

Certified Tester Advanced Level Test Analyst (CTAL-TA) Lehrplan

v3.1.2

International Software Testing Qualifications Board

Deutschsprachige Ausgabe
Herausgegeben durch German Testing Board e.V.



Urheberrechtsvermerk

Auszüge dieses Dokuments dürfen für den nicht-kommerziellen Gebrauch kopiert werden, wenn die Quelle angegeben ist.

Urheberrecht-Hinweis © International Software Testing Qualifications Board (nachstehend ISTQB® genannt).

ISTQB® ist ein eingetragenes Warenzeichen des International Software Testing Qualifications Board.

Copyright © 2021 die Autoren des Updates v3.1.0: Wim Decoutere, István Forgács, Matthias Hamburg, Adam Roman, Jan Sabak, Marc-Florian Wendland.

Copyright © 2019 die Autoren des Updates 2019: Graham Bath, Judy McKay, Jan Sabak, Erik van Veenendaal

Copyright © 2012 die Autoren: Judy McKay, Mike Smith, Erik van Veenendaal
Für den Lehrplan 2012: Judy McKay, Mike Smith, Erik van Veenendaal

Alle Rechte vorbehalten.

Die Autoren übertragen hiermit das Urheberrecht an das International Software Testing Qualifications Board (ISTQB®). Die Autoren (als derzeitige Urheberrechtsinhaber) und das ISTQB® (als zukünftiger Urheberrechtsinhaber) haben den folgenden Nutzungsbedingungen zugestimmt:

Jedes akkreditierte Schulungsunternehmen darf diesen Lehrplan als Grundlage für einen Schulungskurs verwenden, wenn die Autoren und das ISTQB® als Quelle und Urheberrechtsinhaber des Lehrplans angegeben werden und vorausgesetzt, dass jede Werbung für einen solchen Schulungskurs den Lehrplan erst dann erwähnen darf, wenn die offizielle Akkreditierung der Schulungsunterlagen durch ein vom ISTQB® anerkanntes Mitglieds Board erfolgt ist.

Jede Einzelperson oder Gruppe von Einzelpersonen darf diesen Lehrplan als Grundlage für Artikel und Bücher verwenden, wenn die Autoren und das ISTQB® als Quelle und Urheberrechtsinhaber des Lehrplans genannt werden. Jede andere Verwendung dieses Lehrplans ist ohne vorherige Zustimmung des ISTQB® untersagt.

Jedes vom ISTQB® anerkannte Mitglieds-Board kann diesen Lehrplan übersetzen und den Lehrplan (oder seine Übersetzung) an andere Parteien lizenzieren.

