Test, Testausführungswerkzeug, Testdateneditor und -generator, Testentwurfswerkzeug

Lernziele zum Thema Testwerkzeuge

7.2 Testwerkzeuge und Automatisierung

TA-7.2.1 (K2) Sie können die Vorteile für den Einsatz von Testdateneditoren und -generatoren, Testentwurfswerkzeugen und Testausführungswerkzeugen erläutern

TA-7.2.2 (K2) Sie können die Rolle des Test Analysten bei der schlüsselwortgetriebenen Automatisierung erläutern

TA-7.2.3 (K2) Sie können die Schritte für die Fehleranalyse einer Fehlerwirkung bei einer automatisierten Testausführung erläutern

7.1 Einführung

Mit Testwerkzeugen lassen sich Effizienz und Effektivität des Testens in Bezug auf den Testaufwand erheblich steigern, allerdings nur dann, wenn die geeigneten Werkzeuge fachgerecht eingesetzt werden. Testwerkzeuge müssen als ein zusätzlicher Aspekt einer gut geführten Testorganisation gemanagt werden. Es bestehen große Unterschiede in Ausgereiftheit und Anwendbarkeit der Werkzeuge, und zusätzlich ändert sich der Markt für Testwerkzeuge ständig. Werkzeuge sind meist von kommerziellen Anbietern sowie von verschiedenen Freeware- oder Shareware-Webseiten erhältlich.

7.2 Testwerkzeuge und Automatisierung

Ein Grossteil der Arbeit eines Test Analysten betrifft den effizienten Einsatz von Testwerkzeugen. Dazu muss der Test Analyst wissen, welche Werkzeuge wann einzusetzen sind. Dies kann die eigene Effizienz des Test Analysten erhöhen und dazu beitragen, in der verfügbaren Zeit eine bessere Testabdeckung zu erzielen.

7.2.1 Testentwurfswerkzeuge

Testentwurfswerkzeuge werden verwendet, um die Erzeugung von Testfällen und Testdaten für das Testen zu unterstützen. Dazu verwenden die Werkzeuge bestimmte Formate der Anforderungs-spezifikation (z.B. UML) oder Eingaben des Test Analysten. Testentwurfswerkzeuge sind häufig so entworfen und gebaut worden, damit sie mit bestimmten Formaten und Produkten zusammenarbeiten können, wie beispielsweise spezifische Anforderungsmanagementwerkzeuge.

Testentwurfswerkzeuge können dem Test Analyst Informationen darüber liefern, welche Tests er braucht, um die angestrebte Testabdeckung, Vertrauen in das System oder die Maßnahmen zur Reduzierung des Produktrisikos zu erzielen. Ein Beispiel dafür wären Klassifikationsbaumwerkzeuge, die eine Menge von Kombinationen generieren und darstellen, die für eine Erfüllung vordefinierter Überdeckungskriterien erforderlich sind. Diese Informationen dienen dem Test Analysten dazu, die auszuführenden Testfälle festzulegen.

7.2.2 Testdateneditoren und -generatoren

Testdateneditoren und -generatoren bieten einige Vorteile. Einige dieser Werkzeuge sind in der Lage, ein Dokument (z.B. ein Anforderungsdokument) oder sogar den Quellcode zu analysieren und die Daten zu bestimmen, die beim Testen für eine angestrebten Überdeckungsgrad benötigt werden. Andere Testdateneditoren und -generatoren können einen Datensatz aus einem Produktivsystem so „bereinigen“ oder anonymisieren, dass die darin enthaltenen persönlichen Daten entfernt werden, die interne Integrität der Daten jedoch erhalten bleibt. Die bereinigten Daten können dann für Testzwecke verwendet werden, ohne eine Sicherheitslücke oder einen Missbrauch der persönlichen Daten zu riskieren. Das ist vor allem dann von Bedeutung, wenn große Mengen realistischer Daten benötigt werden. Andere Testdatengeneratoren können Testdaten aus einer vorhandenen Menge von Eingabeparametern generieren (z.B. für den Test mit Zufallsdaten). Manche dieser Werkzeuge sind in der Lage, die Datenbankstruktur zu analysieren, um zu bestimmen, welche Eingaben vom Test Analyst benötigt werden.

