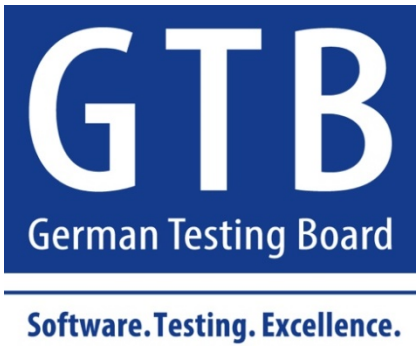
Certified Tester
Advanced Level Syllabus
Test Analyst

Version 2019



German Testing Board e.V.

Übersetzung des englischsprachigen Lehrplans des International Software Testing Qualifications Board (ISTQB®), Originaltitel: Certified Tester Advanced Level, Test Analyst Syllabus des ISTQB, Fassung 2019.

Änderungshistorie

Version	Datum	Bemerkungen
V2019	05.04.2020	Deutschsprachige Fassung freigegeben

Inhaltsverzeichnis

Dank

Der ISTQB CTAL TA Syllabus wurde von einem Kernteam der Advanced Level-Arbeitsgruppe des International Software Testing Qualifications Board (ISTQB) erstellt: Graham Bath, Judy McKay, Mike Smith.

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

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

Laura Albert
Chris Van Bael
Markus Beck
Henriett Braunné Bokor
Guo Chaonian
Wim Decoutere
Milena Donato
Klaudia Dussa-Zieger
Melinda Eckrich-Brajer
Péter Földházi Jr
David Frei
Chen Geng
Matthias Hamburg
Zsolt Hargitai
Zhai Hongbao
Tobias Horn
Ágota Horváth
Beata Karpinska
Ramit Manohar Kaul
Jan te Kock
József Kreisz
Dietrich Leimsner
Ren Liang
Claire Lohr
Attila Kovács
Rik Marselis
Marton Matyas
Don Mills
Blair Mo
Gary Mogyorodi
Ingvar Nordström
Francisca Cano Ortiz
Pálma Polyák
Meile Posthuma
Lloyd Roden
Adam Roman
Abhishek Sharma
Péter Sótér
Lucjan Stapp
Andrea Szabó
Benjamin Timmermans
Erik van Veenendaal
Jan Versmissen
Carsten Weise
Robert Werkhoven
Paul Weymouth

Die englische Originalausgabe wurde von der Hauptversammlung des ISTQB® am 18. Oktober 2019 offiziell freigegeben.

Das German Testing Board (GTB) dankt dem Reviewteam der deutschsprachigen Fassung 2019: Matthias Hamburg, Marc-Florian Wendland, Monika Bögge, Dr. Klaudia Dussa-Zieger (Leitung).

0. Einführung in den Lehrplan

0.1 Zweck dieses Dokuments

Dieser Lehrplan bildet die Grundlage für das Softwaretest-Qualifizierungsprogramm Test Analyst der Aufbaustufe (Advanced Level Test Analyst). Das ISTQB® und das GTB® stellen diesen Lehrplan folgenden Adressaten zur Verfügung:

1. Nationalen/regionalen Boards zur Übersetzung in die jeweilige Landessprache(n) und zur Akkreditierung von Trainingsprovidern. 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. Zertifizierungsstellen zur Ableitung von Prüfungsfragen in ihrer Landessprache, die sich an den Lernzielen der jeweiligen Lehrpläne orientieren.
3. Trainingsprovidern zur Erstellung von Kursmaterialien und zur Bestimmung angemessener Lehrmethoden.
4. Prüfungskandidaten zur Vorbereitung auf die Zertifizierungsprüfung (entweder als Teil eines Seminars oder kursunabhängig).
5. Allen Personen weltweit, die im Bereich Software- und Systementwicklung tätig sind, zur Förderung des Berufsbildes des Software- und Systemtesters, 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 und erhalten.

0.2 Der Certified Tester Advanced Level Test Analyst

Die Advanced Level Core Qualifizierung besteht aus drei separaten Lehrplänen, die sich auf folgende Rollen beziehen:

- Test Manager
- Test Analyst
- Technical Test Analyst

Die ISTQB Advanced Level Overview 2019 [ISTQB_AL_OVIEW] ist ein separates Übersichtsdokument, das folgende Informationen enthält:

- Geschäftlicher Nutzen für die einzelnen Lehrpläne
- Matrix mit Verfolgbarkeit zwischen geschäftlichem Nutzen und Lernzielen
- Zusammenfassung der einzelnen Lehrpläne
- Beziehungen zwischen den Lehrplänen

0.3 Prüfungsrelevante Lernziele und kognitive Stufen

Die Lernziele unterstützen den jeweiligen geschäftlichen Nutzen und dienen zur Ausarbeitung der Prüfung für die Zertifizierung als CTAL Test Analyst.

Den einzelnen Lernzielen ist jeweils eine kognitive Stufe des Wissens, K2, K3 oder K4, zugeordnet, die jeweils ist am Anfang des Kapitels aufgeführt und wie folgt klassifiziert ist:

- K2: Verstehen
- K3: Anwenden

- K4: Analysieren

Die Definitionen aller Begriffe, die direkt unter den Kapitelüberschriften als Schlüsselbegriffe aufgeführt sind, sollen wiedergegeben werden können (K1), auch wenn diese in den Lernzielen nicht ausdrücklich erwähnt werden.

0.4 Die Prüfung

Die Zertifizierungsprüfung für den CTAL Test Analyst basiert auf diesem Lehrplan. Zur Beantwortung einer Prüfungsfrage kann Wissen aus mehreren Abschnitten dieses Lehrplans erforderlich sein. Alle Abschnitte dieses Lehrplans sind prüfungsrelevant, außer der Einführung und der Anhänge. Im Lehrplan sind bibliografische Referenzen zu Standards, Fachbücher und andere ISTQB®-Lehrpläne enthalten. Prüfungsrelevant sind jedoch nur die im vorliegenden Lehrplan zusammengefassten Inhalte der referenzierten Materialien.

Das Format der Prüfung ist Multiple Choice, bestehend aus 40 Fragen. Zum Bestehen der Prüfung müssen mindestens 65% der Fragen korrekt beantwortet werden.

Prüfungen können als Teil eines akkreditierten Trainingsseminars oder unabhängig davon (z.B. bei einer Zertifizierungsstelle oder in einer öffentlichen Prüfung) abgelegt werden. Die Teilnahme an einem akkreditierten Trainingsseminar stellt keine Voraussetzung für das Ablegen der Prüfung dar.

0.5 Voraussetzung für die Prüfung

Voraussetzung für die Zulassung zur Prüfung zum CTAL Test Analysten ist das erworbene Zertifikat zum ISTQB® Certified Tester Foundation Level (CTFL®).

0.6 Erwartete Erfahrung

Keines der Lernziele für den CTAL Test Analysten geht davon aus, dass spezifische Erfahrungen vorhanden sind.

0.7 Akkreditierung von Trainingskursen

Nationale ISTQB-Mitgliedsboards können Trainingsprovider akkreditieren, deren Kursmaterial diesem Lehrplan entspricht. Die Trainingsprovider sollten sich von ihrem nationalen Board oder von der Stelle, die die Akkreditierungen durchführt, die entsprechenden Akkreditierungsrichtlinien einholen. Ein akkreditierter Trainingskurs ist als lehrplankonform anerkannt, und darf im Rahmen des Kurses eine ISTQB-Prüfung beinhalten.

0.8 Detaillierungsgrad des Lehrplans

Der Detaillierungsgrad dieses Lehrplans ermöglicht international einheitliche Kurse und Prüfungen. Um dieses Ziel zu erreichen, besteht der Lehrplan aus den folgenden Elementen:

- Allgemeine Lehrziele, die die Absicht des CTAL Test Analyst-Lehrplans beschreiben.
- Eine Auflistung der Begriffe, an die sich Schulungsteilnehmer erinnern müssen.
- Lernziele der einzelnen Wissensgebiete, die die zu erreichenden kognitiven Lernergebnisse beschreiben.
- Eine Beschreibung der Schlüsselkonzepte, einschließlich Verweisen auf Quellen wie anerkannte Literatur oder Normen.

Der Lehrplaninhalt ist keine Beschreibung des gesamten Wissensgebietes, sondern spiegelt den Detaillierungsgrad wider, der in Advanced Level-Trainingsseminaren abzudecken ist. Der Lehrplan konzentriert sich auf Materialien, die für alle Softwareprojekte gelten können, einschließlich agiler Projekte. Der Lehrplan enthält keine spezifischen Lernziele in Bezug auf einen bestimmten Softwareentwicklungslebenszyklus, aber es wird besprochen, wie diese Konzepte in agilen Projekten, anderen Ausprägungen von iterativen und inkrementellen Lebenszyklen und in sequenziellen Lebenszyklen angewendet werden.

0.9 Aufbau des Lehrplans

Der Lehrplan besteht aus sechs Kapiteln mit prüfungsrelevanten Inhalten. Die Kapitelüberschrift spezifiziert die Mindestzeit, die für den Unterricht und die Übungen des Kapitels vorgesehen ist; eine weitere Differenzierung der Zeitangabe je Unterkapitel ist nicht vorgesehen. Für akkreditierte Trainingskurse erfordert der Lehrplan eine Unterrichtszeit von mindestens 20 Stunden und 30 Minuten, die sich wie folgt auf die sechs Kapitel verteilt:

- Kapitel 1: Die Aufgaben des Test Analysten im Testprozess (150 Minuten)
- Kapitel 2: Die Aufgaben des Test Analysten beim risikobasierten Testen (60 Minuten)
- Kapitel 3: Testverfahren (630 Minuten)
- Kapitel 4: Das Testen von Softwarequalitätsmerkmalen (180 Minuten)
- Kapitel 5: Reviews (120 Minuten)
- Kapitel 6: Testwerkzeuge und Testautomatisierung (90 Minuten)

1. Die Aufgaben des Test Analysten im Testprozess - 150 min

Schlüsselbegriffe

abstrakter Testfall, Endekriterien, konkreter Testfall, Test, Testablauf, Testanalyse, Testausführungsplan, Testbasis, Testbedingung, Testdaten, Testdurchführung, Testentwurf, Testrealisierung, Testsuite

Lernziele für die Aufgaben des Test Analysten im Testprozess

1.1 Einführung

Keine Lernziele

1.2 Testen im Softwareentwicklungslebenszyklus

TA-1.2.1 (K2) Erklären, wie und warum sich der Zeitpunkt und Grad der Beteiligung des Test Analysten bei verschidenen Softwareentwicklungslebenszyklusmodellen unterscheiden

1.3 Testanalyse

TA-1.3.1 (K2) Die Aufgaben von Test Analysten bei der Durchführung von Testanalyseaktivitäten zusammenfassen

1.4 Testentwurf

TA-1.4.1 (K2) Erklären, weshalb die Stakeholder die Testbedingungen verstehen sollten

TA-1.4.2 (K4) Für ein gegebenes Projektszenario bestimmen, welches Abstraktionsniveau für die Testfälle am besten geeignet ist

TA-1.4.3 (K2) Die Faktoren erläutern, die bei der Entwicklung von Testfällen zu berücksichtigen sind

1.5 Testrealisierung

TA-1.5.1 (K2) Die Aufgaben von Test Analysten bei der Durchführung von Testrealisierungs-aktivitäten zusammenfassen

1.6 Testdurchführung

TA-1.6.1 (K2) Die Aufgaben von Test Analysten bei den Testdurchführungsaktivitäten zusammenfassen

1.1 Einführung

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

- Testplanung
- Testüberwachung und -steuerung
- Testanalyse
- Testentwurf
- Testrealisierung
- Testdurchführung
- Testabschluss

In diesem Lehrplan für den Advanced Level Test Analyst werden die Aktivitäten vertieft, die für den Test Analysten von besonderer Bedeutung sind. Dies ermöglicht eine weitere Verfeinerung des Testprozesses, um diesen besser an verschiedene Softwarelebenszyklus-Modellen anzupassen.

Schwerpunktmäßig befasst sich der Test Analyst mit der Bestimmung der geeigneten Tests, sowie mit deren Entwurf, Realisierung und Durchführung. Auch wenn es wichtig ist, die anderen Schritte im Testprozess zu verstehen, so wird sich ein Test Analyst in seiner Arbeit größtenteils auf die folgenden Aktivitäten fokussieren:

- Testanalyse
- Testentwurf
- Testrealisierung
- Testdurchführung

Die anderen Aktivitäten des Testprozesses sind im Foundation Level-Lehrplan beschrieben und bedürfen keiner weiteren Ausarbeitung in diesem Lehrplan.

1.2 Testen im Softwareentwicklungslebenszyklus

Bei der Definition einer Teststrategie sollte der gesamte Softwareentwicklungslebenszyklus berücksichtigt werden. Je nach gewähltem Softwareentwicklungslebenszyklusmodell unterscheidet sich der Zeitpunkt, an dem der Test Analyst an den Testaktivitäten beteiligt wird. Auch der Grad seiner Beteiligung, Zeitaufwand, verfügbare Informationen und Erwartungen können stark variieren. Der Test Analyst muss über die Art der Informationen Bescheid wissen, die an andere organisatorische Aufgabenbereiche zu liefern sind, wie z.B.:

- Anforderungsanalyse und Anforderungsmanagement - Anforderungsreviews
- Projektmanagement - Eingaben für den Zeitplan
- Konfigurations- und Änderungsmanagement - Tests zur Verifizierung von Softwareversionen, Versionskontrolle
- Softwareentwicklung – Mitteilung von Fehlerzuständen, die gefunden wurden
- Softwarewartung - Fehlermanagement, Fehlerbearbeitungszeit (d.h. Zeit zum Erkennen und Melden von Anomalien, dann die Zeit zum Durchführen und Melden von Fehlernachtests)
- Technischer Support - genaue Dokumentation von Lösungsalternativen und bekannten Problemen
- Erstellen technischer Dokumentation (z.B. Datenbankentwurfspezifikationen, Testumgebungsdokumentation) – sowohl Eingaben für diese Dokumente als auch technisches Review dieser Dokumente

Die Testaktivitäten müssen an das gewählte Softwareentwicklungslebenszyklusmodell angepasst werden, das sequenziell, iterativ, inkrementell oder eine Mischform aus diesen sein kann. Beim

sequenziellen V-Modell lässt sich beispielsweise der Testprozess auf Stufe des Systemtests folgendermaßen einbinden:

- Der Systemtest wird gleichzeitig mit dem Projekt geplant, und Testüberwachung und -steuerung dauern bis zum Testabschlusss an. Dies beeinflusst die Beiträge des Test Analysten zur Zeitplanung durch das Projektmanagement.
- Systemtestanalyse und -entwurf richten sich nach Dokumenten wie die Anforderungsspezifikation, System- und (logischen) Architekturentwurf, sowie nach dem (technischen) Komponentenentwurf.
- Die Implementierung der Systemtestumgebung kann während des Systementwurfs beginnen, obwohl der größte Aufwand normalerweise gleichzeitig mit der Implementierung des Systems und dem Komponententest anfällt. Die Testrealisierungsarbeiten des Systemtests dauert dagegen oft bis wenige Tage vor Beginn der Testdurchführung.
- Die Testdurchführung beginnt, wenn alle Eingangskriterien für die Durchführung des Systemtests erfüllt sind (oder auf einige davon verzichtet wird), was normalerweise bedeutet, dass mindestens der Komponententest und oft auch der Komponentenintegrationstest ihre jeweiligen Endekriterien erreicht haben bzw. Definitions of Done erfüllt haben. Die Testdurchführung dauert, bis die Endekriterien des Systemtests erfüllt sind.
- Die Testabschlussaktivitäten des Systemtests erfolgen, nachdem die Endekriterien des Systemtests erfüllt sind.

Bei iterativen und inkrementellen Modellen werden die Aktivitäten möglicherweise nicht in der gleichen Reihenfolge ausgeführt, oder es werden einzelne Aktivitäten ganz weggelassen. Bei einem iterativen Modell könnte beispielsweise in jeder Iteration eine reduzierte Menge an Testaktivitäten verwendet werden. Testanalyse, Testentwurf, Testrealisierung und –durchführung erfolgen hier in jeder Iteration, während die Grobplanung zu Beginn und die Abschlussaktivitäten am Ende des Projektes erfolgen.

In agilen Projekten wird oft ein weniger formalisierter Prozess angewendet und eine deutlich engere Zusammenarbeit der Stakeholder praktiziert. So können Änderungen im Projekt einfacher erfolgen. Möglicherweise gibt es keine klar definierte Rolle für Test Analysten. Es gibt weniger umfassende Testdokumentation, und die Kommunikation ist kürzer und häufiger.

Bei agilen Projekten wird von Anfang an getestet. Dies beginnt mit der Initiierung des Projekts, wenn die Entwickler einen ersten Architektur- und Systementwurf erstellen. Reviews müssen nicht formal erfolgen, sondern werden fortlaufend im Zuge der Softwareentwicklung durchgeführt. Es wird erwartet, dass die Beteiligung während des gesamten Projekts erfolgt und dass die Aufgaben des Test Analysten vom Team übernommen werden.

Iterative und inkrementelle Modelle reichen von agilen Ansätzen, bei denen Änderungen im Laufe der Softwareentwicklung antizipiert werden, bis hin zu iterativ/inkrementellen Softwareentwicklungsmodellen innerhalb eines V-Modells (diese werden manchmal auch als eingebettet-iterativ bezeichnet). Bei eingebettet-iterativen Modellen ist zu erwarten, dass der Test Analyst an den Planungs- und Entwurfsaktivitäten beteiligt ist, dann aber während des Entwurfs, der Entwicklungund des Testens der Software eine interaktivere Rolle übernimmt.

Unabhängig vom eingesetzten Softwareentwicklungslebenszyklusmodell ist es wichtig, dass der Test Analyst versteht, welche Erwartungen bezüglich seiner Mitarbeit bestehen, und wann diese erfolgen sollte. In der Praxis kommen viele Mischformen zum Einsatz, wie beispielsweise das iterative Modell innerhalb des V-Modells wie bereits erwähnt. Test Analysten sollten ihre effektivste Rolle bestimmen und auf den richtigen Zeitpunkt der Beteiligung hinarbeiten, anstatt sich auf ein vordefiniertes, festgelegtes Rollenmodell zu verlassen.

1.3 Testanalyse

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

- Analyse der Testbasis
- Identifizierung von Fehlerzuständen verschiedener Art in der Testbasis
- Identifizierung und Priorisierung der Testbedingungen und der zu testenden Merkmale
- Erfassung der bidirektionalen Verfolgbarkeit zwischen jedem Element der Testbasis und den zugehörigen Testbedingungen
- Ausführen von Aufgaben im Zusammenhang mit risikobasiertem Testen (siehe Kapitel 2)

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

- Es liegen Informationen vor (z.B. Anforderungen, User Stories), in denen das Testobjekt beschrieben ist, und die als Testbasis dienen können (siehe [ISTQB_FL_SYL] Abschnitt 1.4.2 für eine Liste anderer möglicher Quellen der Testbasis).
- Die Testbasishat das Review mit brauchbarem Ergebnis bestanden und wurde wie benötigt aktualisiert. Es ist zu beachten, dass für die Definition von abstrakten Testfällen (siehe Abschnitt 1.4.1) die Testbasis noch nicht zwingend vollständig definiert sein muss. In einem agilen Projekt wird dieser Reviewzyklus iterativ sein, da die User Stories zu Beginn der Iterationen verfeinert werden.
- Für die Durchführung der restlichen Testaktivitäten zum vorliegenden Testobjekt steht ein genehmigtes Budget und ein Zeitplan zur Verfügung.

Testbedingungen werden typischerweise durch eine Analyse der Testbasis in Verbindung mit den Testzielen (wie in der Testplanung definiert) identifiziert. Falls die Dokumentation veraltet ist oder es keine Dokumentation gibt, können die Testbedingungen in Gesprächen mit den relevanten Stakeholdern identifiziert werden (z.B. in Workshops oder bei der Iterationsplanung). In agilen Projekten werden die Abnahmekriterien, die im Rahmen von User Stories definiert werden, oft als Grundlage für den Testentwurf verwendet.

Während die Testbedingungen gewöhnlich spezifisch für das betrachtete Testelement 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 für den Test definieren, wie z.B. “Nachweisen, dass Bildschirmmaske X funktioniert”. Danach werden detailliertere Bedingungen als Basis für spezifische Testfälle identifiziert, wie z.B. “Nachweisen, dass Bildschirmmaske 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. Dieser Ansatz ermöglicht es dem Test Analysten auch, mit der Definition von abstrakten Testbedingungen für User Stories zu beginnen, die noch nicht verfeinert wurden.
- Falls Produktrisiken definiert wurden, müssen die Testbedingungen identifiziert werden, die die jeweiligen Risiken adressieren und sie müssen zum jeweiligen Risiko verfolgbar sein.