Inhaltsverzeichnis

| | |
|---|----|
| Änderungshistorie..... | 3 |
| Inhaltsverzeichnis | 4 |
| Danksagungen | 6 |
| 0. Einführung in den Lehrplan | 8 |
| 0.1 Zweck dieses Dokuments | 8 |
| 0.2 Der Certified Tester Advanced Level Test Analyst | 8 |
| 0.3 Prüfungsrelevante Lernziele und kognitive Stufen | 8 |
| 0.4 Die Prüfung | 9 |
| 0.5 Voraussetzung für die Prüfung | 9 |
| 0.6 Erwartete Erfahrung | 9 |
| 0.7 Akkreditierung von Trainingskursen..... | 9 |
| 0.8 Detaillierungsgrad des Lehrplans | 9 |
| 0.9 Aufbau des Lehrplans | 10 |
| 1. Die Aufgaben des Test Analysten im Testprozess - 150 min..... | 11 |
| 1.1 Einführung..... | 12 |
| 1.2 Testen im Softwareentwicklungslebenszyklus..... | 12 |
| 1.3 Testanalyse..... | 14 |
| 1.4 Testentwurf..... | 15 |
| 1.4.1 Konkrete und abstrakte Testfälle..... | 16 |
| 1.4.2 Testfälle entwerfen | 17 |
| 1.5 Testrealisierung..... | 18 |
| 1.6 Testdurchführung..... | 20 |
| 2. Die Aufgaben des Test Analysten im risikoorientierten Test - 60 min..... | 22 |
| 2.1 Einführung..... | 23 |
| 2.2 Risikoidentifizierung | 23 |
| 2.3 Risikobewertung..... | 24 |
| 2.4 Risikominderung | 24 |
| 2.4.1 Tests priorisieren | 25 |
| 2.4.2 Anpassung des Testens an weitere Testzyklen | 25 |
| 3. Testverfahren - 630 min..... | 27 |
| 3.1 Einführung..... | 28 |
| 3.2 Black-Box-Testverfahren..... | 28 |
| 3.2.1 Äquivalenzklassenbildung | 29 |
| 3.2.2 Grenzwertanalyse | 30 |
| 3.2.3 Entscheidungstabellentest..... | 31 |
| 3.2.4 Zustandsübergangstest | 33 |
| 3.2.5 Klassifikationsbaumverfahren | 35 |
| 3.2.6 Paarweises Testen | 36 |
| 3.2.7 Anwendungsfallbasierter Test | 38 |
| 3.2.8 Testverfahren kombinieren | 39 |
| 3.3 Erfahrungsbasierte Verfahren | 39 |
| 3.3.1 Intuitive Testfallermittlung | 40 |
| 3.3.2 Checklistenbasiertes Testen..... | 41 |
| 3.3.3 Exploratives Testen | 42 |
| 3.3.4 Fehlerbasiertes Testentwurfsverfahren | 43 |
| 3.4 Anwendung der bestgeeigneten Testverfahren | 44 |
| 4. Das Testen von Softwarequalitätsmerkmalen - 180 min..... | 45 |
| 4.1 Einführung..... | 46 |
| 4.2 Qualitätsmerkmale bei fachlichen Tests | 47 |

| | |
|---|----|
| 4.2.1 Testen der funktionalen Korrektheit..... | 47 |
| 4.2.2 Testen der funktionalen Angemessenheit | 47 |
| 4.2.3 Testen der funktionalen Vollständigkeit..... | 48 |
| 4.2.4 Interoperabilitätstest | 48 |
| 4.2.5 Benutzerzentrierte Evaluierung | 49 |
| 4.2.6 Übertragbarkeitstest | 51 |
| 5. Reviews - 120 min | 53 |
| 5.1 Einführung | 54 |
| 5.2 Checklisten in Reviews verwenden | 54 |
| 5.2.1 Reviews von Anforderungen..... | 54 |
| 5.2.2 Reviews von User Stories..... | 55 |
| 5.2.3 Checklisten anpassen..... | 56 |
| 6. Testwerkzeuge und Testautomatisierung - 90 min..... | 57 |
| 6.1 Einführung | 58 |
| 6.2 Schlüsselwortgetriebenes Testen | 58 |
| 6.3 Arten von Testwerkzeugen | 59 |
| 6.3.1 Testentwurfswerkzeuge..... | 59 |
| 6.3.2 Testdateneditoren und -generatoren | 59 |
| 6.3.3 Automatisierte Testausführungswerkzeuge | 60 |
| 7. Referenzen | 61 |
| 7.1 Standards | 61 |
| 7.2 Dokumente von ISTQB und IREB..... | 61 |
| 7.3 Fachliteratur | 61 |
| 7.4 Sonstige Referenzen..... | 63 |
| 8. Anhang A | 64 |
| 9. Index | 65 |

Danksagungen

Dieses Dokument wurde von der Task Force Testanalyst des International Software Testing Qualifications Board Advanced Level Working Group erstellt: Mette Bruhn-Pedersen (Vorsitz der Arbeitsgruppe); Matthias Hamburg (Product Owner); Wim Decoutere, István Forgács, Adam Roman, Jan Sabak, Marc-Florian Wendland (Autoren).

Die Task Force bedankt sich bei Paul Weymouth und Richard Green für die technische Überarbeitung, bei Gary Mogyorodi für die Konformitätsprüfung mit dem Glossar und bei den Mitglieds-Boards für ihre Reviewanmerkungen und Änderungsvorschläge in Bezug auf die veröffentlichte Version 2019 des Lehrplans .