7.2.3 Automatisierte Testausführungswerkzeuge

Testausführungswerkzeuge werden überwiegend von Test Analysten auf allen Teststufen eingesetzt, um Tests auszuführen und die Ergebnisse zu dokumentieren. Testausführungswerkzeuge verfolgen normalerweise eines oder mehrere der folgenden Ziele:

- Kostenreduktion
- Durchführen von weiteren Tests
- Durchführung der selben Tests in unterschiedlichen Umgebungen
- Tests leichter wiederholbar machen
- Tests ausführen, die manuell nicht ausgeführt werden könnten (z.B. umfangreiche Prüfungen der Datenvalidierung)

Diese Ziele überschneiden sich und lassen sich in den Hauptzielen zusammenfassen: Steigerung des Überdeckungsgrades bei gleichzeitiger Reduzierung der Kosten.

7.2.3.1 Anwendbarkeit

Die Rentabilität von Testausführungswerkzeugen ist meist dann am höchsten, wenn diese zur Automatisierung von Regressionstests eingesetzt werden; Gründe dafür sind der erwartete, geringe Wartungsaufwand und die wiederholte Ausführung der Tests. Auch die Automatisierung von Smoke-Tests kann eine effektive Anwendungsmöglichkeit sein, aufgrund der häufigen Ausführung der Tests und weil das Ergebnis schnell benötigt wird. Obwohl bei dieser Anwendung die Wartungskosten höher sein können, bietet sie doch eine automatisierte Methode für die Bewertung neuer Softwareversionen in einer kontinuierlichen Integrationsumgebung.

Testausführungswerkzeuge werden überwiegend in den System- und Integrationsteststufen eingesetzt. Manche Werkzeuge, insbesondere API-Testwerkzeuge, können auch im Komponententest eingesetzt werden. Die sinnvolle Nutzung der Werkzeuge und deren zielgerichteter Einsatz können deren Rentabilität steigern.

7.2.3.2 Grundlagen der Testausführungswerkzeuge

Testausführungswerkzeuge führen eine Reihe von Anweisungen in einer Programmiersprache aus, die oft auch als Skriptsprache bezeichnet wird. Die Anweisungen für das Werkzeug sind sehr detailliert und spezifizieren Eingaben, Reihenfolge der Eingaben, bestimmte Eingabewerte und die erwarteten Ausgaben. Diese detaillierten Skripte werden dadurch sehr anfällig für Änderungen des Testobjekts vor allem wenn sie die grafische Benutzerschnittstelle (GUI) betreffen.

Die meisten Testausführungswerkzeuge haben einen Testkomparator, der die tatsächlichen mit den gespeicherten erwarteten Testergebnissen vergleicht.

7.2.3.3 Testautomatisierung implementieren

Beim automatisierten Testen (wie beim Programmieren) geht der Trend weg von detaillierten Anweisungen und hin zu höheren Sprachen. Es werden auch hier Bibliotheken, Makros und Subroutinen benutzt. Bei Testentwurfsverfahren, wie beim schlüsselwortgetriebenen oder aktionswortgetriebenen Testen, werden Folgen von Anweisungen mit einem bestimmten Namen (Schlüsselwort oder Aktionswort) gekennzeichnet. Dies ermöglicht es dem Test Analyst, die Testfälle in natürlicher Sprache zu erstellen, ohne die zugrundeliegende Programmiersprache und Detailfunktionen zu beachten. Durch die sehr modularen Testskripte wird die Wartung bei Änderungen an der Funktionalität oder an der Schnittstelle der zu testenden Software erleichtert [Bath08]. Die schlüsselwortgetriebene Testautomatisierung wird weiter unten weitergehend behandelt.