Die Anwendung von Testverfahren (wie sie in der Teststrategie und/oder im Testkonzept festgelegt sind) kann für den Prozess der Testanalyse hilfreich sein und kann zur Unterstützung folgender Ziele eingesetzt werden:

- Identifizierung von Testbedingungen
- Reduzierung der Wahrscheinlichkeit, wichtige Testbedingungen auszulassen
- Definieren von präziseren und treffenderen Testbedingungen
- Nachdem die Testbedingungen identifiziert und verfeinert wurden, kann ein Review dieser Bedingungen mit den Stakeholdern durchgeführt werden, um sicherzustellen, dass die

Anforderungen klar verstanden wurden und dass das Testen auf die Ziele des Projekts abgestimmt ist.

Nach Abschluss der Testanalyse für einen bestimmten Bereich (z.B. für eine bestimmte Funktion) sollte der Test Analyst wissen, welche spezifischen Tests für diesen Bereich entworfen werden müssen.

1.4 Testentwurf

Auch der Testentwurf muss sich an dem in der Testplanung bestimmten Umfang orientieren. Der Test Analyst entwirft daraufhin die Tests, die im weiteren Verlauf des Testprozesses realisiert und durchgeführt werden. Der Testentwurf beinhaltet die folgenden Aktivitäten:

- Bestimmen, in welchen Bereichen konkrete und abstrakte Testfälle am besten geeignet sind.
- Die Testverfahren festlegen, die es ermöglichen, die benötigte Überdeckung zu erreichen. Die in Frage kommenden Testverfahren werden bei der Testplanung festgelegt.
- Durch Anwendung der Testverfahren Testfälle und Gruppen von Testfällen entwerfen, die die identifizierten Testbedingungen überdecken.
- Die notwendigen Testdaten zur Unterstützung von Testbedingungen und Testfällen identifizieren.
- Die Testumgebung entwerfen und die erforderliche Infrastruktur einschließlich Werkzeugen identifizieren.
- Die bidirektionale Verfolgbarkeit erfassen (z.B. zwischen Testbasis, Testbedingungen und Testfällen)

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

Während des Testentwurfs muss der Test Analyst mindestens die folgenden Punkte berücksichtigen:

- Für manche Testelemente ist es besser nur die Testbedingungen zu spezifizieren, anstatt detaillierte Testskripte zu definieren, die die Abfolge von Anweisungen für die Durchführung von Tests vorgeben. In diesen Fällen sollten die Testbedingungen so spezifiziert werden, dass sie als Leitfaden für ungeskriptete Tests verwendet werden können.
- Die bestanden/nicht bestanden-Kriterien 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 in der Lage sein, die zuvor spezifizierten Tests zu lesen und zu 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 unter Umständen am Review der Tests beteiligt sind, und für die Auditoren, die die Tests eventuell genehmigen müssen.
- Tests sollten sämtliche Interaktionen mit dem Testobjekt überdecken, nicht nur die Interaktionen mit Nutzern über die sichtbare Benutzungsschnittstelle. Hierzu gehören beispielsweise auch Interaktionen mit anderen Systemen und technische oder physikalische Ereignisse (siehe [IREB_CPRE] für weitere Details).
- Tests sollten so entworfen werden, dass sie die Schnittstellen zwischen den verschiedenen Testobjekten sowie das Verhalten der Objekte selbst testen.
- Der Aufwand für den Testentwurf muss priorisiert und ausgewogen sein, und sich an den Risikostufen und dem Geschäftswert orientieren.

1.4.1 Konkrete und abstrakte Testfälle

Eine Aufgabe des Test Analysten ist es, das für die jeweiliege Situation beste Abstraktionsniveau für die Testfällen zu bestimmen. Konkrete und abstrakte Testfälle werden in [ISTQB_FL_SYL] behandelt. Einige der Vor- und Nachteile der Verwendung von konkreten und abstrakten Testfällen sind nachfolgend aufgelistet:

Konkrete Testfälle bieten folgende Vorteile:

- Testpersonal mit wenig Erfahrung kann sich auf detaillierte Informationen innerhalb des Projekts verlassen. Konkrete Testfälle liefern sämtliche spezifischen Informationen und Vorgehensweisen (einschließlich der benötigten Daten), die der Tester zur Durchführung des Testfalls und zur Verifizierung der Ergebnisse benötigt.
- Tests sind reproduzierbar, d.h., andere Tester können die Test erneut ausführen und erzielen dabei die gleichen Ergebnisse.
- Nicht offensichtliche Fehlerzustände in der Testbasis können aufgedeckt werden.
- Der Detaillierungsgrad ermöglicht bei Bedarf eine unabhängige Verifizierung der Tests, z.B. durch ein Audit.
- Der Zeitaufwand für die Realisierung der automatisierten Testfälle kann reduziert werden.

Konkrete Testfälle können folgende Nachteile haben:

- Sie können einen erheblichen Aufwand sowohl bei der Erstellung als auch bei der Wartung erfordern.
- Sie können den kreativen Gestaltungsspielraum des Testers bei der Testdurchführung einengen.
- Sie setzen voraus, dass die Testbasis gut spezifiziert ist.
- Ihre Verfolgbarkeit zu den Testbedingungen kann einen höheren Aufwand bedeuten als bei abstrakten Testfällen.

Abstrakte Testfälle bieten folgende Vorteile:

- Sie liefern eine Richtlinie und spezifizieren, was getestet werden soll; sie stellen dem Test Analysten jedoch frei, die tatsächlichen Daten und sogar den Ablauf für die Durchführung des Tests zu variieren.
- Sie können eine bessere Risikoüberdeckung liefern als konkrete Testfälle, weil sie bei jeder Ausführung ein wenig variiert werden.
- Sie können schon früh im Anforderungsprozess spezifiziert werden.
- Sie nutzen die Testerfahrung als auch die Erfahrung mit dem Testobjekt des Test Analysten, der die Tests durchführt.
- Sie können auch dann definiert werden, wenn keine detaillierte und formale Dokumentation benötigt wird.
- Sie sind besser geeignet für eine Wiederverwendung, wenn in verschiedenen Testzyklen unterschiedliche Testdaten verwendet werden.

Abstrakte Testfälle haben die folgenden Nachteile:

- Sie sind weniger reproduzierbar, was die Verifizierung erschwert. Dies liegt daran, dass eine detaillierte Beschreibung fehlt, wie sie bei konkreten Testfällen vorhanden ist.
- Für die Testausführung kann erfahreneres Testpersonal erforderlich sein.
- Werden abstrakte Testfälle als Basis für die Automatisierung herangezogen, können die fehlenden Details dazu führen, dass die falschen Ergebnisse validiert werden, bzw. dass Elemente nicht validiert werden, die eigentlich validiert werden sollten.

Abstrakte 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 abstrakten zu den konkreten Testfällen, wobei lediglich die konkreten Testfälle ausgeführt werden.

1.4.2 Testfälle entwerfen

Testfälle werden durch die schrittweise Ausarbeitung und Verfeinerung der identifizierten Testbedingungen mit Hilfe von Testverfahren (siehe Kapitel 3) entworfen. Testfälle sollten wiederholbar, verifizierbar und zur Testbasis (z.B. Anforderungen) verfolgbar sein.

Der Testentwurf beinhaltet dabei die Identifizierung von:

- Zielsetzung (d.h. das beobachtbare, messbare Ziel der Testdurchführung)
- Vorbedingungen, wie z.B. Anforderungen an projekteigene oder lokalisierte Testumgebungen und deren geplante Bereitstellung, der Zustand des Systems vor Testausführung usw.
- Anforderungen an die Testdaten (sowohl Eingabedaten für den Testfall als auch Daten, die im System vorhanden sein müssen, damit der Testfall durchgeführt werden kann)
- erwarteten Ergebnissen mit expliziten bestanden/nicht bestanden-Kriterien
- Nachbedingungen, wie z.B. betroffene Daten, Zustand des Systems nach Testausführung, Auslöser für nachfolgende Prozessabläufe usw.

Oft stellt die Definition der erwarteten Ergebnisse eine besondere Herausforderung dar. Die manuelle Berechnung des Testeregbnisses ist häufig mühsam und fehleranfällig. Es gilt, wenn möglich, ein automatisiertes Testorakel zu finden oder zu erstellen. Bei der Identifizierung der erwarteten Ergebnisse beschäftigen sich die Tester nicht nur mit Bildschirmausgaben, sondern auch mit Daten und umgebungsspezifische Nachbedingungen. Wenn die Testbasis klar definiert ist, sollte die Identifizierung der korrekten Ergebnisse theoretisch einfach sein. In der Praxis ist die Testbasis jedoch oft vage oder widersprüchlich, deckt Schlüsselbereiche nicht ab, oder gar nicht vorhanden. In solchen Fällen muss der Test Analyst entweder selbst fachliche Expertise haben oder zumindest auf sie zurückgreifen können. Auch wenn die Testbasis gut spezifiziert ist, können komplexe Interaktionen zwischen verschiedenen Stimuli und ausgelösten Reaktionen die Definition der erwarteten Testergebnisse verkomplizieren. Ein Testorakel ist daher unverzichtbar. In agilen Projekten kann das Testorakel der Product Owner sein. Die Durchführung von Testfällen ohne Möglichkeit, die Richtigkeit der Ergebnisse zu überprüfen, hat nur einen sehr begrenzten Merhwert, denn daraus können sich ungültige Fehlerberichte oder unbegründetes Vertrauen in das System ergeben.

Die oben beschriebenen Aktivitäten lassen sich in allen Teststufen anwenden, nur die Testbasis ändert sich. Bei der Analyse und beim Entwurf von Tests ist es wichtig, die Stufe, auf die der Test abzielt, sowie das Ziel des Tests zu berücksichtigen. Dies hilft dabei, den benötigten Detaillierungsgrad und die Werkzeuge zu bestimmen, die möglicherweise benötigt werden (z.B. Treiber und Platzhalter für die Komponententeststufe).

Bei der Entwicklung der Testbedingungen und Testfälle werden in der Regel verschiedene Dokumente erstellt, was zu verschiedenen Arbeitsergebnissen führt. In der Praxis gibt es jedoch bei der Dokumentation von Arbeitsergebnissen große Unterschiede hinsichtlich des Umfangs und des Detaillierungsgrads. 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 regulatorische Vorschriften, die eingehalten werden müssen
- das Softwareentwicklungslebenszyklusmodell, das zum Einsatz kommt (bei agilen Methoden soll der Umfang der Dokumentation gerade ausreichend sein)
- den Bedarf an Verfolgbarkeit von der Testbasis über die Testanalyse bis hin zum Testentwurf

Je nach Testumfang können sich die Testanalyse- und Testentwurfsaktivitäten auch mit den Qualitätsmerkmalen des Testobjekts bzw. der Testobjekte befassen. Der ISO-Standard 25010

[ISO25010] ist desibezüglich eine nützliche Referenz. 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 bereits eine Form des statischen Testens, da durch diesen Prozess bereits Probleme in den Dokumenten der Testbasis gefunden werden können. Die Testanalyse und der Testentwurf basierend auf der Anforderungsspezifikation sind beispielsweise eine ausgezeichnete Vorbereitung auf die Anforderungsreviewsitzung . Das Lesen von Anforderungenzum Zwecke der Erstellung von Tests setzt voraus, dass die Anforderungen verstanden werden und dass bewertet werden kann, ob die Anforderungen erfüllt sind. Häufig werden bei dieser Aktivität Anforderungen entdeckt, die unklar sind, oder die nicht testbar sind, oder für die keine Abnahmekriterien definiert sind. Gleichermaßen können auch Arbeitsergebnisse wie Testfälle, Risikoanalysen und Testkonzepte einem Review unterzogen werden.

Während des Testentwurfs lassen sich die detaillierten Anforderungen an die Testinfrastruktur definieren. In der Praxis werden sie aber in der Testrealisierung endgültig festgelegt. Hinweis: Zur Testinfrastruktur gehört mehr als nur Testobjekte und Testmittel. So umfassen Anforderungen an die Testinfrastruktur auch Räumlichkeiten, Ausrüstungen, Personal, Software, Werkzeuge, Peripheriegeräte, Kommunikationseinrichtungen, Zugangsberechtigungen, und alles was sonst zum Durchführen des Tests nötig ist.

Die Endekriterien von Testanalyse und Testentwurf variieren je nach den Projektparametern, aber alle in diesen beiden Abschnitten behandelten Punkte sollten für eine Aufnahme in die definierten Endekriterien in Erwägung gezogen werden. Es ist zu beachten, dass die Endekriterien messbar sind, und dass sie sicherstellen, dass alle Informationen und Vorbereitungen für die nachfolgenden Schritte zur Verfügung gestellt wurden.

1.5 Testrealisierung

Bei der Testrealisierung werden auf der Grundlage von Testanalyse und Testentwurf die für die Testdurchführung erforderlichen Testmittel vorbereitet. Dies umfasst die folgenden Aktivitäten:

- Erstellen eines Testausführungsplans einschließlich Ressourcenzuweisung, damit mit der Testfallausführung beginnen kann (siehe [ISTQB_FL_SYL] Abschnitt 5.2.4)
- Zusammenstellen von (manuellen und automatisierten) Tests in Testsuiten und Festlegen der Reihenfolge ihrer Durchführung
- Erstellen automatisierter Tests (bzw. identifizieren der Testfälle, die von einem Entwickler oder Testautomatisierungsentwickler zu automatisieren sind)
- Entwickeln der Testabläufe
- Testdaten und Testumgebungen festlegen
- Aktualisieren der Verfolgbarkeit zwischen der Testbasis und den Testmitteln wie Test-bedingungen, Testfällen und Testsuiten.

Während der Testrealisierung identifizieren die Test Analysten Testfälle, die gruppiert werden können (z.B. Testfälle, die sich alle auf das Testen eines bestimmten übergeordneten Geschäftsprozesses beziehen) und organisieren diese in Testsuiten. So können zusammengehörige Testfälle miteinander ausgeführt werden.

Es werden Testabläufe erstellt, die die Reihenfolge endgültig festgelegen und bestätigen, in der die manuellen und automatisierten Tests und Testsuiten ausgeführt werden. Die Definition der Testabläufe erfordert die sorgfältige Identifizierung von Einschränkungen und Abhängigkeiten, die die Reihenfolge der Testausführung beeinflussen können. In der Testablaufspezifikation sind außerdem alle Vorbedingungen dokumentiert (z.B. Laden von Testdaten aus einem Datenspeicher) und alle

Aktivitäten nach der Testausführung (z.B. Zurücksetzen des Systemzustands). Falls eine risikobasierte Teststrategie angewendet wird, wird die Reihenfolge, in der die Testfälle ausgeführt werden, mitunter durch die Risikostufe zwingend vorgeben. Darüber hinaus gibt es noch weitere Faktoren, die die Reihenfolge bestimmen, wie z.B. die Verfügbarkeit der richtigen Personen, Ausrüstungen, Daten und der zu testenden Funktionalität.

Es ist nicht ungewöhnlich, dass Code schrittweise freigegeben wird und der Testaufwand mit der Reihenfolge abgestimmt werden muss, in der die Software zum Testen bereitgestellt wird. Insbesondere bei iterativen und inkrementellen Lebenszyklusmodellen muss der Test Analyst den Ablauf mit dem Entwicklungsteam koordinieren, um sicherzustellen, dass die Software in einer testbaren Reihenfolge zum Testen freigegeben wird.

Die oben genannten Faktoren müssen bei der Erstellung eines Testausführungsplans berücksichtigt werden.

Der in der Testrealisierung notwendige Detaillierungsgrad und die damit verbundene Komplexität der Aufgaben können durch den Detaillierungsgrad der Testfälle und der Testbedingungen beeinflusst werden. In manchen Fällen gelten gesetzliche Vorschriften, und die Arbeitsergebnisse des Testens müssen den Nachweis erbringen, dass sie konform zu geltenden Normen und Standardssind, wie beispielsweise die US-amerikanische Norm DO-178C (in Europa ED 12C). [RTCA DO-178C/ED-12C].

Wie oben beschrieben, werden meist Testdaten für das Testen benötigt, und in einigen Fällen, können diese die Datenmengen sehr umfangreich sein. Während der Testrealisierung erzeugen Test Analysten Eingabe- und Umgebungsdaten, die in Datenbanken und anderen Repositorien gespeichert werden. Diese Daten müssen zweckdienlich sein, um Fehlerzustände erkennen zu können. Test Analysten können auch Daten erzeugen, die daten- und schlüsselwortgetriebenes automatisiertes Testen (siehe Abschnitt 6.2), sowie für manuelle Tests verwendet werden.

Die Testrealisierung befasst sich außerdem mit der Testumgebung bzw. den Testumgebungen. Während dieser Aktivität sollte die Umgebung (bzw. die Umgebungen) vollständig eingerichtet und verifiziert werden, bevor die Testdurchführung beginnt. Eine zweckdienliche Testumgebung ist unerlässlich, d.h. die Testumgebung sollte so beschaffen sein, dass sich vorhandene Fehlerzustände beim Testen aufdecken lassen, sie sollte normal funktionieren, wenn keine Fehlerwirkungen auftreten; und schließlich sollte sie bei Bedarf in höheren Teststufen die Produktions- oder Anwendungsumgebung angemessen abbilden. Änderungen an der Testumgebung während der Testdurchführung können aufgrund von unvorhergesehenen Änderungen, Testergebnissen oder aus sonstigen Erwägungen notwendig werden. Falls derartige Änderungen während der Testdurchführung anfallen, muss bewertet werden, wie sich diese auf die bereits durchgeführten Tests auswirken.

In der Testrealisierung müssen die Test Analysten sicherstellen, dass die für Erstellung und Wartung der Testumgebung zuständigen Personen bekannt und verfügbar sind, und dass die Testmittel und Werkzeuge zur Unterstützung des Testens und die damit verbundenen Prozesse einsatzbereit sind. Dazu gehören Konfigurationsmanagement, Fehlermanagement sowie Testprotokollierung und Testmanagement. Darüber hinaus müssen Test Analysten die Verfahren verifizieren, mit denen die Daten zur Bewertung des aktuellen Status bezüglich der Endekriterien und für Berichte über die Testergebnisse gesammelt werden.

Für die Testrealisierung empfiehlt es sich, einen ausgewogenen Ansatz zu verwenden, der bei der Testplanung festgelegt wurde. Beispielsweise werden risikobasierte, analytische Teststrategien oft mit reaktiven Teststrategien kombiniert. In diesem Fall wird ein Teil des Testrealisierungsaufwands für das Testen ohne vorgegebene Testskripte aufgewendet.

Testen ohne Testskripte sollte allerdings nicht zufällig oder ziellos erfolgen, da der Zeitaufwand und die Überdeckung schwer einzuschätzen sind, und dies zu einer geringen Fehlerfindung führt. Vielmehr

sollte dies in zeitlich begränzten Sitzungen erfolgen, die zwar jeweils durch eine Charta gelenkt sind, jedoch mit der Freiheit, von den Vorschriften der Charta abzuweichen, wenn im Laufe der Sitzung potenziell produktivere Testmöglichkeiten entdeckt werden. Im Laufe der Zeit haben Tester eine Vielzahl von erfahrungsbasierten Testverfahren entwickelt, wie z.B. Fehlerangriffe [Whittaker03], intuitive Testfallermittlung [Myers11] und exploratives Testen [Whittaker09]. Dabei finden Testanalyse, Testentwurf und Testrealisierung immer noch statt, allerdings vorwiegend während der Testdurchführung.

Wenn solche reaktiven Teststrategien verfolgt werden, dann beeinflussen die Ergebnisse jedes Tests die Analyse, den Entwurf und die Realisierung der nachfolgenden Tests. Wenngleich diese Teststrategien“leichtgewichtig” und oft effektiv bei der Fehlerfindung sind, gibt es aber auch einige Nachteile, darunter die folgenden:

- Erfahrung von Test Analysten ist erforderlich
- die Dauer kann schwer vorhersagbar sein
- die Überdeckung kann schwer zu verfolgen sein
- die Wiederholbarkeit der Tests kann ohne gute Dokumentation oder Werkzeugunterstützung problematisch sein

1.6 Testdurchführung

Die Testdurchführung erfolgt gemäß des Testausführungsplans und umfasst folgende Aufgaben: (siehe [ISTQB_FL_SYL])

- Ausführen von manuellen Tests, einschließlich explorativer Tests
- Ausführen von automatisierten Tests
- Vergleichen der tatsächlichen Ergebnisse mit den erwarteten Ergebnissen
- Analysieren von Anomalien zur Feststellung ihrer wahrscheinlichen Ursachen
- Berichten von Fehlerzuständen aufgrund der beobachteten Fehlerwirkungen
- Protokollieren der Ergebnisse der Testausführung
- Aktualisieren der Verfolgbarkeit zwischen der Testbasis und den Testmitteln zur Berücksichtigung der Testergebnisse
- Ausführen von Regressionstests

Die oben aufgeführten Aufgaben der Testdurchführung können entweder von Testern oder von Test Analysten durchgeführt werden.

Im Folgenden sind typische zusätzliche Aufgaben aufgeführt, die vom Test Analysten ausgeführt werden können:

- Erkennen von Häufungen von Fehlerzuständen, die ein Hinweis sein können, dass für einen bestimmten Teil des Testobjekts mehr Tests notwendig sind
- Vorschläge für künftige explorative Testsitzungen auf Grundlage der Ergebnisse der explorativen Tests machen
- Identifizieren neuer Risiken aus den Informationen, die im Laufe der Testdurchführung gewonnen wurden
- Vorschläge zur Verbesserung von Arbeitsergebnissen der Testrealisierungsaktivitäten machen (z.B. Verbesserungen bei den Testabläufen)

2. Die Aufgaben des Test Analysten beim risikobasierten Testen - 60 min

Schlüsselbegriffe

Produktrisiko, Risikoidentifizierung, Risikominderung, risikobasiertes Testen, Risikostufe

Lernziele für die Aufgaben des Test Analysten beim risikobasierten Testen

Die Aufgaben des Test Analysten beim risikobasierten Testen

TA-2.1.1 (K3) Für eine vorgegebene Situation an der Risikoidentifizierung mitwirken, eine Risikobewertung durchführen und geeignete Maßnahmen zur Risikominderung vorschlagen

2.1 Einführung

Häufig tragen Testmanager die Gesamtverantwortung dafür, eine risikobasierte Teststrategie aufzusetzen und zu verwalten. In der Regel werden sie jedoch einen Test Analysten daran beteiligen, um sicherzustellen, dass die risikobasierte Testvorgehensweise korrekt implementiert wird.

Test Analysten sollten an den folgenden risikobasierten Testaufgaben aktiv mitwirken:

- Risikoidentifizierung
- Risikobewertung
- Risikominderung

Diese Aufgaben werden im gesamten Softwareentwicklungslebenszyklus iterativ durchgeführt um sich mit neu auftretenden Risiken, sich ändernden Prioritäten und mit der regelmäßigen Bewertung und Kommunikation des Risikostatus (weitere Details, siehe [vanVeenendaal12] und [Black02]) zu befassen. In agilen Projekten werden die drei Aufgaben oft in einer sogenannten Risikositzung kombiniert, die sich entweder mit einer Iteration oder einem Release befasst.

Test Analysten sollten innerhalb des vom Testmanager für das Projekt festgelegten risikobasierten Testrahmens arbeiten. Sie sollten ihr Wissen über die Risiken des Geschäftsbereichs einbringen, wie z.B. Risiken in Bezug auf die Betriebssicherheit, geschäftlichen und wirtschaftlichen Aspekte sowie politischen Faktoren.

2.2 Risikoidentifizierung

Durch die Einbindung einer möglichst breiten Auswahl an Stakeholdern wird der Prozess der Risikoidentifizierung die größtmögliche Menge an signifikanten Risiken höchstwahrscheinlich aufdecken.

Test Analysten verfügen häufig über spezifisches Wissen über den jeweiligen Geschäftsbereich des Systems unter Test. Dies bedeutet, dass sie für die Ausführung der folgenden Aufgaben besonders gut geeignet sind:

- Durchführen von Experten-Interviews mit Experten des Geschäftsbereichs und Benutzern
- Durchführen unabhängiger Bewertungen
- Anwenden von Risikovorlagen
- Teilnehmen an Risiko-Workshops
- Teilnehmen an Brainstorming-Sitzungen mit potenziellen und aktuellen Benutzern
- Definieren von Checklisten für das Testen
- Rückgriff auf Erfahrungen mit ähnlichen Systemen oder Projekten in der Vergangenheit

Insbesondere sollte der Test Analyst eng mit den Benutzern und anderen Experten des Geschäftsbereichs (z.B. Anforderungsingenieuren, Businessanalysten) zusammenarbeiten, um die Bereiche mit Geschäftsrisiken zu bestimmen, die beim Testen adressiert werden sollten. In agilen Projekten ermöglicht diese enge Beziehung zu den Stakeholdern eine regelmäßige Risikoidentifizierung, z.B. während der Iterationsplanungs-Sitzungen.

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

- Probleme mit der funktionalen Korrektheit, z.B. falsche Berechnungen
- Probleme mit der Gebrauchstauglichkeit, z.B. zu wenig Tastaturkürzel
- Probleme mit der Übertragbarkeit, z.B. dass die Anwendung auf bestimmten Plattformen nicht installiert werden kann

2.3 Risikobewertung

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

Zur Bestimmung der Risikostufe wird normalerweise für jedes einzelne Risiko die Eintrittswahrscheinlichkeit und das Schadensausmaß bewertet. Die Eintrittswahrscheinlichkeit wird gewöhnlich als die Wahrscheinlichkeit interpretiert, dass das potenzielle Problem im System unter Test vorhanden ist und auch beobachtet wird, wenn das System in Produktion ist. Technische Test Analysten sollten dazu beitragen, die potenzielle Eintrittswahrscheinlichkeit für jedes einzelne Risiko festzulegen und zu verstehen. Test Analysten sollten andererseits dazu beitragen, die potenziellen geschäftlichen Auswirkungen eines Problems zu verstehen, falls dieses auftritt (in agilen Projekten kann diese rollenbasierte Unterscheidung weniger stark ausgeprägt sein).

Das Schadensausmaß bei Eintreten eines Risikos wird oft als das Schwere der Auswirkung auf die Benutzer, Kunden oder andere Stakeholder interpretiert. Es entsteht also aus einem Geschäftsrisiko. Der Test Analyst sollte dazu beitragen, die potenziellen Auswirkungen der einzelnen Risiken auf den Geschäftsbereich oder die Benutzer zu identifizieren und zu bewerten. Folgende Faktoren beeinflussen die Geschäftsrisiken:

- Häufigkeit der Nutzung der betroffenen Feature
- entgangene Geschäfte
- finanzielle Verluste
- ökologische oder soziale Verluste oder Haftungsansprüche
- zivil- oder strafrechtliche Maßnahmen
- Bedenken zur funktionalen Sicherheit
- Geldstrafen, Aberkennung von Lizenzen
- keine vernünftigen Lösungsalternativen, falls Personen nicht weiterarbeiten können
- Sichtbarkeit des Features
- das negative Erscheinungsbild, wenn Mängel bekannt werden, Negativschlagzeilen, Schaden für die Reputation (Imageschaden)
- Verlust von Kunden

Anhand der vorhandenen Informationen muss der Test Analyst die Risikostufen für die Geschäftsrisiken basierend auf den vom Testmanager vorgegebenen Richtlinien festlegen. Diese können mit Begriffen (z.B. gering, mittelhoch, hoch) oder mit Zahlen und/oder Farben klassifiziert werden.

Solange es nicht möglich ist ein Risiko objektiv anhand einer definierten Skala zu messen, kann die Messung nicht als eine echte quantitative Messung angesehen werden. Zwar können den qualitativen Werten Zahlen zugeordnet werden, aber dies macht die Messung noch nicht zu einer echten quantitativen Messung. Beispielsweise können Testmanager festlegen, das Geschäftsrisiko anhand einer bestimmten numerischen Skala zu kategorisieren (z.B. 1-5 für Eintrittswahrscheinlichkeit und Schadensausmaß). Nachdem Eintrittswahrscheinlichkeit und Schadensausmaß bewertet wurden, lässt sich anhand dieser Werte das Gesamtrisiko für jedes einzelne Risiko bestimmen. Diese Einstufung wird dann für die Priorisierung der Risikominderungsaktivitäten verwendet. [vanVeenendaal12].

2.4 Risikominderung

Im Projekt sollten die Test Analysten folgende Aufgaben übernehmen:

- Das Produktrisiko mindern indem sie geeignete Testfälle entwerfen, die eindeutig belegen, ob die Tests bestehen oder nicht, sowie an Reviews von Softwarearbeitsergebnissen teilnehmen (wie z.B. Anforderungen, Entwürfe und Benutzerdokumentation)
- Geeignete Risikominderungsaktivitäten umsetzen, die in der Teststrategie und im Testkonzept festgelegt sind (z.B. spezielle Testverfahren zum Testen eines Geschäftsprozesses mit besonders hohem Risiko)
- Bekannte Risiken neu bewerten, wenn im Laufe des Projekts neue Informationen gesammelt werden, und anhand dieser die Eintrittswahrscheinlichkeit, das Schadensausmaß oder beides gegebenenfalls anpassen
- Neue Risiken aus den Informationen identifizieren, die während des Testens gewonnenen wurden

Im Hinblick auf Produktrisiken trägt Testen wesentlich dazu bei, diese Risiken zu mindern. Wenn die Tester Fehler finden, mindern sie das Risiko, weil sie das Bewusstsein für vorhandene Fehler schärfen und die Möglichkeit schaffen, diese vor der Systemfreigabe zu beheben. Wenn die Tester keine Fehler finden, dann mindert das Testen das Risiko, indem es den Nachweis erbringt, dass das System unter bestimmten (d.h. den getesteten) Bedingungen richtig funktioniert. Test Analysten wirken bei der Bestimmung von möglichen Maßnahmen zur Risikominderung mit, indem sie u.a. Möglichkeiten zur Sammlung genauer Testdaten untersuchen, realistische Szenarien erstellen und testen, und Gebrauchstauglichkeitsstudien durchführen oder leiten.

2.4.1 Tests priorisieren

Die Risikostufe wird auch für die Priorisierung der Tests verwendet. Zum Beispiel: Ein Test Analyst stellt fest, dass ein hohes Risiko im Bereich der Transaktionsgenauigkeit in einem Buchhaltungssystem besteht. Um dieses Risiko zu mindern, 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 Gebrauchstauglichkeit ein bedeutendes Risiko für ein neues Testobjekt sind. Anstatt abzuwarten, bis die Probleme beim Benutzerabnahmetest entdeckt würden, weist der Test Analyst einem auf Grundlage eines Prototyps frühzeitig durchgeführten Gebrauchstauglichkeitstest eine hohe Priorität zu. So sollen Probleme mit der Gebrauchstauglichkeit wesentlich früher identifiziert und behoben werden als im Benutzerabnahmetest. 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 Risiken ("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 risikobasierten Testen in die Tiefe oder in die Breite getestet wird, ist es möglich, dass die verfügbare Zeit für das Testen abläuft, ohne dass alle Tests durchgeführt wurden. Beim risikobasierten Testen können die Tester in solchen Fällen das Management über das zu diesem Zeitpunkt verbleibende Restrisiko informieren, und das Management kann entscheiden, ob weiter getestet werden soll, oder ob das Restrisiko an die Benutzer, Kunden, Helpdesks bzw. technischen Support und/oder an das Betriebspersonal weitergegeben wird.

2.4.2 Anpassung des Testens für weitere Testzyklen

Die Risikobewertung ist keine einmalige Aktivität, die vor Beginn der Testrealisierung erfolgt. Sie ist 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 Fehlerzustände
- typische Fehlerzustände, die beim Testen gefunden wurden
- Bereiche, die nicht ausreichend getestet wurden (geringe Überdeckung)

3. Testverfahren - 630 min

Schlüsselbegriffe

anwendungsfallbasierter Test, Äquivalenzklassenbildung, Black-Box-Testverfahren, checklistenbasiertes Testen, Entscheidungstabellentest, erfahrungsbasiertes Testen, erfahrungsbasiertes Testverfahren , exploratives Testen, fehlerbasiertes Testverfahren, Fehlertaxonomie, Grenzwertanalyse, intuitive Testfallermittlung, Klassifikationsbaumverfahren, paarweises Testen, Test-Charta, Zustandsübergangstest

Lernziele zum Thema Testverfahren

3.1 Einführung

Keine Lernziele

3.2 Black-Box-Testverfahren

TA-3.2.1 (K4) Eines oder mehrere vorgegebene Spezifikationselemente analysieren und unter Verwendung der Äquivalenzklassenbildung Testfälle entwerfen

TA-3.2.2 (K4) Eines oder mehrere vorgegebene Spezifikationselemente analysieren und unter Verwendung der Grenzwertanalyse Testfälle entwerfen

TA-3.2.3 (K4) Eines oder mehrere vorgegebene Spezifikationselemente analysieren und unter Verwendung des Entscheidungstabellentests Testfälle entwerfen

TA-3.2.4 (K4) Eines oder mehrere vorgegebene Spezifikationselemente analysieren und unter Verwendung des Zustandsübergangstests Testfälle entwerfen

TA-3.2.5 (K2) Erläutern, wie Klassifikationsbaumdiagramme Testverfahren unterstützen

TA-3.2.6 (K4) Eines oder mehrere vorgegebene Spezifikationselemente analysieren und unter Verwendung des paarweisen Testens Testfälle entwerfen

TA-3.2.7 (K4) Eines oder mehrere vorgegebene Spezifikationselemente analysieren und unter Verwendung des anwendungsfallbasierten Tests Testfälle entwerfen

TA-3.2.8 (K4) Ein System oder dessen Anforderungsspezifikation analysieren, um zu bestimmen, welche Fehlerarten wahrscheinlich gefunden werden und das/die geeignete(n) Black-Box-Testverfahren auswählen

3.3 Erfahrungsbasierte Testverfahren

TA-3.3.1 (K2) Die Prinzipien der erfahrungsbasierten Testverfahren erklären, sowie deren Vor- und Nachteile mit Black-Box- und fehlerbasierten Testverfahren vergleichen

TA-3.3.2 (K3) Für ein vorgegebenes Szenario explorative Tests identifizieren

TA-3.3.3 (K2) Die Anwendung von fehlerbasierten Testverfahren beschreiben und ihren Einsatz von Black-Box-Testverfahren abgrenzen

3.4 Anwendung der bestgeeigneten Testverfahren

TA-3.4.1 (K4) Für eine vorgegebene Projektsituation festlegen, welche Black-Box- oder erfahrungsbasierten Testverfahren zur Erreichung bestimmter Ziele eingesetzt werden sollen

3.1 Einführung

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

- Black-Box-Testverfahren
- Erfahrungsbasierte Testverfahren

Diese Verfahren ergänzen sich gegenseitig und können je nach Bedarf für jede Testaktivität unabhängig von der Teststufe eingesetzt werden.

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

Die in den nachfolgenden Abschnitten behandelten Testverfahren konzentrieren sich auf die Bestimmung optimaler Testdaten (z.B. aus Äquivalenzklassen) oder die Ableitung von Verhaltensweisen des Testobjekts (z.B. aus Zustandsmodellen). Es ist üblich, die Verfahren zu kombinieren , um vollständige Testfälle zu erstellen.

3.2 Black-Box-Testverfahren

Black-Box-Testverfahren werden im ISTQB Foundation Level-Lehrplan [ISTQB_FL_SYL] eingeführt.

Gemeinsame Merkmale der Black-Box-Testverfahren sind:

- Entsprechend dem Testverfahren werden beim Testentwurf Modelle erstellt (z.B. Zustandsdiagramme oder Entscheidungstabellen).
- Aus diesen Modellen werden die Testbedingungen systematisch abgeleitet.

Testverfahren liefern in der Regel Kriterien für die Überdeckung, die als Maß für die Testentwurfs- und Testdurchführungsaktivitäten verwendet werden können. Eine vollständige Erfüllung dieser Überdeckungskriterien bedeutet nicht, dass die Menge von Tests vollständig ist, sondern dass das Modell keine weiteren Tests vorschlägt, um mit dem zugrundeliegenden Testverfahren eine höhere Überdeckung zu erzielen.

Black-Box-Testverfahren basieren meist auf Spezifikationsdokumenten, wie z.B. auf einer Anforderungsspezifikation für das System oder auf User Stories. Da die Spezifikationsdokumentation das Systemverhalten beschreiben sollte, insbesondere hinsichtlich der Funktionalität, lassen sich aus den spezifizierten Anforderungen Tests ableiten, die das Verhalten des Systems testen. Es kann vorkommen, dass keine dokumentierte Spezifikation vorhanden ist, aber die Anforderungen implizit vorliegen, beispielsweise wenn die Funktionalität eines vorhandenen Systems ersetzt werden soll.

Es gibt zahlreiche Black-Box-Testverfahren. Diese Verfahren zielen auf unterschiedliche Arten von Software und Szenarien ab. In den nachfolgenden Abschnitten wird sowohl die Anwendbarkeit der einzelnen Verfahren aufgezeigt als auch einige Einschränkungen und Schwierigkeiten, auf die der Test Analyst stoßen könnte. Außerdem wird die Methode zur Messung der Überdeckung beschrieben, sowie die Fehlerarten, die mit den jeweiligen Verfahren aufgedeckt werden.

Weitere Einzelheiten entnehmen Sie bitte [ISO29119-4], [Bath14], [Beizer95], [Black07], [Black09], [Copeland04], [Craig02], [Koomen06], und [Myers11].

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 sollten. Durch die Auswahl eines repräsentativen Wertes aus einer Äquivalenzklasse wird die Überdeckung aller Werte in derselben Äquivalenzklasse angenommen.

Anwendbarkeit
Dieses Verfahren ist in allen Teststufen anwendbar und ist dann geeignet, wenn erwartet wird, dass die in Gruppen aufgeteilten Werte auf dieselbe Weise verarbeitet werden und die Mengen der durch die Anwendung verwendeten Werte nicht interagieren. Die Auswahl von Wertemengen (Äquivalenzklassen) kann sowohl für gültige als auch für ungültige Daten erfolgen (d.h. Äquivalenzklassen mit Werten, die für die zu testende Software als ungültig angesehen werden sollten).

Die folgenden Kategorien von Werten müssen berücksichtigt werden:

- Werte aus einem ordinalskalierten Bereich. Zum Beispiel stellt der Ausdruck $0 < x < 5000$ den Bereich für die Variable "x" dar, deren gültige Äquivalenzklasse in der Rangordnung zwischen 1 und 4999 liegt. Ein repräsentativer gültiger Wert aus diesem Bereich ist beispielsweise 3249.
- Werte auf einer Nominalskala, die keine Rangordnung haben. Zum Beispiel sind die Farben "rot, blau, grün, gelb" allesamt gültige Äquivalenzklassen, und es ist nur ein Wert pro Äquivalenzklasse möglich.

Die Äquivalenzklassenbildung ist am nützlichsten, wenn sie mit der Grenzwertanalyse kombiniert wird, die auch die Werte im Bereich der Grenzen der einzelnen Äquivalenzklassen miteinschließt. Äquivalenzklassenbildung mit Werten aus den gültigen Äquivalenzklassen ist beim Smoke-Test eines neuen Builds oder einer neuen Version sehr gebräuchlich, da damit sehr schnell festgestellt wird, ob die grundlegende Funktionalität erfüllt ist.

Einschränkungen/Schwierigkeiten
Wenn die Annahme, dass die Werte in der Äquivalenzklasse genau gleichbehandelt werden, nicht zutrifft, dann können mit diesem Verfahren Fehler übersehen werden. Es ist auch wichtig, dass die Äquivalenzklassen sorgfältig ausgewählt werden. Beispiel: Für ein Eingabefeld für positive und negative Zahlen empfiehlt es sich, zwei gültige Äquivalenzklassen zu bilden und zu testen, eine für die positiven und eine für die negativen Zahlen, da diese sehr wahrscheinlich unterschiedlich verarbeitet werden. Je nachdem, ob die Null zulässig ist oder nicht, würde dafür eine weitere Äquivalenzklasse gebildet werden (hier ist zu beachten, dass die Null weder positiv noch negativ ist). Für einen Test Analysten ist es wichtig die zugrunde liegenden Verarbeitungsprozesse zu verstehen, um die bestmögliche Aufteilung in Äquivalenzklassen zu bestimmen. Dafür kann es erforderlich werden sich bei der Analyse des Codes Unterstützung zu holen, um zu Verstehen, was der Code überhaupt macht.

Überdeckung
Die Überdeckung wird ermittelt, indem die Anzahl der Äquivalenzklassen, für die ein Wert getestet wurde, durch die Gesamtzahl der identifizierten Äquivalenzklassen geteilt wird. Die Überdeckung wird dann in Prozent angegeben. Die Verwendung mehrerer Werte für eine einzelne Äquivalenzklasse erhöht die prozentuale Überdeckung nicht.

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

3.2.2 Grenzwertanalyse

Die Grenzwertanalyse wird eingesetzt, um zu testen, ob die Werte an den Grenzen der Äquivalenzklassen richtig verarbeitet werden. 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 (um das kleinstmögliche Inkrement) nächste Wert unmittelbar außerhalb der Grenze verwendet. 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 innerhalb der definierten 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 höhere Risiken mit drei Werten getestet werden sollten.