Die Bestimmung von Schlüssel- oder Aktionswörtern kann auf Modellen basieren, z.B. auf Geschäftsprozessmodellen, die häufig in den Anforderungsdokumenten enthalten sind. Auf dieser Grundlage lassen sich die wichtigen Geschäftsprozesse bestimmen, die getestet werden müssen. Danach können die Schritte dieser Prozesse bestimmt werden, einschließlich der Entscheidungspunkte im Verlauf des Prozesses. Die Entscheidungspunkte können in Aktionswörter umgewandelt werden, die die Testautomatisierung aus den Schlüsselwort- bzw. Aktionsworttabellen erkennen und

verwenden kann. Die Geschäftsprozess-Modellierung ist eine Methode zur Abbildung von Geschäftsprozessen und kann eingesetzt werden, um die wichtigen Prozesse und Entscheidungspunkte zu identifizieren. Die Modellierung kann manuell oder mit Werkzeugunterstützung durchgeführt werden; bei der werkezuggestützten Modellierung dienen die Geschäftsregeln und Prozessbeschreibungen als Eingabequellen.

7.2.3.4 Den Erfolg der Testautomatisierung verbessern

Bei der Entscheidung, welche Tests automatisiert werden sollen, muss jeder in Frage kommende Testfall bzw. jede Testsuite 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. Optimal ist wahrscheinlich, wenn jeder gegebene Satz von Testfällen oder jede Testsuite manuelle, halbautomatisierte und automatisierte Tests enthält.

Bei der Implementierung eines Automatisierungsprojekts zur Testausführung sollten folgende Aspekte berücksichtigt werden:

Mögliche Nutzen:

- Die Zeit für die automatisierte Testdurchführung lässt sich leichter schätzen.
- Regressionstests und Fehlervalidierung in späteren Projektphasen sind mit automatisierten Tests schneller und sicherer.
- Der Einsatz von Testautomatisierungswerkzeugen kann den Status und die technische Kompetenz des Testers oder des Testteams erhöhen.
- Automatisierung ist besonders hilfreich bei iterativen und inkrementellen Entwicklungslebenszyklen, um besseres Regressionstesten bei jedem Build oder bei jeder Iteration zu ermöglichen.
- Die Abdeckung bestimmter Testarten ist möglicherweise nur mit Testautomatisierung möglich (z.B. umfangreiche Datenvalidierungsaufgaben).
- Eine automatisierte Testdurchführung kann kosteneffektiver sein als manuelles Testen, wenn es um Eingabe, Konvertierung und Vergleich großer Datenmengen geht, da die Automatisierung konsistente Eingaben und Verifizierungen liefert.

Mögliche Risiken:

- Unvollständiges, ineffektives oder inkorrektes manuelles Testen wird genauso automatisiert, ohne Optimierungen durchzuführen.
- Es ist schwierig, die Testmittel zu warten; wenn die zu testende Software geändert wird, müssen mehrere Änderungen nachgezogen werden.
- Da die Tester bei der Testdurchführung nicht mehr direkt involviert sind, kann die Fehlerfindungsrate sinken
- Die Fähigkeiten im Testteam sind für einen effektiven Einsatz der Testautomatisierungs-werkzeuge möglicherweise nicht ausreichend.
- Möglicherweise werden irrelevante Tests, die nicht zur Gesamttestüberdeckung beitragen, automatisiert, nur weil sie existieren und stabil sind.
- Mit zunehmender Stabilisierung der Software können Tests überflüssig werden (Wiederholungen haben keine Wirksamkeit, Pesticide Paradoxon)

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, fest programmierte Eingabewerte durch Variablen erweitert und die Vorteile des Testwerkzeugs in vollem Umfang genutzt werden. Viele Testausführungswerkzeuge haben die Fähigkeit zum Verbinden von Testfällen, Gruppieren von Tests, Wiederholen, Ändern der Reihenfolge, und sie bieten 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 die Anwendung der Testwerkzeuge, Programmierung und Entwurfsverfahren sehr wertvoll, damit sich die Vorteile der Werkzeuge vollständig nutzen lassen.

Bei der Testplanung sollte Zeit eingeplant werden, um die automatisierten Testfälle regelmäßig auch manuell auszuführen, damit das Wissen nicht verloren geht, wie der Test funktioniert, und um die korrekte Arbeitsweise zu verifizieren, sowie um die Gültigkeit von Eingabedaten und deren Überdeckung zu prüfen.