Anwendbarkeit
Dieses Verfahren ist in allen Teststufen anwendbar und ist dann geeignet, wenn die Äquivalenzklassen geordnet sind. Aus diesem Grund wird die Grenzwertanalyse oft zusammen mit der Äquivalenzklassenbildung durchgeführt. Geordnete Äquivalenzklassen sind erforderlich, da das Konzept vorsieht, dass Werte auf oder unmittelbar neben der Grenze liegen. Ein Zahlenbereich ist beispielsweise eine geordnete Äquivalenzklasse. Eine Äquivalenzklasse, die ausschließlich aus rechteckigen Objekten besteht, ist keine geordnete Äquivalenzklasse und hat keine Grenzwerte. Außer für Zahlenbereiche kann die Grenzwertanalyse auch angewendet werden für:

- numerische Attribute nicht-numerischer Variablen (z.B. Länge einer Zeichenkette)
- Schleifen, einschließlich von Schleifen in Zustandsdiagrammen, in Anwendungsfällen und in gespeicherten Datenstrukturen wie z.B. Arrays
- physikalische Objekte (einschließlich Speicher)
- zeitbestimmte Aktivitäten

Einschränkungen/Schwierigkeiten
Da die Korrektheit der Grenzwertanalyse von der korrekten Festlegung der Äquivalenzklassen abhängt, gelten dieselben Einschränkungen und Schwierigkeiten wie bei der Äquivalenzklassenbildung. Der Test Analyst muss die Inkremente bei den gültigen und ungültigen Werten genau berücksichtigen, um die zu testenden Werte korrekt bestimmen zu können. Durch Grenzwertanalyse können nur geordnete Äquivalenzklassen getestet werden, aber das beschränkt sich nicht auf den Wertebereich gültiger Eingaben. Beispiel: Wenn eine Anzahl von Zellen einer Tabellenkalkulationsanwendung zu testen ist, dann gibt es eine Äquivalenzklasse, die alle Zellen bis zur maximalen Anzahl enthält, einschließlich der Zelle an der Grenze; und eine weitere Äquivalenzklasse, die mit der Zelle genau über der Grenze beginnt.

Überdeckung
Die Überdeckung wird ermittelt, indem die Anzahl der getesteten Grenzwerte durch die Anzahl der insgesamt identifizierten Grenzwerte geteilt wird (unter Nutzung entweder der 2-Wert oder der 3-Wert-Methode). Die Überdeckung wird in Prozent angegeben.

Fehlerarten
Die Grenzwertanalyse deckt verschobene oder fehlende Grenzen zuverlässig auf, und kann zusätzliche Grenzen finden. 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, z.B. das System unterstützt 10.000 gleichzeitige Benutzer, aber nicht 10.001.

3.2.3 Entscheidungstabellentest

Entscheidungstabellen ermöglichen die Erfassung von Bedingungen, die von der zu testenden Software verarbeitet werden, und die Bewertung der Ergebnisse der Anwendung verschiedener wahr- oder falsch-Werte auf diese Bedingungen. Test Analysten können Entscheidungstabellen verwenden, um die Entscheidungsregeln zu erfassen, die für die Software unter Test gelten.
Bei diesem Verfahren richtet der Test Analyst initial die Entscheidungstabelle als vollständige Tabelle ein, wobei die Anzahl der Spalten gleich 2 hoch der Anzahl der Bedingungen ist (2^n, wobei n die Anzahl der Bedingungen ist). Eine initiale Bedingungstabelle für drei Bedingungen hätte beispielsweise acht Spalten (2^3).

Jede Spalte in der Tabelle stellt eine Entscheidungsregel dar. Eine typische (generische) Entscheidungsregel könnte lauten: „Wenn Bedingung A wahr ist und Bedingung B wahr ist und Bedingung C falsch ist, dann wird die Aktion X erwartet".

Beim Entscheidungstabellentest können zwei Ansätze angewandt werden, je nachdem, was überdeckt werden soll:

- Überdeckung der Kombinationen von Bedingungen, aus denen sich die Regeln zusammensetzen
- Überdeckung von einzelnen Bedingungen

Überdecken von Kombinationen von Bedingungen
Bei der "klassischen" Verwendung von Entscheidungstabellen werden die Entscheidungsregeln getestet, die aus Kombinationen von Bedingungen bestehen.

Wenn jede mögliche Kombination von Bedingungen getestet wird, können die Entscheidungstabellen sehr groß werden. Eine Methode, die Anzahl der Kombinationen systematisch zu reduzieren, so dass anstatt aller möglichen Bedingungskombinationen nur solche getestet werden, die signifikant von anderen Kombinationen abweichen, wird konsolidierter Entscheidungstabellentest genannt [Copeland04]. Bei diesem Verfahren werden die Kombinationen der Bedingungen auf diejenigen reduziert, die unterschiedliche Ausgaben erzeugen. Bedingungen, die die Ausgabe nicht beeinflussen, werden weggelassen. Diese Regeln werden als redundant betrachtet und in der Entscheidungstabelle als “irrelevant” gekennzeichnet. Außerdem werden Entscheidungsregeln weggelassen, die unmögliche Kombinationen von Bedingungen enthalten. Beispiel: Wenn ein Fahrkartenautomat unterschiedliche Entscheidungsregeln in Abhängigkeit von den Bedingungen "vor 9 Uhr" und "nach 17 Uhr" anwendet, dann ist es unmöglich, eine Entscheidungsregel zu definieren, bei der beide Bedingungen (am selben Tag) wahr sind. Unmögliche Entscheidungsregeln werden aus der Tabelle entfernt. Die Eliminierung von Regeln mit redundanten oder von unmöglichen Kombinationen von Bedingungen führt zu einer teilweise konsolidierten Entscheidungstabelle. Bei einer vollständig konsolidierten Tabelle werden beide Arten von Bedingungen entfernt.

Zur Überdeckung von Kombinationen von Bedingungen versucht der Test Analyst eine konsolidierte Entscheidungstabelle zu erstellen. Wenn die Anzahl der aus der konsolidierten Tabelle abgeleiteten Testfälle immer noch als zu groß erachtet wird, muss eine risikobasierte Auswahl getroffen werden.

Überdeckung von einzelnen Bedingungen
Ziel bei diesem Ansatz ist es, sicherzustellen, dass das wahre und falsche Ergebnis jeder Bedingung einzeln getestet wird. Die erste zu berücksichtigende Entscheidungsregel definiert den Wert für alle Einzelbedingungen als gleich (z.B. wahr). Diese Entscheidungsregel wird zusammen mit der erwarteten Aktion in die Entscheidungstabelle eingetragen. Bei der Definition der zweiten Entscheidungsregel wird der Wert der ersten Bedingung in den entgegengesetzten Zustand (z.B. falsch) gesetzt. Auch diese zweite Entscheidungsregel wird zusammen mit der erwarteten Aktion in die Entscheidungstabelle eingetragen. Die Definition der dritten Entscheidungsregel erfordert, dass der Wert der ersten Bedingung auf den in der ersten Entscheidungsregel verwendeten Wert

zurückgesetzt wird und dass der Wert der zweiten Bedingung umgestellt wird. Auch hier wird die Entscheidungsregel in die Entscheidungstabelle eingetragen und die erwartete Aktion definiert. Der Vorgang wird so lange wiederholt, bis jede Bedingung in den Entscheidungsregeln den Wert "wahr" und "falsch" angenommen hat. Dies resultiert in einer Entscheidungstabelle mit der Anzahl der Entscheidungsregeln, die der Anzahl der Bedingungen + 1 entspricht. Wie bei dem oben beschriebenen Ansatz („Überdeckung aller Kombinationen von Bedingungen") werden alle Regeln, die unmöglich sind, aus der Tabelle entfernt.

Im Vergleich zum ersten Ansatz („Überdeckung von Kombinationen von Bedingungen") führt dieser systematische Ansatz im Allgemeinen zu weniger Entscheidungsregeln, die getestet werden müssen. Jede Bedingung wird mindestens einmal auf den Wert "wahr" und mindestens einmal auf den Wert "falsch" getestet. Da jede Entscheidungsregel auf eine bestimmte Bedingung ausgerichtet ist, wird die Lokalisierung der aufgedeckten Fehlerzustände vereinfacht. Mit diesem Ansatz können Fehlerzustände, die nur bei bestimmten Bedingungskombinationen auftreten, übersehen werden, da nur einfache Kombinationen berücksichtigt werden

Anwendbarkeit
Beide Ansätze dieses Verfahrens werden üblicherweise in den Teststufen Integrations-, System- und Abnahmetest angewendet. Abhängig vom Programmcode ist das Verfahren auch beim Komponententest anwendbar, wenn die zu testende Komponente für Entscheidungslogik verantwortlich ist. Dieses Verfahren ist besonders nützlich, wenn die Anforderungen in Form von Ablaufdiagrammen oder Tabellen mit Geschäftsregeln vorliegen.

Entscheidungstabellen können auch zur Spezifikation von Anforderungen verwendet werden, und es kommt vor, dass Anforderungsspezifikationen bereits in diesem Format definiert sind. Auch wenn die Anforderungen nicht in tabellarischer Form oder als Ablaufdiagramm vorliegen, werden Bedingungen und Kombinationen von Bedingungen oft im Text vorgefunden.

Beim Entwurf von Entscheidungstabellen ist es wichtig, sowohl die spezifizierten Entscheidungsregeln zu berücksichtigen, als auch solche, die nicht ausdrücklich spezifiziert (aber trotzdem vorhanden) sind. Im Allgemeinen muss ein Test Analyst in der Lage sein, die erwarteten Aktionen für alle Entscheidungsregeln gemäß der Spezifikation oder dem Testorakel abzuleiten.

Einschränkungen/Schwierigkeiten
Beim Ansatz "Kombinationen von Bedingungen" kann die Identifizierung aller interagierenden Bedingungen eine Herausforderung darstellen, insbesondere dann, wenn die Anforderungen an das System nicht klar definiert sind oder nicht vorliegen. Bei der Auswahl der Anzahl von Bedingungen in einer Entscheidungstabelle muss darauf geachtet werden, dass die Anzahl der Kombinationen dieser Bedingungen überschaubar bleibt. Wenn die Anzahl der Bedingungen zu einer unüberschaubaren Anzahl von Kombinationen führt, kann der Test Analyst eine Hierarchie von Entscheidungstabellen definieren, in der die Bedingungen einer bestimmten Entscheidungstabelle aus dem Aufruf einer anderen Entscheidungstabelle resultieren können.

Überdeckung
Es werden typischerweise drei Grade der Überdeckung in Betracht gezogen:

- Überdeckung aller Bedingungsergebnisse: Es werden ausreichend viele Entscheidungsregeln angewendet, um die Wahr/Falsch-Ergebnisse jeder Bedingung auszuführen. Dies ist die kosteneffektivste Überdeckung (Anzahl der Bedingungen + 1 Testfälle), aber es werden nicht notwendigerweise alle möglichen erwarteten Aktionen ausgeführt.
- Überdeckung aller Aktionen: Es werden genügend Entscheidungsregeln angewendet, um alle möglichen erwarteten Aktionen zu erzeugen.
- Überdeckung aller nicht redundanten und möglichen Entscheidungsregeln: Es werden alle Entscheidungsregeln in einer vollständig konsolidierten Entscheidungstabelle überdeckt. Dies

erzielt einen höheren Überdeckungsgrad, kann aber zu einer großen Anzahl von Testfällen führen.

Beim Entwurf von Tests aus einer Entscheidungstabelle sollten auch die Grenzwerte beachtet werden, die zu testen sind. Hierdurch kann sich die Anzahl von Testfällen erhöhen, die für ein adäquates Testen der Software benötigt werden. Grenzwertanalyse und Äquivalenzklassenbildung sind Verfahren, die den Entscheidungstabellentest ergänzen.

Fehlerarten
Zu den typischen Fehlerzuständen, die mit diesem Verfahren aufgedeckt werden, gehört die fehlerhafte Verarbeitung, die bei bestimmten Kombinationen von Bedingungen zu unerwarteten Ergebnissen führt. Beim Erstellen der Entscheidungstabellen können Fehlerzustände in den Spezifikationsdokumenten aufgedeckt werden. Es ist nicht ungewöhnlich eine Menge von Bedingungen in der Entscheidungstabelle anzulegen und dann herauszufinden, dass das erwartete Ergebnis für eine oder mehrere Entscheidungsregeln nicht spezifiziert ist. Die häufigsten Fehlerarten sind Lücken (es ist nicht spezifiziert, was in einer bestimmten Situation tatsächlich geschehen soll) und Widersprüche. Beim Testen können auch Probleme mit Kombinationen von Bedingungen gefunden werden, die gar nicht oder nicht korrekt verarbeitet werden.

3.2.4 Zustandsübergangstest

Der Zustandsübergangstest wird eingesetzt, um die Fähigkeit des Testobjekts zu testen, definierte Zustände über gültige Zustandsübergänge zu betreten und zu verlassen, sowie ungültige Zustände zu betreten und ungültige Zustandsübergänge auszuführen. Ereignisse lösen im Testobjekt den Übergang von einem Zustand in einen anderen Zustand und die Ausführung von Aktionen aus. Ereignisse können durch von Bedingungen abhängen (manchmal auch als Wächterbedingungen oder Übergangswächter bezeichnet), die die Auswahl des Zustandsübergangspfads beeinflussen. Beispiel: Das Anmelden mit einer gültigen Kombination von Benutzername und Passwort führt zu einem anderen Zustandsübergang als das Anmelden mit einem ungültigen Passwort. Zustandsübergänge werden in einem Zustandsdiagramm oder in einer Zustandsübergangstabelle dargestellt (die mitunter auch die ungültigen Übergänge zwischen Zuständen auflisten) [Black09].

Anwendbarkeit
Der Zustandsübergangstest ist für jede Software anwendbar, die definierte Zustände hat und bei der die Übergänge zwischen diesen Zuständen durch Ereignisse ausgelöst werden (z.B. Wechsel der Bildschirmmasken). Der Zustandsübergangstest kann auf jeder Teststufe eingesetzt werden. Eingebettete Software, Web-Software und jede Art von Transaktionssystemen bieten sich für dieses Testverfahren an. Auch für das Testen von Steuerungssystemen (z.B. Systeme zur Steuerung von Ampeln) ist der Zustandsübergangstest gut geeignet.

Einschränkungen/Schwierigkeiten
Die Bestimmung der Zustände ist oft die schwierigste Aufgabe beim Erstellen von Zustandsdiagrammen oder Zustandsübergangstabellen. Wenn das Testobjekt eine Benutzungsschnittstelle besitzt, dann werden häufig die einzelnen Bildschirmmasken, die dem Benutzer angezeigt werden, zur Definition der Zustände verwendet. Bei eingebetteter Software können die Zustände von den Zuständen der Hardware abhängen.

Abgesehen von den Zuständen ist der einzelne Zustandsübergang (auch als 0-Switch bezeichnet) die Basisgröße beim Zustandsübergangstest. Das Testen der einzelnen Zustandsübergänge kann einige Fehlerarten aufdecken. Mehr Fehlerzustände werden jedoch gefunden, wenn Folgen von Übergängen getestet werden. Eine Sequenz mit zwei aufeinanderfolgenden Zustandsübergängen wird als 1-Switch bezeichnet; eine Sequenz von drei aufeinanderfolgenden Übergängen ist ein 2-Switch, und so weiter. Diese “Switches” werden manchmal auch als N-1-Switches bezeichnet, wobei N für die Anzahl der zu

durchlaufenden Zustandsübergänge steht. Zum Beispiel wäre der einfache Zustandsübergang (0-Switch) gleichbedeutend mit dem 1-1-Switch [Bath14].

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

Der Begriff "N-Switch-Überdeckung" bezieht sich auf den prozentuellen Anteil der getesteten N-Switches (d.h., Sequenzen von N+1 Zustandsübergängen) an der Gesamtzahl der N-Switches. Beispiel: Für 100% 1-Switch-Überdeckung muss jede gültige Sequenz von zwei aufeinander folgenden Zustandsübergängen mindestens einmal getestet werden. Dieses Testen kann einige Arten von Fehlerwirkungen auslösen, die mit 100% 0-Switch-Überdeckung unentdeckt geblieben wären.

"Rundreise-Überdeckung" betrifft Situationen, in denen Folgen von Zustandsübergängen Schleifen bilden. 100% Rundreise-Überdeckung ist erzielt, wenn alle Schleifen von einem beliebigen Zustand zurück in denselben Zustand für alle Zustände getestet wurden, in denen Schleifen beginnen und enden. In dieser Schleife darf jeder bestimmte Zustand (außer dem Anfangs-/Endzustand) nicht mehr als einmal vorkommen [Offutt16].

Bei all diesen Ansätzen kann man versuchen, eine noch höhere Überdeckung zu erzielen, indem auch alle ungültigen Zustandsübergänge, die in einer Zustandsübergangstabelle identifiziert wurden, mit einbezogen werden. Aus den Überdeckungsanforderungen und den Überdeckungsmengen für den Zustandsübergangstest muss hervorgehen, ob ungültige Zustandsübergänge enthalten sind.

Das Entwerfen von Testfällen zur Erzielung der gewünschten Überdeckung wird durch das Zustandsdiagramm oder die Zustandsübergangstabelle für das jeweilige Testobjekt unterstützt. Diese Informationen können auch in einer Tabelle dargestellt werden, die die N-Switch-Zustandsübergänge für einen bestimmten Wert von "N" enthält [Black09].

Zur Identifizierung der zu überdeckenden Elemente (z.B. Zustandsübergänge, Zustände oder N-Switches) kann ein manuelles Verfahren angewandt werden. Eine empfohlene Methode ist, das Zustandsdiagramm und die Zustandsübergangstabelle auszudrucken und mit einem Stift die getesteten Elemente zu markieren, bis die erforderliche Überdeckung erzielt ist [Black09]. Dieser Ansatz wäre bei komplexeren Zustandsdiagrammen und Zustandsübergangstabellen jedoch zu zeitaufwendig. Daher sollte ein Werkzeug zur Unterstützung des Zustandsübergangstests verwendet werden.

Fehlerarten
Zu den typischen Fehlerzuständen [Beizer95], die mit diesem Verfahren aufgedeckt werden, gehören:

- falsche Ereignisse bzw. Werte
- ungültige Aktionen bzw. Werte
- falscher Ausgangszustand
- Unfähigkeit, einen Endzustand zu erreichen
- Unfähigkeit, erforderliche Zustände zu betreten
- überschüssige (nicht benötigte) Zustände
- Unfähigkeit, gültige Zustandsübergänge korrekt auszuführen

- Möglichkeit, ungültige Zustandsübergänge auszuführen

Beim Erstellen des Zustandsmodells können Fehlerzustände in den Spezifikationen aufgedeckt werden. Am häufigsten handelt es sich dabei um Lücken (es ist nicht spezifiziert, was in einer bestimmten Situation tatsächlich geschehen soll) und um Widersprüche.

3.2.5 Klassifikationsbaumverfahren

Klassifikationsbäume unterstützen bestimmte Black-Box-Testverfahren, indem sie eine grafische Darstellung des für das Testobjekt geltenden Datenraums ermöglichen.

Die Daten werden wie folgt in Klassifikationen und Klassen organisiert:

- Klassifikationen: Diese stellen Parameter innerhalb des Datenraums für das Testobjekt dar. Dies können sowohl Eingabeparameter sein, die zusätzlich Umgebungszustände und Vorbedingungen enthalten können, als auch Ausgabeparameter. Beispiel: Wenn eine Anwendung auf viele verschiedene Arten konfiguriert werden kann, können die Klassifikationen Client, Browser, Sprache und Betriebssystem umfassen.
- Klassen: Jede Klassifikation kann beliebig viele Klassen und Unterklassen haben, die das Auftreten des Parameters beschreiben. Jede Klasse bzw. Äquivalenzklasse ist ein bestimmter Wert innerhalb einer Klassifikation. Im obigen Beispiel könnte die Sprachklassifikation Äquivalenzklassen für Deutsch, Englisch, Französisch und Spanisch enthalten.

Klassifikationsbäume ermöglichen es dem Test Analysten, nach eigenem Ermessen Kombinationen einzugeben. Dazu gehören z.B. paarweise (oder 2-fache) Kombinationen (siehe Abschnitt 3.2.6), dreifache Kombinationen, sowie einzelne Eingaben.

Weitere Informationen zur Anwendung des Klassifikationsbaumverfahrens sind in [Bath14] und [Black09] enthalten.

Anwendbarkeit
Die Erstellung eines Klassifikationsbaums hilft einem Test Analysten, Äquivalenzklassen (Klassifikationen) und die darin enthaltenen Werte (Klassen) zu identifizieren, die von Interesse sind.

Bei der weiteren Analyse des Klassifikationsbaums lassen sich nicht nur mögliche Grenzwerte identifizieren, sondern auch bestimmte Kombinationen von Eingaben, die entweder von besonderem Interesse sind oder nicht berücksichtigt werden müssen (z.B. weil sie inkompatibel sind). Der resultierende Klassifikationsbaum kann dann zur Unterstützung von Äquivalenzklassentests, Grenzwertanalysen oder des paarweisen Testens verwendet werden (siehe Abschnitt 3.2.6).

Einschränkungen/Schwierigkeiten
In dem Maße, wie die Anzahl der Klassifikationen und/oder Klassen zunimmt, wird das Diagramm größer und schwieriger zu benutzen.

Überdeckung
Testfälle können so entworfen werden, dass z.B. eine Mindestüberdeckung der Klassen erzielt wird, d.h. jeder Wert einer Klassifikation wird mindestens einmal getestet. Der Test Analyst kann auch entscheiden, dass paarweise (2-fache) oder sogar dreifache Kombinationen überdeckt werden sollen.

Fehlerarten
Die Art der gefundenen Fehlerzustände hängt von den Verfahren ab, die die Klassifikationsbäume unterstützen (d. h. Äquivalenzklassenbildung, Grenzwertanalyse oder paarweises Testen).

3.2.6 Paarweises Testen

Das paarweise Testen wird für das Testen von Software verwendet, bei der mehrere Eingabeparameter mit jeweils mehreren möglichen Werten in Kombination getestet werden müssen, wodurch mehr Kombinationen entstehen, als in der verfügbaren Zeit getestet werden können. Die Eingabeparameter sollten insofern kompatibel sein, dass jede Option für jeden beliebigen Faktor (d.h. jeder ausgewählte Wert für einen beliebigen Eingabeparameter) mit jeder Option eines beliebigen anderen Faktors kombiniert werden kann. Die Kombination eines bestimmten Parameters (Variable oder Faktor) mit einem bestimmten Wert dieses Parameters wird als Parameter-Wert-Paar bezeichnet (z.B. wenn 'Farbe' ein Parameter mit (beispielsweise) sieben zulässigen Werten ist, dann wäre 'Farbe = rot' ein Parameter-Wert-Paar).

Das paarweise Testen verwendet kombinatorische Arithmetik, um sicherzustellen, dass jedes Parameter-Wert-Paar einmal gegen jedes Parameter-Wert-Paar jedes anderen Parameters getestet wird (d.h. 'alle Paare' von Parameter-Wert-Paaren werden getestet). Dabei wird vermieden, dass alle Kombinationen von Parameter-Wert-Paaren getestet werden. Bei einem manuellen Ansatz (nicht empfohlen!) würde der Test Analyst eine Tabelle mit einer Zeile für jeden Testfall und je einer Spalte für jeden Parameter erstellen; diese würde er dann mit Werten so füllen, dass alle Wertepaare in der Tabelle vorkommen (siehe [Black09]). Alle leer gebliebenen Felder der Tabelle können vom Test Analysten unter Verwendung seines eigenen Wissens über den Geschäftsbreich mit Werten gefüllt werden.

Es gibt verschiedene Werkzeuge, die den Test Analysten bei dieser Aufgabe unterstützen (siehe www.pairwise.org für Beispiele). Diese Werkzeuge benötigen als Eingabe eine Liste der Parameter und deren Werte und berechnen eine Mindestmenge von Kombinationen, die alle Paare von Parameter-Wert-Paaren überdecken. Die Ausgabe des Werkzeugs kann als Eingabe für Testfälle verwendet werden. Dabei ist zu beachten, dass der Test Analyst für jede vom Werkzeug erstellte Kombination die erwarteten Ergebnisse bereitstellen muss.

Klassifikationsbäume (siehe Abschnitt 3.2.5) werden häufig in Verbindung mit paarweisem Testen verwendet [Bath14]. Das Erstellen von Klassifikationsbäumen wird durch Werkzeuge unterstützt und ermöglicht es, Kombinationen von Parametern und deren Werte zu visualisieren (einige Werkzeuge bieten eine Erweiterung für paarweises Testen). Dies hilft, die folgenden Informationen zu identifizieren:

- Eingaben für das paarweise Testen
- bestimmte Kombinationen von besonderem Interesse (z.B. häufig verwendete Kombinationen oder häufige Fehlerquellen)
- bestimmte Kombinationen, die nicht kompatibel sind. Daraus kann jedoch nicht abgeleitet werden, dass sich die miteinander kombinierten Faktoren nicht gegenseitig beeinflussen; das ist durchaus möglich, sollte allerdings in einer akzeptablen Art und Weise erfolgen
- logische Beziehungen zwischen den Variablen. Beispiel: "Wenn Variable 1 = x, dann kann Variable 2 nicht y sein". Klassifikationsbäume, die diese Beziehungen erfassen, werden "Feature-Modelle" genannt.

Anwendbarkeit
Das Problem zu vieler Kombinationen von Parameterwerten wird in mindestens zwei verschiedenen Situationen beim Testen deutlich. Einige Systeme behandeln mehrere Parameter mit jeweils mehreren möglichen Werten, z.B. eine Bildschirmmaske mit mehreren Eingabefeldern. In diesem Fall sind die Kombinationen von Parameterwerten die Eingabedaten für die Testfälle. Ferner können einige Systeme in mehreren Dimensionen konfigurierbar sein, was zu einem potenziell großen

Konfigurationsraum führt. In beiden Fällen kann das paarweise Testen eingesetzt werden, um eine Teilmenge von Kombinationen in einem akzeptablen Umfang zu identifizieren.

Bei Parametern mit sehr vielen Werten kann Äquivalenzklassenbildung oder irgendein anderer Auswahlmechanismus zunächst auf jeden Parameter einzeln angewendet werden, um die Anzahl der Werte für jeden Parameter zu reduzieren. Anschließend wird das paarweise Testen eingesetzt, um die Menge der resultierenden Kombinationen zu reduzieren. Die Erfassung der Parameter und ihrer Werte in einem Klassifikationsbaum unterstützt diese Aktivität.

Diese Verfahren werden üblicherweise in den Teststufen Komponentenintegrations-, System- und Systemintegrationstest angewendet.

Einschränkungen/Schwierigkeiten
Die größte Einschränkung bei diesen Verfahren ist die Annahme, dass die Ergebnisse einiger weniger Tests repräsentativ für alle Tests sind, und dass diese wenigen Tests die erwartete Nutzung abbilden. Falls es eine unerwartete Interaktion zwischen bestimmten Variablen gibt, kann sie bei diesem Testverfahren unentdeckt bleiben, wenn diese bestimmte Kombination nicht getestet wird. Diese Verfahren können für Personen ohne technischen Hintergrund schwierig zu erklären sein, da sie die logische Reduzierung der Tests möglicherweise nicht verstehen. Bei den Erklärungen sollten die Ergebnisse aus empirischen Studien [Kuhn16] erwähnt werden, die zeigten, dass im Bereich der untersuchten Medizinprodukte 66% der Fehlerwirkungen durch eine einzige Variable und 97% durch eine oder zwei miteinander interagierende Variablen ausgelöst wurden. Es besteht ein Restrisiko, dass beim paarweisen Testen Fehlerwirkungen des Systems nicht erkannt werden, die auftreten, wenn drei oder mehr Variablen interagieren.

Die Identifizierung der Parameter und ihrer jeweiligen Werte kann schwierig sein. Diese Aufgabe sollte daher möglichst mit Hilfe von Klassifikationsbäumen durchgeführt werden (siehe Abschnitt 3.2.5). Auch ist es schwierig, die minimale Anzahl von Kombinationen manuell zu bestimmen, die eine bestimmte Überdeckung erfüllen. Zur Bestimmung der minimalen Menge von Kombinationen werden meist Werkzeuge eingesetzt. Einige Werkzeuge bieten die Möglichkeit, vorgegebene Kombinationen in die endgültige Auswahl der Kombinationen einzubeziehen oder auszuschließen. Diese Fähigkeit der Werkzeuge kann der Test Analyst nutzen, um basierend auf seiner Kenntnis des Geschäftsbereichs oder den Informationen über die Produktverwendung einzelne Faktoren mehr oder weniger zu gewichten.

Überdeckung
100% Überdeckung beim paarweisen Testen erfordert, dass jedes Wertepaar eines jeden Parameterpaares in mindestens einer Kombination enthalten sein muss.

Fehlerarten
Mit diesem Testverfahren werden meistens Fehlerzustände in Zusammenhang mit den kombinierten Werten von zwei Parametern aufgedeckt.

3.2.7 Anwendungsfallbasierter Test

Der anwendungsfallbasierte Test liefert transaktionale, auf Szenarien des Verhaltens basierende Tests, die die beabsichtigte Nutzung der durch den Anwendungsfall spezifizierten Komponente oder des Systems nachahmen. Anwendungsfälle spezifizieren die Interaktionen zwischen den Akteuren und einer Komponente oder einem System, die zu einem Ziel führen. Die Akteure können Benutzer, externe Hardware oder andere Komponenten oder Systeme sein.

Anwendbarkeit
Der anwendungsfallbasierte Test wird meist beim System- und Abnahmetest angewendet. Er kann auch für Integrationstests verwendet werden, bei denen das Verhalten der Komponenten oder Systeme durch Anwendungsfälle spezifiziert ist, und sogar für Komponententests, sofern das Verhalten der Komponenten durch Anwendungsfälle spezifiziert ist. Anwendungsfälle dienen häufig auch als Basis für den Performanztest, da sie eine realistische Nutzung des Systems widerspiegeln. Die in den Anwendungsfällen beschriebenen Szenarien können virtuellen Benutzern zugewiesen werden, um eine realistische Last zu erzeugen (sofern Last- und Performanzanforderungen in den Anwendungsfällen bzw. für diese spezifiziert sind).

Einschränkungen/Schwierigkeiten
Um Gültigkeit zu besitzen, müssen Anwendungsfälle realistische Benutzertransaktionen ausdrücken. Die Spezifikation von Anwendungsfällen ist eine Form des Systementwurfs. Die Informationen dafür sollten von Benutzern oder Benutzervertretern kommen und sollten vor dem Entwurf entsprechender Anwendungsfälle gegen organisatorische Anforderungen geprüft werden. Der Wert eines Anwendungsfalles wird gemindert, wenn dieser die tatsächlichen Anforderungen der Benutzer und der Organisation nicht genau wiedergibt oder die Erfüllung der Benutzeraufgaben eher behindert als unterstützt.

Eine genaue Spezifikation des Verhaltens bei Ausnahmenbehandlungen, alternativen Abläufen und Fehlerbehandlungen ist für eine gründliche Überdeckung unerlässlich. Anwendungsfälle sollten als eine Art Richtlinie aufgefasst werden, nicht als vollständige Definition dessen, was zu testen ist, da sie möglicherweise keine eindeutige Definition der gesamten Menge von Anforderungen liefern. Es kann sich auch auszahlen, andere Modelle wie Ablaufdiagramme oder Entscheidungstabellen aus der Anwendungsfallbeschreibeung zu erstellen. Dadurch lässt sich die Genauigkeit des Testens verbessern und der Anwendungsfall an sich verifizieren. Wie auch bei anderen Formen der Spezifikation ist es wahrscheinlich, dass dadurch in der Spezifikation der Anwendungsfälle vorhandene logische Anomalien aufdeckt werden (falls vorhanden).

Überdeckung
Für die Mindestüberdeckung eines Anwendungsfalls ist ein Testfall für das Standardverhalten erforderlich und darüber hinaus genügend zusätzliche Testfälle, um alle alternativen Verhalten und Fehlerbehandlungen zu überdecken. Wenn die Testmenge minimal sein soll, können mehrere alternative Verhaltenmit einem Testfall abgedeckt werden, sofern diese miteinander kompatibel sind. Wenn eine bessere Diagnosefähigkeit benötigt wird (z.B. zur Unterstützung bei der Isolierung von Fehlerzuständen), kann für jedes Alternativverhalten ein zusätzlicher Testfall entworfen werden, wobei verschachtelte Alternativverhalten immer noch erfordern, dass einige zu einzelnen Testfällen zusammengefasst werden müssen (z.B. alternatives Abbruch- versus Nicht-Abbruch-Verhalten bei einem Neuversuch nach einer Ausnahme).

Fehlerarten
Zu den mit diesem Testverfahren aufgedeckten Fehlerzuständen gehören die falsche Verarbeitung von definierten Abläufen, fehlende alternative Abläufe, die inkorrekte Verarbeitung der vorliegenden Bedingungen, sowie schlecht umgesetzte oder inkorrekte Fehlermeldungen.

3.2.8 Testverfahren kombinieren

Manchmal werden für die Erstellung von Testfällen auch Verfahren kombiniert. So können beispielsweise für die in einer Entscheidungstabelle identifizierten Bedingungen Äquivalenzklassen gebildet werden, um mehrere Möglichkeiten herauszufinden, mit denen eine Bedingung erfüllt werden kann. Die Testfälle decken dann nicht nur jede Kombination von Bedingungen ab, sondern es werden für die erstellten Äquivalenzklassen zusätzliche Testfälle generiert, um diese zu überdecken. Bei der Auswahl der geeigneten Testverfahren sollte der Test Analyst die Anwendbarkeit des Verfahrens, die Einschränkungen und Schwierigkeiten sowie die Testziele bezüglich der Überdeckung und die

aufzudeckenden Fehlerzustände berücksichtigen. Diese Aspekte sind für die einzelnen, in diesem Kapitel behandelten Testverfahren beschrieben. Es gibt nicht immer das einzig richtige und beste Verfahren für eine bestimmte Situation. Häufig liefert erst eine Kombination von Verfahren die vollständigste Überdeckung, vorausgesetzt, für die korrekte Anwendung ist ausreichend Zeit und Können vorhanden.

3.3 Erfahrungsbasierte Verfahren

Erfahrungsbasiertes Testen nutzt die fachliche Kompetenz und die Intuition der Tester sowie deren Erfahrungen mit ähnlichen Anwendungen oder Technologien, um das Testen gezielt zu steuern und die Fehlerfindung zu erhöhen. Diese Testverfahrenen reichen von "Schnelltests", bei denen der Tester keine formal vorgeplanten Aktivitäten durchführen muss, über vorgeplante Sitzungen bis hin zu skriptbasierten Sitzungen. Sie sind fast immer nützlich, sind jedoch dann besonders wertvoll, wenn damit die in der folgenden Liste enthaltenen Vorteile erzielt werden können.

Erfahrungsbasiertes Testen hat folgende Vorteile:

- Es ist effektiv beim Finden von Fehlern.
- Es kann eine gute Alternative zu strukturierten Vorgehensweisen sein, wenn die Systemdokumentation schlecht ist.
- Es dann genutzt werden, wenn die verfügbare Zeit zum Testen extrem knapp ist.
- Beim Testen kann vorhandenes Fachwissen über den Geschäftsbereich und die Technologie genutzt werden, das auch von Personen stammt, die nicht in das Testen involviert sind (z.B. Businessanalysten, Kunden oder Auftraggeber).
- Es kann den Entwicklern frühzeitiges Feedback liefern.
- Es hilft dem Team, sich mit der Software im Laufe ihrer Herstellung vertraut zu machen.
- Es ist effektiv bei der Analyse von Betriebsstörungen oder -ausfällen.
- Es ermöglicht die Anwendung einer Vielzahl von Testverfahren.

Erfahrungsbasiertes Testen hat die folgenden Nachteile:

- Es kann für Systeme ungeeignet sein, die eine detaillierte Testdokumentation benötigen.
- Hohe Wiederholbarkeit ist damit nur schwierig zu erreichen.
- Eine genaue Beurteilung der Überdeckung ist nur begrenzt möglich.
- Die Tests sind für eine nachfolgende Automatisierung weniger geeignet.

Beim Einsatz reaktiver oder heuristischer Vorgehensweisen verwenden Tester normalerweise erfahrungsbasierte Tests, um besser auf Ereignisse reagieren zu können als bei einer vorgeplanten Testvorgehensweise. Außerdem finden Testdurchführung und Testauswertung gleichzeitig statt. Einige strukturierte Vorgehensweisen beim erfahrungsbasierten Testen sind nicht durchgehend dynamisch, d.h. die Tests werden nicht vollkommen gleichzeitig erstellt und durchgeführt. Dies kann beispielsweise der Fall sein, wenn Tester schon vor der Testausführung durch intuitive Testfallermittlung bestimmte Aspekte des Testobjekts gezielt angehen.

Es ist zu beachten, dass nachfolgend bei den behandelten Testverfahren zwar einige Ideen bezüglich Überdeckung erwähnt werden, aber erfahrungsbasierte Verfahren haben grundsätzlich keine formalen Kriterien für die Überdeckung.

3.3.1 Intuitive Testfallermittlung

Bei der intuitiven Testfallermittlung (auch Error Guessing) nutzen Test Analysten ihre Erfahrung, um die Fehler zu erraten, die beim Entwurf und der Entwicklung des Codes möglicherweise gemacht wurden. Wenn die erwarteten Fehlhandlungen identifiziert wurden, bestimmt der Test Analyst die am besten geeigneten Methoden, um die resultierenden Fehlerzustände aufzudecken. Beispiel: Wenn der Test Analyst erwartet, dass die Software bei Eingabe eines ungültigen Passworts Fehlerwirkungen

zeigen wird, dann werden Tests durchgeführt, bei denen eine Vielzahl verschiedener Werte in das Passwortfeld eingegeben werden, um zu verifizieren, dass die Fehlhandlung tatsächlich erfolgt ist und zu einem Fehlerzustand geführt hat, der bei der Durchführung der Tests als Fehlerwirkung in Erscheinung tritt.

Die intuitive Testfallermittlung ist nicht nur als Testverfahren nützlich, sondern auch bei der Risikoanalyse, um potenzielle Fehlerauswirkungen zu identifizieren [Myers11].

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 den Umfang der bestehenden Testfälle zu erweitern. Die intuitive Testfallermittlung kann auch beim Testen einer neuen Version der Software effektiv eingesetzt werden, um diese auf häufige Fehlerzustände zu testen, noch bevor gründlicheres und skriptbasiertes Testen erfolgt.

Einschränkungen/Schwierigkeiten
Für die intuitive Testfallermittlung gelten die folgenden Einschränkungen und 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 angewendet wird, der sich gut mit den Fehlerarten auskennt, die bei der Art des zu testenden Codes häufig vorkommen.
- Das Verfahren wird häufig verwendet, ist aber häufig nicht dokumentiert und daher möglicherweise weniger reproduzierbar als andere Arten des Testens.
- Testfälle können ggf. dokumentiert sein, allerdings oft in einer Art und Weise, dass nur der Autor selbst versteht, was gemeint ist, und die Testfälle reproduzieren kann.

Überdeckung
Bei Verwendung einer Taxonomie wird die Überdeckung ermittelt, indem die Anzahl der getesteten Elemente durch die Gesamtzahl der Elemente der Taxonomie geteilt und als Prozentsatz angegeben wird. Ohne Taxonomie ist die Überdeckung durch Erfahrung und Wissen der Tester sowie die zur Verfügung stehende Zeit beschränkt. Die Menge der mit diesem Verfahren gefundenen Fehler hängt stark davon ab, wie gut der Tester die problematischen Bereiche bestimmen kann.

Fehlerarten
Typische Fehlerarten, die mit diesem Verfahren aufgedeckt werden, sind Fehlerzustände, die in der jeweiligen Taxonomie definiert sind, oder Fehlerzustände, die der Test Analyst intuitiv "errät", und die durch Black-Box-Testverfahren möglicherweise nicht gefunden würden.

3.3.2 Checklistenbasiertes Testen

Beim checklistenbasierten Testverfahren benutzen erfahrene Test Analysten eine abstrakte, allgemein gehaltene Liste von Punkten, welche beachtet, überprüft oder in Erinnerung gerufen werden müssen, oder eine Menge von Regeln oder Kriterien, gegen welche ein Testobjekt verifiziert werden muss. Diese Checklisten werden auf Grundlage von Standards, Erfahrungen und anderen Überlegungen zusammengestellt. Ein Beispiel für einen checklistenbasierten Test ist eine Checkliste mit Standards für Benutzungsschnittstellen, die als Grundlage für den Test einer Anwendung dient. In agilen Projekten können Checklisten aus den Abnahmekriterien einer User-Story erstellt werden.

Anwendbarkeit
Checklistenbasiertes Testen ist am effektivsten in Projekten mit einem erfahrenen Testteam, das die Software unter Test oder den Bereich, der von der Checkliste überdedckt wird, gut kennt. Um

beispielsweise eine Checkliste für die Benutzungsschnittstelle erfolgreich anzuwenden, könnte sich ein Test Analyst zwar mit dem Testen von Benutzungsschnittstellen auskennen, aber nicht unbedingt mit dem System unter Test. Da Checklisten abstrakt sind und auf detaillierte Schritte verzichten, wie sie normalerweise in Testfällen und Testabläufen zu finden sind, nutzt der Tester sein Wissen, um die Lücken zu füllen. Dadurch, dass die detaillierten Schritte weggelassen werden, ist der Wartungsaufwand der Checklisten gering und sie können für mehrere ähnliche Softwareversionen verwendet werden.

Checklisten eignen sich gut für Projekte, in denen Software schnell geändert und freigegeben wird. So lässt sich die Zeit für die Vorbereitung und Wartung der Testdokumentation reduzieren. Checklisten können in jeder Teststufe verwendet werden, auch für Regressionstests und Smoke-Tests.

Einschränkungen/Schwierigkeiten
Die abstrakt gehaltenen Checklisten können die Reproduzierbarkeit der Testergebnisse beeinträchtigen. Es ist möglich, dass mehrere Tester die Checklisten unterschiedlich interpretieren und unterschiedliche Ansätze verfolgen, um die Punkte der Checklisten zu erfüllen. Dies kann zu unterschiedlichen Ergebnissen führen, auch wenn dieselbe Checkliste verwendet wird. Die Folge ist eine breitere Überdeckung, allerdings wird die Reproduzierbarkeit manchmal dafür geopfert. Checklisten können auch zu einem übermäßigen Vertrauen und einer Überschätzung der erzielten Überdeckung führen, da der eigentliche Test von der Einschätzung des Testers abhängt. Checklisten können aus detaillierteren Testfällen oder Listen abgeleitet werden und tendieren dazu, mit der Zeit immer umfangreicher zu werden. Eine Wartung der Checklisten ist erforderlich, um sicherzustellen, dass sie die wichtigen Aspekte der zu testenden Software überdecken.