7.2.3.5 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 die Funktionalität des Schlüsselworts implementieren. Die Testskripte sind sehr modular implementiert, damit sie sich leicht auf bestimmte Schlüsselwörter abbilden lassen. Für die Implementierung der modularen Skripte sind Programmierkenntnisse erforderlich.

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.
- Eine Person, die vorwiegend Fachexpertise hat, kann von der automatischen Testfalldurchführung profitieren (nachdem die Schlüsselwörter in Form von Testskripten implementiert wurden) denn sie braucht den zugrundeliegenden Automatisierungscode nicht zu verstehen.
- Testfälle, die unter Verwendung der Schlüsselwörter geschrieben wurden, sind leichter wartbar, weil sie mit geringerer Wahrscheinlichkeit modifiziert werden müssen, wenn sich Details der zu testenden Software ändern.
- Testfallspezifikationen sind unabhängig von ihrer Implementierung. Die Schlüsselwörter können mit verschiedenen Skriptsprachen und Werkzeugen implementiert werden.

Die automatisierten Skripte (der eigentliche Automatisierungscode), die die Schlüssel- bzw. Aktionswörter verwenden, werden normalerweise von Entwicklern oder Technical Test Analysten erstellt, während die Test Analysten normalerweise für die Erstellung und Pflege der Schlüsselwort- bzw. Aktionswortdaten zuständig sind. Während die schlüsselwortgetriebenen automatisierten Tests normalerweise in der Systemtestphase durchgeführt werden, kann die Erstellung des Codes bereits in den Integrationstestphasen beginnen. Bei iterativen Vorgehensweisen ist die Entwicklung der automatisierten Tests ein fortlaufender Prozess.

Nachdem Schlüsselwörter und Eingabedaten erstellt wurden, sind die Test Analysten normalerweise für die Ausführung der schlüsselwortgetriebenen Testfälle und die Analyse von aufgedeckten Fehlerwirkungen zuständig. Wenn eine Anomalie entdeckt wird, muss der Test Analyst die Ursache der Fehlerwirkung untersuchen, um herauszufinden, ob das Problem durch die Schlüsselwörter, die Eingabedaten, die automatisierten Testskripts oder durch die getestete Anwendung verursacht wird.

Der erste Schritt bei der Fehleranalyse ist meist die manuelle Durchführung des betroffenen Tests mit denselben Daten, um herauszufinden, ob die Fehlerwirkung die Anwendung selbst betrifft. Wenn der manuelle Test keine Fehlerwirkung aufgedeckt hat, dann sollte der Test Analyst die Reihenfolge der Tests prüfen, bei der es zur Fehlerwirkung kam. So lässt sich feststellen, ob der Ursprung des Problems zu einem früheren Zeitpunkt erfolgte (vielleicht durch die Produktion inkorrekter Daten), sich aber erst später bei der Verarbeitung zeigte. Wenn der Test Analyst die Ursache der Fehlerwirkung nicht bestimmen kann, dann sollten die Informationen der Fehleranalyse an den Technical Test Analyst oder Entwickler zur weiteren Analyse übergeben werden.

7.2.3.6 Gründe für das Scheitern der Testautomatisierung

Häufig erreichen Testautomatisierungsprojekte die gesteckten Ziele nicht. Die Gründe für dieses Scheitern können sein: unzureichende Flexibilität bei der Verwendung des Testwerkzeugs, mangelnde Programmierfähigkeiten im Testteam oder unrealistische Erwartungen hinsichtlich der Probleme, die durch die Testautomatisierung gelöst werden können. Es ist zu beachten, dass jede Testautomatisierung, genau wie jedes andere Softwareentwicklungsprojekt, Management, Aufwand, Fähigkeiten und Aufmerksamkeit in Anspruch nimmt. Für die Erstellung einer nachhaltigen Systemarchitektur, die Befolgung geeigneter Entwurfsmethoden, für das Konfigurationsmanagement und für den Einsatz guter Programmiertechniken muss Zeit bereitgestellt werden. Die automatisierten Testskripte müssen getestet werden, weil sie wahrscheinlich Fehler enthalten. Möglicherweise muss die Performanz optimiert werden. Die Benutzbarkeit der Werkzeuge muss berücksichtigt werden, nicht nur für den Entwickler, sondern auch für die Personen, die die Werkzeuge zum Ausführen der Skripte benutzen werden. Es kann notwendig sein, eine Schnittstelle zwischen Werkzeug und Benutzer zu entwerfen, die Zugriff auf die Testfälle ermöglicht und für den Tester logisch aufgebaut ist. Diese Schnittstelle muss aber trotzdem die vom Werkzeug benötigte Zugänglichkeit bieten.