Überdeckung
Die Überdeckung kann ermittelt werden, indem man die Anzahl der getesteten Punkte der Checkliste durch die Gesamtzahl der Checklistenpunkte dividiert, und in Prozent angibt. Die Überdeckung ist so gut wie die Checkliste, die verwendet wird. Da die Checkliste jedoch abstrakt gehalten ist, variieren die Ergebnisse je nach dem welcher Test Analyst die Checkliste verwendet.

Fehlerarten
Zu den typischen Fehlerzuständen, die mit diesem Verfahren aufgedeckt werden, gehören solche, die beim Variieren von Daten, der Ablaufsequenz oder dem allgemeinen Workflow zu Fehlerwirkungen führen.

3.3.3 Exploratives Testen

Beim explorativen Testen lernen die Tester gleichzeitig etwas über das Testobjekt und dessen Fehlerzustände, planen die durchzuführenden Testaufgaben, entwerfen die Tests und führen sie aus, und berichten über die Testergebnisse. Die Tester passen die Testziele während der Testdurchführung dynamisch an und erstellen nur eine leichtgewichtige Dokumentation [Whittaker09].

Anwendbarkeit
Gutes explorativen Testen ist geplant, interaktiv und kreativ. Es erfordert wenig Dokumentation über das zu testende System und wird daher häufig angewendet, wenn Dokumentation nicht verfügbar oder für andere Testverfahren nicht geeignet ist. Exploratives Testen wird häufig als Ergänzung anderer Testverfahren eingesetzt und dient als Grundlage für die Erstellung zusätzlicher Testfälle. Exploratives Testen wird in agilen Projekten häufig eingesetzt, um User-Story-basierte Tests schnell und flexibel mit nur minimaler Dokumentation durchzuführen. Das Verfahren kann jedoch auch in Projekten eingesetzt werden, die einen sequenziellen Softwareentwicklungslebenszyklus nutzen.

Einschränkungen/Schwierigkeiten

Die Überdeckung kann bei explorativem Testen sporadisch sein, und die Reproduzierbarkeit der durchgeführten Tests kann schwierig sein. Als eine Methode des Managements des explorativen Testen können Test-Chartas erstellt werden, die die in einer Testsitzung zu überdeckenden Aufgaben und den verfügbaren Zeitrahmen (time-boxing) für das Testen festlegen. Am Ende einer oder mehrerer Testsitzung(en) hält der Testmanager eine Abschlussbesprechung, um die Ergebnisse der Tests zu sammeln und die Chartas für die nächsten Testsitzungen zu bestimmen.

Eine weitere Schwierigkeit bei explorativen Testsitzungen besteht darin, sie in einem Testmanagementsystem genau zu verfolgen. Dies geschieht manchmal durch die Erstellung von Testfällen, die eigentlich explorative Testsitzungen sind. Dadurch lassen sich die für das explorative Testen zur Verfügung gestellte Zeit und die geplante Überdeckung gemeinsam mit den anderen Testaufwänden verfolgen.

Die schwierige Reproduzierbarkeit beim explorativen Testen kann auch zu Problemen führen, wenn die Schritte zur Reproduktion einer Fehlerwirkung benötigt werden. Manche Organisationen nutzen die Mitschnittfunktion (Capture/Playback) eines Testautomatisierungswerkzeugs, um die durchgeführten Schritte eines explorativen Testers aufzuzeichnen. Dies liefert eine vollständige Aufzeichnung aller Aktivitäten während einer explorativen Testsitzung (oder während jeder anderen erfahrungsbasierten Testsitzung). Die Suche nach der tatsächlichen Ursache einer Fehlerwirkung kann mühsam sein, aber zumindest sind die durchgeführten Schritte allesamt aufgezeichnet.

Für die Erfassung explorativer Testsitzungen können auch andere Werkzeuge verwendet werden, allerdings zeichnen diese die erwarteten Ergebnisse nicht auf, weil sie die Interaktionen mit der grafischen Benutzungsoberfläche (GUI) nicht erfassen. In diesem Fall müssen die erwarteten Ergebnisse notiert werden, so dass bei Bedarf eine korrekte Analyse der Fehlerzustände durchgeführt werden kann. Es wird generell empfohlen während des explorativen Testens Notizen anzufertigen, um die Reproduzierbarkeit zu unterstützen, falls benötigt.

Überdeckung
Für bestimmte Aufgaben, Ziele und Arbeitsergebnisse können Test-Chartas erstellt werden. Dann werden die explorativen Testsitzungen geplant, um diese Kriterien zu erreichen. In der Charta kann auch bestimmt werden, worauf sich der Testaufwand konzentrieren soll, was innerhalb und außerhalb des Umfangs der Testsitzung liegt, und welche Ressourcen für die Durchführung der geplanten Tests vorzusehen sind. Eine Sitzung kann auf bestimmte Fehlerarten und auf andere potenzielle Problembereiche abzielen, die ohne die Formalität von skriptbasiertem Testen adressiert werden können.

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

3.3.4 Fehlerbasiertes Testverfahren

Bei fehlerbasierten Testverfahren dient die Art des gesuchten Fehlers als Basis für den Testentwurf, wobei die Tests systematisch von dem abgeleitet werden, was über die Fehlerart bekannt ist. Anders als bei Black-Box-Testverfahren, bei denen die Tests aus der Testbasis abgeleitet werden, werden beim fehlerbasierten Testverfahren die Tests aus Listen abgeleitet, die Fehler im Mittelpunkt haben. Diese Listen können nach Fehlerarten, Grundursachen, Symptomen von Fehlerwirkungen und anderen fehlerbezogenen Daten kategorisiert sein. Standardlisten gelten für mehrere Arten von Software und sind nicht produktspezifisch. Bei Verwendung dieser Standardlisten wird das branchenspezifische Wissen genutzt, um bestimmte Tests abzuleiten. Metriken über das Aufreten von Fehlerzuständen können über Projekte und sogar Organisationen hinweg verfolgt werden, wenn

branchenspezifische Listen befolgt werden. Die gebräuchlichsten Fehlerlisten sind jedoch organisations- ode projektspezifisch und beruhen auf besonderer Expertise oder Erfahrungen.

Beim fehlerbasierten Testen können auch Listen mit identifizierten Risiken und Risikoszenarien als Grundlage für zielgerichtetes Testen verwendet werden. Dieses Testverfahren ermöglicht es dem Test Analysten, bestimmte Arten von Fehlern gezielt zu adressieren, oder eine Liste bekannter und häufig auftretender Fehlerzustände bzw. Fehlerarten systematisch abzuarbeiten. Basierend auf diesen Informationen erstellt der Test Analyst die Testfälle und Testbedingungen, die den Fehlerzustand aufdecken werden, falls dieser vorhanden ist.

Anwendbarkeit
Fehlerbasiertes Testen kann in jeder beliebigen Teststufe eingesetzt werden, wird aber am häufigsten beim Systemtest eingesetzt.

Einschränkungen/Schwierigkeiten
Es gibt mannigfache Fehlertaxonomien, die auf bestimmte Testarten (z.B. Gebrauchstauglichkeits-tests) fokussiert sind. Es ist wichtig, eine Taxonomie auszuwählen, die auf die zu testende Software anwendbar ist. Für innovative Software gibt es möglicherweise (noch) gar keine Taxonomien. Manche Organisationen haben ihre eigenen Taxonomien mit wahrscheinlichen oder häufig auftretenden Fehlern zusammengestellt. Unabhängig davon, welche Taxonomie verwendet wird, muss vor Testbeginn die erwartete Überdeckung festgelegt werden.

Überdeckung
Das Verfahren liefert Überdeckungskriterien anhand derer sich bestimmen lässt, wann alle nützlichen Testfälle identifiziert wurden. Überdeckungselemente können je nach Fehlerliste Strukturelemente, Spezifikationselemente, Nutzungsszenarien oder eine beliebige Kombination dieser Elemente sein. In der Praxis sind die Überdeckungskriterien beim fehlerbasierten Testverfahren tendenziell weniger systematisch als bei Black-Box-Testverfahren, da nur allgemeine Regeln zur Überdeckung vorgegeben werden. Dabei ist die Entscheidung darüber, wann eine sinnvolle Überdeckung erreicht ist, eine Ermessensentscheidung. Wie bei anderen Verfahren bedeuten die Überdeckungskriterien nicht, dass die Testsuite vollständig ist, sondern dass dieses Testverfahren für die betrachteten Fehlerzustände keine weiteren sinnvollen Tests mehr vorschlägt.

Fehlerarten
Die Fehlerarten, die gefunden werden, hängen gewöhnlich von der verwendeten Taxonomie ab. Wenn beispielsweise eine Benutzungsschnittstellen-Fehlerliste verwendet wird, dann wird der Großteil der entdeckten Fehlerzustände wahrscheinlich auf die Benutzungsschnittstelle bezogen sein. Andere Fehlerzustände können jedoch durchaus als Nebenprodukt der spezifischen Tests gefunden werden.

3.4 Anwendung der bestgeeigneten Testverfahren

Black-Box- und erfahrungsbasierte Testverfahren sind am effektivsten, wenn sie zusammen eingesetzt werden. Erfahrungsbasierte Verfahren füllen die Lücken in der Überdeckung, die sich aus systematischen Schwächen von Black-Box-Testverfahren ergeben können.

Es gibt das einzige perfekte Verfahren, das für alle Situationen passt. Der Test Analyst muss die Vor- und Nachteile der einzelnen Verfahren verstehen und in der Lage sein, das beste Verfahren oder eine Menge von Verfahren für die jeweilige Situation auszuwählen. Dabei muss die Art des Projekts, der Zeitplan, der Zugang zu Informationen, die Fähigkeiten der Tester und weitere Faktoren berücksichtigt werden, die die Auswahl beeinflussen können.

Bei der Auswahl der bestgeeigneten Testverfahren sollten sich Test Analysten von den für die einzelnen Black-Box- und erfahrungsbasierten Testverfahren (siehe Abschnitt 3.2 bzw. 3.3)

beschriebenen Informationen leiten lassen, insbesondere von den Informationen in den Unterabschnitten “Anwendbarkeit”, "Einschränkungen/Schwierigkeiten" und "Überdeckung".

4. Das Testen von Softwarequalitätsmerkmalen - 180 min

Schlüsselbegriffe

Ästhetik der Benutzungsschnittstelle, Benutzererlebnis, Benutzerfehlerschutz, Erlernbarkeit, funktionale Angemessenheit, funktionale Eignung, funktionale Korrektheit, funktionale Vollständigkeit, Gebrauchstauglichkeit, Interoperabilität, Kompatibilität, Operabilität, Software-Gebrauchstauglichkeits-Messinventar, Website Analysis and MeasureMent Inventory, Zugänglichkeit

Lernziele für das Testen von Softwarequalitätsmerkmalen

4.1 Einführung

Keine Lernziele

4.2 Qualitätsmerkmale bei fachlichen Tests

TA-4.2.1 (K2) Erläutern, welche Testverfahren geeignet sind, um die funktionale Vollständigkeit, Korrektheit und Angemessenheit zu testen

TA-4.2.2 (K2) Die typischen Fehlerzustände bestimmen, auf die das Testen der funktionalen Vollständigkeits-, Korrektheits- und Angemessenheitsmerkmale abzielen soll

TA-4.2.3 (K2) Erläutern, in welcher Stufe des Softwareentwiclkungslebenszyklus die funktionalen Vollständigkeits-, Korrektheits- und Angemessenheitsmerkmale getestet werden sollten

TA-4.2.4 (K2) Die Ansätze erläutern, die geeignet wären, um sowohl die Umsetzung der Gebrauchstauglichkeitsanforderungen als auch die Erfüllung der Benutzererwartungen zu verifizieren und zu validieren

TA-4.2.5 (K2) Die Rolle von Test Analysten beim Testen der Interoperabilität erläutern, einschließlich der Identifizierung der Fehlerzustände, die damit aufgedeckt werden sollen

TA-4.2.6 (K2) Die Rolle von Test Analysten beim Testen der Übertragbarkeit erläutern, einschließlich der Identifizierung der Fehlerzustände, die damit aufgedeckt werden sollen

TA-4.2.7 (K4) Für eine vorgegebene Menge von Anforderungen die Testbedingungen bestimmen, die für die Verifizierung der funktionalen und/oder nicht-funktionalen Qualitätsmerkmale erforderlich sind, für die der Test Analyst zuständig ist

4.1 Einführung

Während das vorige Kapitel die verschiedenen Testverfahren beschreibt, die dem Tester zur Verfügung stehen, behandelt dieses Kapitel die Anwendung dieser Verfahren bei der Bewertung der Qualitätsmerkmale von Softwareanwendungen und -systemen.

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 behandelt [CTAL-TTA].

Die Beschreibung der Qualitätsmerkmale orientiert sich am ISO Standard 25010 [ISO25010]. Das Softwarequalitätsmodell nach ISO unterteilt die Produktqualität in verschiedene Qualitätsmerkmale, die jeweils weitere Untermerkmale haben können. Diese sind in der nachstehenden Tabelle aufgeführt, aus der ebenfalls hervorgeht, welche Merkmale/Untermerkmale im Test Analyst-Lehrplan und welche im Technical Test Analyst-Lehrplan behandelt werden:

Qualitätsmerkmal	Untermerkmale	Test Analyst	Technical Test Analyst
Funktionale Eignung	Funktionale Korrektheit, funktionale Angemessenheit, funktionale Vollständigkeit	X	
Zuverlässigkeit	Softwarereife, Fehlertoleranz, Wiederherstellbarkeit, Verfügbarkeit		X
Gebrauchstauglichkeit (Usability)	Erkennbare Angemessenheit, Erlernbarkeit, Operabilität, Ästhetik der Benutzungsschnitt-stelle, Benutzerfehlerschutz, Zugänglichkeit	X	
Performanz	Zeitverhalten, Ressourcennutzung, Kapazität		X
Wartbarkeit	Analysierbarkeit, Modifizierbarkeit, Testbarkeit, Modularität, Wiederverwendbarkeit		X
Übertragbarkeit	Anpassbarkeit, Installierbarkeit, Austauschbarkeit	X	X
IT-Sicherheit	Vertraulichkeit, Integrität, Nichtabstreitbarkeit, Zurechenbarkeit, Authenzität		X
Kompatibilität	Koexistenz		X
	Interoperabilität	X	

Die Zuständigkeitsbereiche von Test Analysten und Technischen Test Analysten sind in der obigen Tabelle zusammengefasst. Auch wenn sich die Arbeitszuteilung in verschiedenen Organisationen davon unterscheiden kann, halten sich die zugehörigen ISTQB®-Lehrpläne an diese Aufteilung.

Für alle in diesem Abschnitt behandelten Qualitätsmerkmale und -untermerkmale müssen die typischen Risiken erkannt werden, damit eine geeignete Teststrategie erarbeitet und dokumentiert werden kann. Beim Testen von Qualitätsmerkmalen müssen der Zeitplan im Softwareentwicklungslebenszyklus, benötigte Werkzeuge, Verfügbarkeit von Software und Dokumentation sowie technisches Fachwissen besondere Beachtung finden. Ohne eine Strategie zur Behandlung jedes einzelnen Merkmals und dessen spezifischen Testbedarfs wird der Tester möglicherweise nicht genügend Zeit für Planung, Vorbereitung und Durchführung der entsprechenden Tests im Zeitplan bekommen [Bath14]. Einige dieser Tests, z.B. Gebrauchstauglichkeitstests, können Spezialisten, umfangreiche Planung, spezielle Labore, bestimmte Werkzeuge, spezielle Testfähigkeiten und in den meisten Fällen einen erheblichen Zeitaufwand erfordern. In manchen Fällen können Gebrauchstauglichkeitstests von einer separaten Gruppe von Usability-Experten oder User Experience-Spezialisten durchgeführt werden.

Auch wenn Test Analysten nicht für solche Qualitätsmerkmale verantwortlich sind, die einen mehr technisch ausgerichteten Ansatz erfordern, sollten sie diese anderen Qualitätsmerkmale kennen und wissen, welche Bereiche sich beim Testen überschneiden. Zum Beispiel wird ein Produkt, das den Performanztest nicht besteht, wahrscheinlich auch den Gebrauchstauglichkeitstest nicht bestehen, wenn es zu langsam ist und die Benutzer es nicht effektiv nutzen können. In ähnlicher Weise wird ein Produkt, das Probleme mit der Interoperabilität bei einigen Komponenten hat, wahrscheinlich für den Übertragbarkeitstest nicht bereit sein, da die zugrunde liegenden Probleme bei einer Änderung der Umgebung weniger gut erkennbar sind.

4.2 Qualitätsmerkmale bei fachlichen Tests

Das Testen der funktionalen Eignung (funktionaler Test) ist ein primärer Aufgabenschwerpunkt von Test Analysten. Die funktionalen Tests sind darauf fokussiert, "was" das Produkt leistet. Die Testbasis beim Testen auf funktionale Eignung besteht in der Regel aus Elementen wie Anforderungsspezifikationen, Systemspezifikationen, spezifsches Wissen über den Geschäftsbereich oder einem implizit erwarteten Bedarf. Funktionale Tests unterscheiden sich je nach Teststufe, in der sie durchgeführt werden, und können auch vom Softwareentwicklungslebenszyklus beeinflusst werden. Bei einem funktionalen Test, der während des Integrationstests durchgeführt wird, wird beispielsweise die Funktionalität der Schnittstellenmodule getestet, die eine einzelne definierte Funktion implementieren. In der Systemteststufe beinhalten die funktionalen Tests das Testen der Gesamtfunktionalität des Systems. Bei Multisystemen werden mit funktionalen Tests hauptsächlich die gesamten integrierten Systeme Ende-zu-Ende getestet. Bei funktionalen Tests werden viele unterschiedliche Testverfahren eingesetzt (siehe Kapitel 3).

In einem agilen Umfeld umfasst das Testen der funktionalen Eignung gewöhnlich Folgendes:

- Testen der spezifischen Funktionalität (z.B. User-Stories), die in der jeweiligen Iteration zur Verfügung gestellt werden soll
- Regressionstests für die gesamte unveränderte Funktionalität

Zusätzlich zum Testen der funktionalen Eignung, das in diesem Abschnitt beschrieben wird, gibt es bestimmte weitere Qualitätsmerkmale, die in den Zuständigkeitsbereich von Test Analysten fallen, und als nicht-funktional gelten (d.h. sie konzentrieren sich darauf, “wie” das Testobjekt die Funktionalität liefert).

4.2.1 Testen der funktionalen Korrektheit

Beim Testen der funktionalen Korrektheit wird verifiziert, dass die spezifizierten oder impliziten Anforderungen eingehalten wurden; dies kann auch die Genauigkeit von Berechnungen einschließen. Funktionale Korrektheitstests nutzen viele der in Kapitel 3 beschriebenen Testverfahren. Oft dient die Spezifikation oder ein Altsystem als Testorakel. Das Testen der funktionalen Korrektheit kann in jeder Teststufe durchgeführt werden. Es zielt auf eine inkorrekte Handhabung von Daten oder Situationen ab.

4.2.2 Testen der funktionalen Angemessenheit

Das Testen der funktionalen Angemessenheit bewertet und validieret, ob sich eine Menge von Funktionen für die vorgesehenen Aufgaben eignen. Diese Tests können auf dem funktionalen Entwurf (z.B. Anwendungsfälle und/oder User-Stories) basieren. Tests auf funktionale Angemessenheit werden meist beim Systemtest durchgeführt, können aber auch in späteren Integrationsteststufen durchgeführt werden. Die Fehlerzustände, die bei diesen Tests aufgedeckt werden, sind Hinweise darauf, dass das System die Erfordernisse der Benutzer nicht in akzeptabler Weise erfüllen wird.

4.2.3 Testen der funktionalen Vollständigkeit

Beim Testen der funktionalen Vollständigkeit wird ermittelt, in wieweit die implementierte Funktionalität die spezifizierten Aufgaben und Benutzerziele abdeckt. Die Verfolgbarkeit zwischen Spezifikationselementen (z.B. Anforderungen, User-Stories, Use Cases) und der implementierten Funktionalität (z.B. Funktion, Unit, Workflow) ist essentiell, um die erforderliche Vollständigkeit zu ermitteln. Die Messung der funktionalen Vollständigkeit kann je nach Teststufe und/oder verwendetem Softwareentwicklungslebenszyklus variieren. Beispielsweise kann die funktionale Vollständigkeit für eine agile Iteration auf den implementierten User-Stories und Features basieren. Beim Systemintegrationstest kann die funktionale Vollständigkeit hauptsächlich auf die Überdeckung von Geschäftsszenarien fokussiert sein.

Die Ermittlung der funktionalen Vollständigkeit wird in der Regel durch Testmanagementwerkzeuge unterstützt, wenn der Test Analyst für die Verfolgbarkeit zwischen den Testfällen und den Elementen der funktionalen Spezifikation sorgt. Wenn der Grad an funktionaler Vollständigkeit geringer ist als erwartet, dann weist das darauf hin, dass das System nicht vollständig implementiert wurde.

4.2.4 Interoperabilitätstest

Interoperabilitätstests verifizieren den Informationsaustausch zwischen zwei oder mehr Systemen oder Komponenten. Die Tests konzentrieren sich auf die Fähigkeit der Systeme bzw. Komponenten, Informationen auszutauschen und die ausgetauschten Informationen anschließend zu nutzen. Das Testen auf Interoperabilität muss alle vorgesehenen Zielumgebungen überdecken (einschließlich Varianten der Hardware, Software, Middleware, des Betriebssystems 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 beschränkt werden. Für die Spezifikation von Interoperabilitätstest müssen die Kombinationen der vorgesehenen Zielumgebungen identifiziert, konfiguriert und dem Testteam zur Verfügung gestellt werden. Diese Umgebungen werden dann anhand einer Auswahl von funktionalen Testfällen getestet, die die verschiedenen Datenaustauschpunkte in den Umgebungen prüfen.

Interoperabilität betrifft das Zusammenwirken verschiedener Komponenten und Softwaresysteme. Software mit guten Interoperabilitätseigenschaften lässt sich mit anderen Systemen integrieren, ohne dass größere Änderungen nötig sind. Messen lässt sich die Interoperabilität durch die Anzahl notwendiger Änderungen und dem Aufwand für die Implementierung und das Testen dieser Änderungen.

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

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

Interoperabilitätstests sind besonders wichtig für:

- kommerzielle Standardsoftware und -werkzeuge
- Anwendungen, die auf einem System von Systemen basieren
- Systeme auf Basis des Internets der Dinge
- Web Services mit Konnektivität zu anderen Systemen

Das Testen auf Interoparabilität wird während der Komponenten- und Systemintegrationstests ausgeführt und fokussiert auf das Zusammenwirken der Komponente oder des Systems mit seiner Umgebung. Beim Systemintegrationstest wird mit diesen Tests geprüft, wie gut das fertig entwickelts System mit anderen Systemen zusammenwirkt. Da Systeme auf mehreren Ebenen interagieren

können, 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 erforderlichen Daten und Transaktionen zu erzeugen, die für den Datenaustausch erforderlich sind. Dabei ist zu beachten, dass nicht alle Interaktionen in den Anforderungsdokumenten klar spezifiziert sind. Viele dieser Interaktionen sind stattdessen in der Dokumentation der Systemarchitektur und des Systementwurfs spezifiziert. Daher muss der Test Analyst in der Lage und bereit sein, diese Dokumente zu untersuchen, um die Punkte des Datenaustausches zwischen den Systemen sowie zwischen dem System und seiner Umgebung zu bestimmen. Nur so kann sichergestellt werden, dass alle Punkte getestet werden. Testverfahren wie die Äquivalenzklassenbildung, Grenzwertanalyse, Entscheidungstabellen, Zustandsdiagramme, Anwendungsfälle und paarweises Testen sind allesamt im Interoperabilitätstest anwendbar. Zu den typischen Fehlerzuständen, die aufgedeckt werden, gehört der inkorrekte Datenaustausch zwischen interagierenden Komponenten.

4.2.5 Evaluierung der Gebrauchstauglichkeit (Usability)

Test Analysten sind häufig in der Lage, die Bewertung der Gebrauchstauglichkeit zu koordinieren und zu unterstützen. Dies kann das Spezifizieren von Gebrauchstauglichkeitstests betreffen, oder die Moderation von Tests, die mit Benutzern durchgeführt werden. Um diese Aufgaben effektiv durchzuführen muss ein Test Analyst die wichtigsten Aspekte, Ziele und Vorgehensweisen dieser Art von Tests verstehen. Bitte beachten Sie hierzu den ISTQB Specialist Foundation Level-Lehrplan Usability Testing [ISTQB_FL_UT], der weitere Details enthält, die über die Beschreibung in diesem Abschnitt hinausgehen.

Es ist wichtig zu verstehen, warum Benutzer bei der Nutzung des Systems Schwierigkeiten oder kein positives Benutzererlebnis (engl. user experience, UX) haben könnten (z.B. bei der Nutzung von Unterhaltungssoftware). Dabei gilt es zu verstehen, dass die Benutzer eines Systems sehr unterschiedlichen Personengruppen angehören können, angefangen von IT-Experten bis hin zu Kindern oder Menschen mit Behinderung.

4.2.5.1 Aspekte der Gebrauchstauglichkeit

In diesem Abschnitt werden die folgenden drei Aspekte behandelt:

- Gebrauchstauglichkeit (Usability)
- Benutzererlebnis (UX)
- Zugänglichkeit

Gebrauchstauglichkeit

Das Testen der Gebrauchstauglichkeit zielt auf Fehlerzustände der Software ab, die Benutzer bei der Ausführung von Aufgaben über die Benutzungsschnittstelle beeinträchtigen. Solche Fehlerzustände können die Fähigkeit von Benutzern beeinträchtigen, ihre Ziele effektiv, effizient und zu ihrer Zufriedenheit zu erreichen. Probleme mit der Gebrauchstauglichkeit können zu Verwirrung, Fehlhandlungen oder Verzögerungen führen, oder dazu, dass Benutzer eine Aufgabe überhaupt nicht ausführen können.

Im Folgenden sind die einzelnen Untermerkmale [ISO 25010] der Gebrauchstauglichkeit aufgeführt:

- erkennbare Angemessenheit (d.h. Verständlichkeit) – Eigenschaften der Software, die beeinflussen, welchen Aufwand die Benutzer leisten müssen, um das logische Konzept und dessen Anwendbarkeit zu erkennen
- Erlernbarkeit – Eigenschaften der Software, die beeinflussen, welchen Aufwand die Benutzer leisten müssen, um die Anwendung zu erlernen
- Operabilität - Eigenschaften der Software, die beeinflussen, welchen Aufwand die Benutzer leisten müssen, um Aufgaben effektiv und effizient zu erledigen
- Ästhetik der Benutzungsschnittstelle (d.h. Attraktivität) - visuelle Eigenschaften der Software, die vom Benutzer geschätzt werden

- Benutzerfehlerschutz - Grad, zu dem ein System den Benutzer vor Fehlhandlungen schützt
- Zugänglichkeit (siehe unten)

Benutzererlebnis (User Experience)
Bei der Evaluierung des Benutzererlebnisses geht es um das gesamte Benutzererlebnis mit dem Testobjekt, nicht nur um die direkte Interaktion. Dies ist besonders wichtig bei Testobjekten, bei denen Faktoren wie Vergnügen und Benutzerzufriedenheit für den Geschäftserfolg entscheidend sind.

Typische Faktoren, die das Benutzererlebnis beeinflussen, sind:
- Markenimage (d.h. das Vertrauen der Nutzer in den Hersteller)
- interaktives Verhalten
- Hilfsbereitschaft des Produktes, einschließlich Hilfesystem, Support und Schulung

Zugänglichkeit
Es ist wichtig, die Zugänglichkeit der Software (auch Barrierefreiheit genannt) zu berücksichtigen, damit Menschen mit besonderen Bedürfnissen oder Einschränkungen die Software nutzen können. Dies schließt Menschen mit Behinderungen mit ein. Beim Zugänglichkeitstest sollten die einschlägigen Normen, wie die Richtlinie für barrierefreie Webinhalte (engl. Web Content Accessibility Guidelines bzw. WCAG), und die einschlägige Gesetzgebung, wie die Disability Discrimination Acts (Nordirland, Australien), der Equality Act 2010 (England, Schottland, Wales), Section 508 (US), oder die Barrierefreie-Informationstechnik-Verordnung (BITV) (Deutschland) berücksichtigt werden. Wie die Gebrauchstauglichkeit muss auch die Zugänglichkeit beim Entwurf berücksichtigt werden. Das Testen erfolgt oft in den Integrationsteststufen und wird im Systemtest und im Abnahmetest fortgesetzt. Fehlerzustände werden gewöhnlich dann festgestellt, wenn die Software nicht den für die Software spezifizierten Vorschriften oder Standards entspricht.

Typische Maßnahmen zur Verbesserung der Zugänglichkeit richten sich auf die Möglichkeiten, die die Software Benutzern mit Behinderungen für die Interaktion mit der Anwendung bietet. Dazu gehören folgende Maßnahmen:
- Spracherkennung für Eingaben
- Sicherstellen, dass dem Benutzer äquivalente Textalternativen für alle angezeigten Nicht-Text-Inhalte zur Verfügung gestellt werden
- Skalierbarkeit der Textgröße ohne dass Inhalte oder Funktionalität verloren gehen

Leitlinien für die Zugänglichkeit sind für den Test Analysten Informationsquellen und Checklisten, die für das Testen verwendet werden können (Beispiele für Richtlinien für Zugänglichkeit sind in [ISTQB_FL_UT] enthalten). Darüber hinaus stehen Werkzeuge und Browser-Plugins zur Verfügung, die den Testern helfen, Probleme mit der Zugänglichkeit von Softwareprodukten zu erkennen, wie z.B. eine schlechte Farbauswahl in Webseiten, die gegen die Richtlinien für Farbenblindheit verstößt.

4.2.5.2 Ansätze für die Evaluierung der Gebrauchstauglichkeit
Gebrauchstauglichkeit, Benutzererlebnis und Zugänglichkeit können durch einen oder mehrere der folgenden Ansätze getestet werden:
- Gebrauchstauglichkeitstests
- Reviews auf Gebrauchstauglichkeit
- Gebrauchstauglichkeitsumfragen und Fragebögen

Gebrauchstauglichkeitstest
Gebrauchstauglichkeitstests bewerten die Einfachheit, mit der die Benutzer das System nutzen bzw. erlernen können, um damit spezifizierte Ziele in bestimmten Anwendungskontexten zu erreichen. Beim Gebrauchstauglichkeitstest wird folgendes gemessen:

- Effektivität – Die Fähigkeit des Testobjekts die Benutzer in die Lage zu versetzen, bestimmte Ziele in einem spezifizierten Anwendungskontext mit Genauigkeit und Vollständigkeit zu erreichen
- Effizienz – Die Fähigkeit des Testobjekts die Benutzer in die Lage zu versetzen, in einem spezifizierten Anwendungskontext mit einem angemessenen Aufwand bestimmte Ziele effektiv zu erreichen
- Zufriedenheit – Die Fähigkeit des Testobjekts, die Benutzer in einem bestimmten Nutzungskontext zufriedenzustellen

Es ist zu beachten, dass der Test Analyst Entwurf und Spezifikation von Gebrauchstauglichkeitstests häufig in Zusammenarbeit mit Testern durchführt, die auf den Gebrauchstauglichkeitstest spezialisiert sind, und mit Gebrauchstauglichkeitsexperten, die sich mit dem menschzentrierten Gestaltungsprozess auskennen (siehe [ISTQB_FL_UT] für Details).

Reviews auf Gebrauchstauglichkeit
Inspektionen und Reviews aus der Gebrauchstauglichkeitsperspektive tragen dazu bei, die Beteiligung von Benutzern zu erhöhen. Dies kann kosteneffektiv sein, weil Gebrauchstauglichkeitsprobleme in den Anforderungsspezifikationen und im Entwurf frühzeitig im Softwareentwicklungslebenszyklus gefunden werden. Mit der heuristischen Evaluierung (systematische Prüfung einer Benutzungsschnittstelle oder deren Entwurf auf Gebrauchstauglichkeit) können Gebrauchstauglichkeitsprobleme im Entwurf aufgedeckt werden, damit diese im iterativen Entwurfsprozess bearbeitet werden können. Dabei untersucht ein kleines Gutachterteam die Schnittstelle und beurteilt deren Konformität mit anerkannten Grundsätzen der Gebrauchstauglichkeit (die "Heuristiken"). Reviews sind effektiver, wenn die Benutzungsschnittstelle sichtbar ist. Zum Beispiel sind Screenshots in der Regel einfacher zu verstehen und zu interpretieren als die Beschreibung einer bestimmten Bildschirmfunktionalität in Textform. Die Visualisierung ist für ein angemessenes Review der Dokumentation auf Gebrauchstauglichkeit wichtig.

Gebrauchstauglichkeitsumfragen und Fragebögen
Mit Umfragen und Fragebögen lassen sich Erkenntnisse und Feedback über das Verhalten der Benutzer bei der Systemnutzung sammeln. Standardisierte und öffentlich zugängliche Fragenkataloge wie das Software-Gebrauchstauglichkeits-Messinventar (Software Usability Measurement Inventory, SUMI) und Website Analysis and Measurement Inventory (WAMMI) ermöglichen ein Benchmarking gegen eine Datenbank mit früheren Gebrauchstauglichkeitsmessungen. SUMI liefert darüber hinaus konkrete Messgrößen für die Gebrauchstauglichkeit, die als Ausgangs-/Abnahmekriterien verwendet werden können.

4.2.6 Übertragbarkeitstest

Übertragbarkeitstests beziehen sich auf den Grad, zu dem eine Softwarekomponente oder ein System entweder initial oder von einer bestehenden Umgebung in eine Zielumgebung versetzt werden kann.

Die Klassifikation der Produktqualitätsmerkmale nach ISO 25010 enthält die folgenden Untermerkmale für Übertragbarkeit (bzw. Portabilität):
- Installierbarkeit
- Anpassbarkeit
- Austauschbarkeit

Die Aufgaben, die entsprechenden Risiken zu identifizieren und Übertragbarkeitstests zu entwerfen, teilen sich Test Analysten und Technical Test Analysten (siehe [ISTQB_ALTTA_SYL] Abschnitt 4.7).

4.2.6.1 Installationstest
Installationstests beziehen sich auf die Software und die schriftlich dokumentierten Verfahren, die zur Installation und Deinstallation der Software in der Zielumgebung verwendet werden.

Die typischen Testziele, auf die sich Test Analysten konzentrieren, sind:

- Validieren, dass verschiedene Konfigurationen der Software erfolgreich installiert werden können. Falls eine große Anzahl von Parametern konfiguriert werden kann, kann der Test Analyst mit Hilfe des paarweisen Testens die Anzahl der getesteten Parameterkombinationen reduzieren und sich auf bestimmte Konfigurationen konzentrieren, die von besonderem Interesse sind (z.B. die häufig verwendeten Konfigurationen).
- Testen der funktionalen Korrektheit von Installations- und Deinstallationsverfahren.
- Durchführen von funktionalen Tests nach einer Installation oder Deinstallation, um eventuell eingeschleuste Fehlerzustände zu erkennen (z.B. falsche Konfigurationen, nicht verfügbare Funktionen).
- Erkennen von Gebrauchstauglichkeitsproblemen bei Installations- und Deinstallationsprozeduren (z.B. überprüfen, ob die Benutzer bei Durchführung der Prozedur verständliche Anweisungen und Feedback/Fehlermeldungen erhalten).

4.2.6.2 Anpassbarkeitstest

Der Anpassbarkeitstest prüft, ob eine bestimmte Anwendung in allen vorgesehenen Zielumgebungen (Hardware, Software, Middleware, Betriebssystem etc.) korrekt funktionieren kann. Der Test Analyst unterstützt das Testen auf Anpassbarkeit, indem er Tests entwirft, die Kombinationen der vorgesehenen Zielumgebungen identifizieren (z.B. Versionen verschiedener unterstützter mobiler Betriebssysteme, verschiedene Browserversionen, die verwendet werden können). Diese Umgebungen werden dann anhand einer Auswahl funktionaler Testfälle getestet, die die verschiedenen in der Umgebung vorhandenen Komponenten ausführen.

4.2.6.3 Austauschbarkeitstest

Der Austauschbarkeitstest konzentriert sich auf die Fähigkeit, Softwarekomponenten oder -versionen innerhalb eines Systems gegen andere austauschen zu können. Dies kann insbesondere bei Systemarchitekturen relevant sein, die auf dem Internet der Dinge basieren, wo der Austausch unterschiedlicher Hardwaregeräte und/oder Softwareinstallationen häufig vorkommt. Beispiel: Ein Hardwaregerät, das in einem Warenlager zur Erfassung und Kontrolle der Lagerbestände verwendet wird, kann durch ein moderneres Hardwaregerät (z.B. mit einem besseren Scanner) ersetzt werden, oder es gibt ein Software-Upgrade für die installierte Software, das es ermöglicht, automatisch Nachbestellungen für Lagerbestände an das System eines Lieferanten zu senden.

Austauschbarkeitstests können vom Test Analysten parallel zu funktionalen Integrationstests durchgeführt werden, wenn mehr als eine alternative Komponente zur Integration in das Gesamtsystem zur Verfügung steht.

5. Reviews - 120 min

Schlüsselbegriffe
checklistenbasiertes Review

Lernziele für Reviews

5.1 Einführung
Keine Lernziele

5.2 Checklisten in Reviews verwenden
TA-5.2.1 (K3) Probleme in einer Anforderungsspezifikation anhand der im Lehrplan enthaltenen Checklisten identifizieren
TA-5.2.2 (K3) Probleme in einer User-Story anhand der im Lehrplan enthaltenen Checklisten identifizieren

5.1 Einführung

Zu einem erfolgreichen Reviewprozess gehören die Planung, Durchführung und Nachbereitung von Reviews. Test Analysten müssen aktiv am Reviewprozess beteiligt sein und ihre individuelle Sichtweise einbringen. Wenn sie richtig durchgeführt werden, dann leisten Reviews nicht nur den größten einzelnen, sondern auch den kosteneffektivsten Beitrag zur ausgelieferten Gesamtqualität.

5.2 Checklisten in Reviews verwenden

Das checklistenbasierte Review ist das häufigste Verfahren, das Test Analysten zum Review der Testbasis verwenden. Checklisten werden bei Reviews verwendet, damit die Teilnehmer angehalten sind, bestimmte Punkte während des Reviews zu überprüfen. Checklisten können auch dazu beitragen das Review zu entpersonalisieren (z.B. mit der Aussage, "Dies ist dieselbe Checkliste, die für alle Reviews verwendet wird, nicht nur für das vorliegende Arbeitsergebnis").

Checklistenbasierte Reviews können allgemein gehalten sein und für alle Reviews verwendet werden, oder sie können sich zielgerichtet mit bestimmten Qualitätsmerkmalen, Themenbereichen oder Ergebnistypen befassen. Beispiel: Mit einer generischen Checkliste können die allgemeinen Eigenschaften eines Dokuments verifiziert werden, z.B. ob das Dokument eine eindeutige Kennung hat, dass es keine offenen Punkte enthält, dass Formatierung und ähnliche Elemente korrekt bzw. konform sind. Eine spezifische Checkliste für ein Anforderungsdokument könnte Prüfpunkte dazu enthalten, ob die Begriffe "soll" und "sollte" richtig verwendet werden, ob jede Anforderung testbar ist usw.

Das Format, in dem die Anforderungen vorliegen, kann auch die Art der zu verwendenden Checkliste beeinflussen. Für ein Anforderungsdokument, das in Textform vorliegt, werden andere Reviewkriterien verwendet als für eines, das auf Diagrammen basiert.

Checklisten können auch auf einen bestimmten Aspekt ausgerichtet sein, wie zum Beispiel:

- auf die spezifischen Kompetenzen von Programmierern/Systemarchitekten oder Testern
  - Für Test Analysten sind am ehesten Checklisten geeignet, die auf die spezifischen Kenntnisse von Testern ausgerichtet sind.
  - Diese Checklisten können die in den Abschnitten 5.2.1 und 5.2.2 beschriebenen Punkte enthalten.
- auf eine bestimmte Risikostufe (z.B. bei sicherheitskritischen Systemen) - Solche Checklisten enthalten in der Regel die für die Risikostufe erforderlichen spezifischen Informationen.
- auf ein bestimmtes Testverfahren - Diese Checklisten sind auf die für ein bestimmtes Verfahren benötigten Informationen fokussiert (z.B. Regeln, die in einer Entscheidungstabelle dargestellt werden sollen).
- auf eine bestimmte Art von Spezifikation, wie z.B. eine Anforderung, ein Anwendungsfall oder eine User-Story – Diese Checklisten werden in den folgenden Abschnitten behandelt. Sie haben im Allgemeinen einen anderen Schwerpunkt als Checklisten, die von Technical Test Analysten für Code- oder Architekturreviews verwendet werden.