8. Referenzen

8.1 Standards

- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE)
  Kapitel 1 und 4
- [ISO9126] ISO/IEC 9126-1:2001, Software Engineering - Software Product Quality,
  Kapitel 1 und 4
- [RTCA DO-178B/ED-12B]: Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12B.1992.
  Kapitel 1

8.2 Dokumente des ISTQB

- [ISTQB_AL_OVIEW] ISTQB Advanced Level Übersicht, Entwurf, Version 0.1
- [ISTQB_ALTM_SYL] ISTQB Advanced Level Testmanager Syllabus, Entwurf, Version 0.1
- [ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Entwurf, Version 0.1
- [ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2011
- [ISTQB_GLOSSARY] ISTQB/GTB Standardglossar der Testbegriffe, Version 2.1, 2011

8.3 Literatur

[Bath08] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook”, Rocky Nook, 2008, ISBN 978-1-933952-24-6

[Beizer95] Boris Beizer, “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

[Black07]: Rex Black, “Pragmatic Software Testing”, John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Buwalda01]: Hans Buwalda, “Integrated Test Design and Automation”, Addison-Wesley Longman, 2001, ISBN 0-201-73725-6

[Cohn04]: Mike Cohn, “User Stories Applied: For Agile Software Development”, Addison-Wesley Professional, 2004, ISBN 0-321-20568-5

[Copeland03]: Lee Copeland, “A Practitioner's Guide to Software Test Design”, Artech House, 2003, ISBN 1-58053-791-X

[Craig02]: Rick David Craig, Stefan P. Jaskiel, “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]: Tom Gilb, 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 proceedings STAR 1994

[Koomen06]: Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon "TMap NEXT, for result driven testing", UTN Publishers, 2006, ISBN 90-72194-80-2

[Myers79]: Glenford J. Myers, “The Art of Software Testing”, John Wiley & Sons, 1979, ISBN 0-471-46912-2

[Splaine01]: Steven Splaine, Stefan P. Jaskiel, “The Web-Testing Handbook”, STQE Publishing, 2001, ISBN 0-970-43630-0

[vanVeenendaal12]: Erik van Veenendaal, “Practical risk-based testing – The PRISMA approach”, UTN Publishers, The Netherlands, ISBN 9789490986070

[Wiegers03]: Karl Wiegers, “Software Requirements 2”, Microsoft Press, 2003, ISBN 0-735-61879-8

[Whittaker03]: James Whittaker, “How to Break Software”, Addison-Wesley, 2003, ISBN 0-201-79619-8

[Whittaker09]: James Whittaker, “Exploratory Software Testing”, Addison-Wesley, 2009, ISBN 0-321-63641-4

8.4 Sonstige Referenzen

Die folgenden Referenzen verweisen auf Informationen im Internet. Diese Referenzen wurden zum Zeitpunkt der Veröffentlichung dieses Advanced Level Lehrplans geprüft. Das ISTQB übernimmt keine Verantwortung dafür, wenn diese Referenzen nicht mehr verfügbar sind.

- Kapitel 3
  - Czerwonka, Jacek: www.pairwise.org
  - Bug Taxonomy: www.testingeducation.org/a/bsct2.pdf
  - Sample Bug Taxonomy based on Boris Beizer’s work: inet.uni2.dk/~vinter/bugtaxst.doc
  - Good overview of various taxonomies: testingeducation.org/a/bugtax.pdf
  - Heuristic Risk-Based Testing By James Bach
  - 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
- Kapitel 4
  - www.testingstandards.co.uk

9. Index