5.2.1 Reviews von Anforderungen

Checklisten, die an den Anforderungen orientiert sind, können beispielsweise die folgenden Elemente enthalten:

- Quelle der Anforderung (z.B. Person, Abteilung)
- Testbarkeit der einzelnen Anforderungen
- Abnahmekriterien für die einzelnen Anforderungen
- Verfügbarkeit einer Aufrufstruktur für Anwendungsfälle (falls zutreffend)

- eindeutige Identifizierung der einzelnen Anforderungen/Anwendungsfälle/User Stories
- Versionierung der einzelnen Anforderungen/Anwendungsfälle/User Stories
- Verfolgbarkeit jeder einzelnen Anforderung zu den Anforderungen des Fachbereichs/Marketings
- Verfolgbarkeit zwischen Anforderungen und/oder Anwendungsfällen (falls zutreffend)
- Verwendung einer konsistenten Terminologie (z.B. Verwendung eines Glossars)

Es ist zu beachten, dass wenn eine Anforderung nicht testbar ist, ein Fehlerzustand in dieser Anforderung vorliegt. Nicht testbar bedeutet, dass die Anforderung so spezifiziert ist, dass der Test Analyst nicht bestimmen kann, wie sie zu testen ist. Beispiel: Eine Anforderung, die besagt "Die Software sollte sehr benutzerfreundlich sein", ist nicht testbar. Wie soll ein Test Analyst bestimmen, ob die Software nur "benutzerfreundlich" oder "sehr benutzerfreundlich" ist? Wenn die Anforderung stattdessen festlegt "Die Software muss den Gebrauchstauglichkeitsstandards entsprechen, die im Dokument Gebrauchstauglichkeitsrichtlinie, Version xxx, festgelegt sind", dann ist die Anforderung testbar, vorausgesetzt das spezifizierte Dokument existiert. Hierbei handelt es sich außerdem um eine übergreifende Anforderung, die jedes Element der Schnittstelle betrifft. In diesem Fall könnte diese eine Anforderung bei einer nichttrivialen Anwendung zu vielen einzelnen Testfällen führen. Außerdem ist die Verfolgbarkeit von dieser Anforderung bzw. von der festgelegten Gebrauchstauglichkeitsrichtlinie zu den Testfällen kritisch. Falls es nämlich Änderungen der referenzierten Gebrauchstauglichkeitsrichtlinie gibt, müssen alle betroffenen Testfälle überprüft und bei Bedarf aktualisiert werden.

Eine Anforderung ist auch dann nicht testbar, wenn der Tester nicht in der Lage ist festzustellen, ob der Test besteht oder fehlschlägt, oder wenn kein Test entwickelt werden kann, der bestehen oder fehlschlagen kann. Ein Beispiel für eine nicht testbare Anforderung ist: "Das System muss 100% der Zeit zur Verfügung stehen, 24 Stunden pro Tag, 7 Tage pro Woche, 365 (bzw. 366) Tage im Jahr".

In einer einfachen Checkliste[1] für Reviews von Anwendungsfällen könnten folgende Fragen enthalten sein:

- Ist das Standardverhalten (Hauptszenario) genau spezifiziert?
- Sind alle alternativen Verhalten (Szenarien) identifiziert, einschließlich der Fehlerbehandlungen?
- Sind die Meldungen der Benutzungsschnittstelle spezifiziert?
- Gibt es nur ein Standardverhalten? (Das sollte der Fall sein, denn sonst gäbe es mehrere Anwendungsfälle.)
- Ist jedes Verhalten testbar?

5.2.2 Reviews von User-Stories

In agilen Projekten liegen die Anforderungen in der Regel als User-Stories vor. Diese User-Stories repräsentieren kleine Inkremente mit demonstrierbarer Funktionalität. Während es sich bei Anwendungsfällen um eine Benutzertransaktion handelt, die mehrere Funktionalitätsbereiche durchläuft, ist eine User-Story ein isolierteres Feature, deren Umfang im Allgemeinen durch die Zeit bestimmt wird, die für ihre Entwicklung notwendig ist. Eine Checkliste[2] für eine User-Story könnte folgende Fragen enthalten:

- Ist die User-Story für die vorgesehene Iteration/Sprint angemessen?
- Ist die User-Story aus der Sicht der Person geschrieben, die sie anfordert?
- Sind die Abnahmekriterien spezifiziert und testbar?
- Ist das Feature klar spezifiziert?
- Ist die User-Story unabhängig von anderen User-Stories?

[1] Die Prüfungsfrage liefert zur Beantwortung einen Teil einer Anwendungsfall-Checkliste
[2] Die Prüfungsfrage liefert zur Beantwortung einen Teil einer User-Story-Checkliste

- Ist die User-Story priorisiert?
- Ist die User-Story nach dem üblichen Format erstellt: Als ein <Typ von Benutzer>, möchte ich <ein Ziel>, so dass <ein Grund>. [Cohn04]

Falls eine User-Story eine neue Benutzungsschnittstelle definiert, dann wäre es sinnvoll, sowohl eine allgemeine User-Story-Checkliste wie im obigen Beispiel als auch eine detaillierte Checkliste für die Benutzungsschnittstelle zu verwenden.

5.2.3 Checklisten anpassen

Checklisten können an folgende Aspekte individuell angepasst werden:

- Organisation/Unternehmen (z.B. Berücksichtigung von Unternehmensrichtlinien, -standards, -konventionen, rechtlichen Einschränkungen)
- Projekt-/Entwicklungsaufwand (z.B. Schwerpunkt, technische Standards, Risiken)
- Reviewgegenstand (beispielsweise können Code-Reviews auf eine bestimmte Programmiersprache ausgerichtet sein)
- Risikostufe des Reviewgegenstands
- anzuwendende Testverfahren

Gute Checklisten decken Probleme auf, und sie regen Diskussionen über andere Aspekte an, auf die die Checkliste möglicherweise nicht explizit verweist. Checklisten zu kombinieren ist ein sehr gutes Mittel, damit das Review die beste Qualität für das Arbeitsergebnis erzielt. Durch die Verwendung von Standard-Checklisten beim Review, wie diejenigen, auf die im Foundation Level-Lehrplan verwiesen wird, und die Entwicklung eigener, unternehmensspezifischer Checklisten, wie diejenigen, die oben beschrieben wurden, können Test Analysten einen effektiven Beitrag bei Reviews leisten.

Weitere Informationen über Reviews und Inspektionen, siehe [Gilb93] und [Wiegers03]. Weitere Beispiele für Checklisten, siehe Abschnitt 7.4.

6. Testwerkzeuge und Testautomatisierung - 90 min

Schlüsselbegriffe
schlüsselwortgetriebenes Testen, Testdatenvorbereitung, Testdurchführung, Testentwurf, Testskript

Lernziele für Testwerkzeuge und Testautomatisierung

6.1 Einführung
Keine Lernziele

6.2 Schlüsselwortgetriebene Testautomatisierung
TA-6.2.1 (K3) Für ein bestimmtes Szenario die Aktivitäten des Test Analysten in einem schlüsselwortgetriebenen Testautomatisierungsprojekt bestimmen

6.3 Arten von Testwerkzeugen
TA-6.3.1 (K2) Die Verwendung von verschiedenen Arten von Testwerkzeugen erläutern, die beim Testentwurf, der Testdatenvorbereitung und bei der Testdurchführung eingesetzt werden

6.1 Einführung

Testwerkzeuge können die Effizienz und Genauigkeit des Testens erheblich verbessern. Die von Test Analysten verwendeten Testwerkzeuge und Automatisierungsansätze werden in diesem Kapitel beschrieben. Es ist zu beachten, dass Test Analysten bei der Erstellung von Testautomatisierungslösungen mit Entwicklern, Testautomatisierungsentwicklern und Technischen Test Analysten zusammenarbeiten. Sie wirken insbesondere bei der schlüsselwortgetriebenen Testautomatisierung mit und können ihre Erfahrung mit der Fachlichkeit und der Systemfunktionalität beisteuern.

Für weitere Informationen zum Thema Testautomatisierung und zur Rolle des Testautomatisierungsentwicklers siehe ISTQB-Lehrplan Advanced Level Testautomatisierungsentwickler [ISTQB_ALTAE_SYL].

6.2 Schlüsselwortgetriebene Testautomatisierung

Das schlüsselwortgetriebene Testen ist einer der wichtigsten Ansätze zur Testautomatisierung und bezieht den Test Analysten in die Bereitstellung der wichtigsten Inputs ein: Schlüsselwörter und Testdaten.

Schlüsselwörter (manchmal auch als Aktionswörter bezeichnet) werden meist, aber nicht ausschließlich, verwendet, um logische geschäftliche Interaktionen mit einem System darzustellen (z.B. "Auftrag stornieren"). Jedes Schlüsselwort bezeichnet dabei normalerweise eine Reihe detaillierter Interaktionen zwischen einem Akteur und dem System unter Test. Testfälle werden als Sequenzen von Schlüsselwörtern (zusammen mit den relevanten Testdaten) spezifiziert. [Buwalda02]

Beim automatisierten Testen werden die einzelnen Schlüsselwörter als ein oder mehrere auszuführende Testskripte implementiert. Die Werkzeuge lesen die Testfälle ein, die in einer Abfolge von Schlüsselwörtern verfasst sind, und rufen die entsprechenden Testskripte auf, welche die Funktionalität der jeweiligen Schlüsselwörter implementieren. Die Testskripte sind sehr modular implementiert, damit sie den einzelnen Schlüsselwörtern einfach zugeordnet werden können. Für die Implementierung dieser modularen Skripte sind Programmierkenntnisse erforderlich.

Die wichtigsten Vorteile der schlüsselwortgetriebenen Testautomatisierung sind:

- Schlüsselwörter, die sich auf eine bestimmte Anwendung oder einen bestimmten Geschäftsbereich beziehen, können von Fachexperten definiert werden. Das kann die Ausarbeitung der Testfallspezifikationen effizienter gestalten.
- Personen, die eher Fachexpertise besitzen, können von der automatischen Testfallausführung profitieren (sobald die Schlüsselwörter als Testskripte implementiert wurden) ohne den zugrunde liegenden Automatisierungscode verstehen zu müssen.
- Testfälle, die modular unter Verwendung von Schlüsselwörtern erstellt wurden, sind für Testautomatisierungsentwickler effizienter wartbar, falls es Änderungen der Funktionalität und der Schnittstelle zur getesteten Software gibt [Bath14].
- Testfallspezifikationen sind unabhängig von ihrer Implementierung.

In der Regel sind Test Analysten für die Erstellung und Pflege der Schlüsselwort-/Aktionswortdaten zuständig. Sie müssen verstehen, dass für die Implementierung der Schlüsselwörter stets noch die entsprechenden Skripte entwickelt werden müssen. Wenn die zu verwendenden Schlüsselwörter und Daten definiert sind, erstellen Technical Test Analysten oder Testautomatisierungsentwickler aus den Geschäftsprozess-Schlüsselwörtern und grundlegenden Aktionen die automatisierten Testskripte.

Während eine schlüsselwortgetriebene Testautomatisierung gewöhnlich im Systemtest eingesetzt wird, kann die Erstellung des Codes bereits während des Testentwurfs beginnen. Bei iterativen

Vorgehensweisen, insbesondere bei kontinuierlicher Integration/kontinuierlichem Deployment (continuous integration/continuous deployment), ist die Entwicklung der automatisierten Tests ein kontinuierlicher Prozess.

Nachdem die Schlüsselwörter und Eingabedaten erstellt wurden, sind die Test Analysten für die Ausführung der schlüsselwortgetriebenen Testfälle und die Analyse der aufgetretenen Fehlerwirkungen zuständig.

Wenn eine Anomalie entdeckt wird, muss der Test Analyst bei der Untersuchung der Ursache der Fehlerwirkung helfen, um herauszufinden, ob das Problem durch die Schlüsselwörter, die Eingabedaten, die automatisierten Testskripte selbst oder durch die getestete Anwendung verursacht wird. Der erste Schritt bei der Fehlersuche ist üblicherweise die manuelle Durchführung des entsprechenden Tests mit denselben Daten, um herauszufinden, ob die Fehlerwirkung die Anwendung selbst betrifft. Wenn beim manuellen Testen keine Fehlerwirkung auftritt, dann sollte der Test Analyst die Abfolge der Tests prüfen, die zur Fehlerwirkung geführt hat. So lässt sich feststellen, ob der Ursprung des Problems zu einem früheren Zeitpunkt erfolgte (vielleicht durch inkorrekte Eingabedaten), sich aber erst später bei der Verarbeitung auswirkte. Wenn der Test Analyst die Ursache der Fehlerwirkung nicht bestimmen kann, sollten die Informationen zur Fehleranalyse an den Technischen Test Analysten oder Entwickler zur weiteren Analyse übergeben werden.

6.3 Arten von Testwerkzeugen

Ein Großteil der Arbeit eines Test Analysten erfordert den effektiven Einsatz von Werkzeugen. Diese Effektivität wird durch folgende Faktoren verstärkt:

- durch das Wissen, welche Werkzeuge einzusetzen sind
- durch das Wissen, dass Werkzeuge die Effizienz des Test Analysten steigern können (z.B. indem sie helfen, in der verfügbaren Zeit eine bessere Überdeckung zu erzielen)

6.3.1 Testentwurfswerkzeuge

Testentwurfswerkzeuge werden verwendet, um die Erzeugung von Testfällen und Testdaten für das Testen zu unterstützen. Dazu verwenden die Werkzeuge bestimmte Formate der Anforderungsspezifikation, Modelle (z.B. UML) oder Eingaben des Test Analysten. Testentwurfswerkzeuge werden häufig so konzipiert und erstellt, dass sie bestimmte Formate und bestimmte Werkzeuge unterstützen, wie z.B. bestimmte Anforderungsmanagementwerkzeuge.

Testentwurfswerkzeuge können Informationen liefern, anhand derer der Test Analyst bestimmen kann, welche Testverfahren erforderlich sind, um die angestrebte Überdeckung, das Vertrauen in das System oder die Maßnahmen zur Minderung des Produktrisikos zu erzielen. Klassifikationsbaumeditoren generieren (und zeigen) beispielsweise die Menge der Kombinationen, die benötigt werden, um eine vollständige Überdeckung auf Grundlage eines vordefinierten Überdeckungskriteriums zu erzielen. Diese Informationen dienen dem Test Analysten dazu, die auszuführenden Testfälle festzulegen.

6.3.2 Werkzeuge für die Testdatenvorbereitung

Werkzeuge für die Testdatenvorbereitung bieten die folgenden Nutzen:

- Analyse eines Dokuments (z.B. Anforderungsdokument) oder sogar von Quellcode, um die beim Testen benötigten Daten zur Erzielung einer vordefinierten Überdeckung zu bestimmen.
- Maskierung oder Anonymisierung von Daten aus einem Produktionssystem, so dass alle personenbezogenen Informationen entfernt wird, die interne Integrität dieser Daten jedoch erhalten bleibt. Die so bereinigten Daten können dann für Testzwecke verwendet werden, ohne eine Sicherheitslücke oder den Missbrauch personenbezogener Daten zu riskieren. Dies

ist besonders wichtig, wenn große Mengen realistischer Daten benötigt werden und wenn IT-Sicherheits- und Datenschutzrisiken bestehen.
- Generierung synthetischer Testdaten aus einer vorgegebenen Menge von Eingabeparametern (z.B. zur Verwendung beim Zufallstest). Manche dieser Werkzeuge sind in der Lage, Datenbankstrukturen zu analysieren, um zu bestimmen, welche Eingaben vom Test Analysten benötigt werden.

6.3.3 Automatisierte Testausführungswerkzeuge

Testausführungswerkzeuge werden von Test Analysten auf allen Teststufen eingesetzt, um automatisierte Tests durchzuführen und das Ergebnis der Tests zu überprüfen. Testausführungswerkzeuge dienen normalerweise einem oder mehreren der folgenden Zwecke:
- Kosten zu reduzieren (in Bezug auf Aufwand und/oder Zeit)
- mehr Tests auszuführen
- dieselben Tests in unterschiedlichen Umgebungen auszuführen
- Testausführung wiederholbar zu machen
- Tests auszuführen, die manuell nicht ausgeführt werden können (z.B. umfangreiche Prüfungen der Datenvalidierung)

Diese Zwecke überschneiden sich und lassen sich im Hauptzweck zusammenfassen: Überdeckung zu steigern bei gleichzeitiger Reduzierung der Kosten.

Die Rentabilität von Testausführungswerkzeugen ist am höchsten, wenn diese zur Automatisierung von Regressionstests eingesetzt werden; Gründe dafür sind der geringe Wartungsaufwand, der zu erwarten ist, sowie die wiederholte Ausführung der Tests.

Auch die Automatisierung von Smoke-Tests kann eine effektive Nutzungsmöglichkeit sein. Die Gründe liegen darin, dass diese Tests häufig eingesetzt werden, dass das Ergebnis schnell benötigt wird, und dass sie eine automatisierte Methode für die Bewertung neuer Softwareversionen in einer kontinuierlichen Integrationsumgebung bieten, auch wenn dies möglicherweise zu höheren Wartungskosten führen kann.

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 ihr zielgerichteter Einsatz können zu einer verbesserten Rentabilität beitragen.

7. Referenzen

7.1 Standards

[ISO25010] ISO/IEC 25010 (2014) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models, Kapitel 4

[ISO29119-4] ISO/IEC/IEEE 29119-4 Software and Systems Engineering – Software Testing – Part 4, Test Techniques, 2015

[RTCA DO-178C/ED-12C] Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12C, 2013.

7.2 Dokumente von ISTQB und IREB

[IREB_CPRE] IREB Certified Professional for Requirements Engineering Foundation Level Syllabus, Version 2.2.2, 2017

[ISTQB_AL_OVIEW] ISTQB Advanced Level Overview, Version 2.0

[ISTQB_ALTAE_SYL] ISTQB Advanced Level Testautomatisierungsentwickler Syllabus, Version 2019 deutschsprachige Ausgabe

[ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 2019

[ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2018 V3.1D

[ISTQB_FL_UT] ISTQB Foundation Level Specialist Syllabus Usability Testing, Version 2017, deutschsprachige Ausgabe

[ISTQB_GL] ISTQB/GTB Standardglossar der Testbegriffe, Version 3.3, abrufbar von www.gtb.de

7.3 Fachliteratur

[Bath14] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook (2nd Edition)”, Rocky Nook, 2014, 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: Becoming an effective and efficient test professional”, John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Black09] Rex Black, “Advanced Software Testing, Volume 1”, Rocky Nook, 2009, ISBN 978-1-933-952-19-2

[Buwalda02] Hans Buwalda, “Integrated Test Design and Automation: Using the Test Frame Method”, Addison-Wesley Longman, 2002, ISBN 0-201-73725-6

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

[Copeland04] Lee Copeland, “A Practitioner's Guide to Software Test Design”, Artech House, 2004, ISBN 1-58053-791-X

[Craig02] Rick David Craig, Stefan P. Jaskiel, “Systematic Software Testing”, Artech House, 2002, ISBN 1-580-53508-9

[Gilb93] Tom Gilb, Graham Dorothy, “Software Inspection”, Addison-Wesley, 1993, ISBN 0-201-63181-4

[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

[Kuhn16] D. Richard Kuhn et al, “Introduction to Combinatorial Testing, CRC Press, 2016, ISBN 978-0-429-18515-1

[Myers11] Glenford J. Myers, “The Art of Software Testing 3rd Edition”, John Wiley & Sons, 2011, ISBN: 978-1-118-03196-4

[Offutt16] Jeff Offutt, Paul Ammann, Introduction to Software Testing – 2nd Edition, Cambridge University Press, 2016, ISBN 13: 9781107172012,

[vanVeenendaal12] Erik van Veenendaal, “Practical risk-based testing." Product Risk Management: The PRISMA Method ”, 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: tips, tricks, tours, and techniques to guide test design”, Addison-Wesley, 2009, ISBN 0-321-63641-4

7.4 Sonstige Referenzen

Die folgenden Referenzen verweisen auf Informationen im Internet. Diese Referenzen wurden zum Zeitpunkt der Veröffentlichung dieses Advanced Level Lehrplans überprüft. 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
  - Exploring Exploratory Testing, Cem Kaner and Andy Tinkham, www.kaner.com/pdfs/ExploringExploratoryTesting.pdf
  - Pettichord, Bret, “An Exploratory Testing Workshop Report”, www.testingcraft.com/exploratorypettichord
- Kapitel 5
  http://www.tmap.net/checklists-and-templates

8. Anhang A

Die folgende Tabelle ist aus der vollständigen Tabelle der ISO 25010 abgeleitet. Sie konzentriert sich ausschließlich auf die Qualitätsmerkmale, die im Test Analyst-Lehrplan behandelt werden, und vergleicht die in ISO 9126 verwendeten Begriffe (wie in Lehrplan Version 2012 verwendet) mit den Begriffen der neueren ISO 25010 (wie im vorliegenden Lehrplan verwendet).

ISO/IEC 25010	ISO/IEC 9126-1	Bemerkungen
Funktionale Angemessenheit	**Funktionalität**	
Funktionale Vollständigkeit		
Funktionale Korrektheit	Genauigkeit	
Funktionale Angemessenheit (functional appropriateness)	Angemessenheit (suitability)	
	Interoperabilität	Jetzt unter Kompatibilität hinzugefügt
Gebrauchstauglichkeit		
Erkennbare Angemessenheit	Verständlichkeit	Neue Bezeichnung
Erlernbarkeit	Erlernbarkeit	
Operabilität	Operabilität	
Benutzerfehlerschutz		Neues Untermerkmal
Ästhetik der Benutzungsschnitt-stelle	Attraktivität	Neue Bezeichnung
Zugänglichkeit		Neues Untermerkmal
Kompatibilität		Neue Definition
Interoperabilität		
Koexistenz		Siehe Technical Test Analyst-Lehrplan

9. Index