Die folgenden Personen haben am Review und der Kommentierung dieses Lehrplans teilgenommen: Gery Ágneecz, Armin Born, Chenyifan, Klaudia Dussa-Zieger, Chen Geng (Kevin), Istvan Gercsák, Richard Green, Ole Chr. Hansen, Zsolt Hargitai, Andreas Hetz, Tobias Horn, Joan Killeen, Attila Kovacs, Rik Marselis, Marton Matyas, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Pe'er, Palma Polyak, Nishan Portoyan, Meile Posthuma, Stuart Reid, Murian Song, Péter Sótér, Lucjan Stapp, Benjamin Timmermans, Chris van Bael, Stephanie van Dijck, Paul Weymouth.

Dieses Dokument wurde vom ISTQB® am 23. Feb 2021 veröffentlicht.

Die Version 2019 dieses Dokument wurde von einem Kernteam der Advanced Level-Arbeitsgruppe des International Software Testing Qualifications Board (ISTQB) erstellt: Graham Bath, Judy McKay, Mike Smith.

Folgende Personen haben an Review, Kommentierung und der Abstimmung der Version 2019 dieses Lehrplans mitgearbeitet:

Laura Albert, Markus Beck, Henriett Braunné Bokor, Francisca Cano Ortiz, 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, Attila Kovács, József Kreisz, Dietrich Leimsner, Ren Liang, Claire Lohr, Ramit Manohar Kaul, Rik Marselis, Marton Matyas, Don Mills, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Peer, Pálma Polyák, Meile Posthuma, Lloyd Roden, Adam Roman, Abhishek Sharma, Péter Sótér, Lucjan Stapp, Andrea Szabó, Jan te Kock, Benjamin Timmermans, Chris Van Bael, Erik van Veenendaal, Jan Versmissen, Carsten Weise, Robert Werkhoven, Paul Weymouth.

Die Version 2012 dieses Dokuments wurde von einem Kernteam der International Software Testing Qualifications Board Advanced Level Sub Working Group - Advanced Test Analyst erstellt: Judy McKay (Vorsitz), Mike Smith, Erik van Veenendaal.

Zum Zeitpunkt der Fertigstellung des Advanced Level Syllabus bestand die Advanced Level Working Group aus den folgenden Mitgliedern (in alphabetischer Reihenfolge):

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

Die folgenden Personen haben am Review, der Kommentierung und Abstimmung der Version 2012 des Lehrplans teilgenommen:

Graham Bath, Arne Becher, Rex Black, Piet de Roo, Frans Dijkman, Mats Grindal, Kobi Halperin, Bernard Homès, Maria Jönsson, Junfei Ma, Eli Margolin, Rik Marselis, Don Mills, Gary Mogyorodi,

Stefan Mohacsi, Reto Mueller, Thomas Mueller, Ingvar Nordstrom, Tal Pe'er, Raluca Madalina Popescu, Stuart Reid, Jan Sabak, Hans Schaefer, Marco Sogliani, Yaron Tsubery, Hans Weiberg, Paul Weymouth, Chris van Bael, Jurian van der Laar, Stephanie van Dijk, Erik van Veenendaal, Wenqiang Zheng, Debi Zylbermann.

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 Rückverfolgbarkeit 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 Advanced Level Test Analyst (CTAL-TA).

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

- 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 Advanced Level 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 Standards, Fachbücher und andere ISTQB®-Lehrpläne als Referenzen genannt; deren Inhalt ist jedoch nicht über das hinaus prüfungsrelevant, was im vorliegenden Lehrplan in zusammengefasster Form enthalten ist

Das Format der Prüfung ist Multiple Choice. Es sind 40 Fragen zu beantworten. Zum Bestehen der Prüfung müssen mindestens 65% der Gesamtpunktzahl erreicht 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 Seminar stellt keine Voraussetzung für das Ablegen der Prüfung dar.

0.5 Voraussetzung für die Prüfung

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

0.6 Erwartete Erfahrung

Keines der Lernziele für den Advanced Level Test Analyst geht davon aus, dass spezifische Erfahrungen vorhanden sind.

0.7 Akkreditierung von Trainingskursen

Nationale ISTQB®-Mitgliedsboards können Trainingsprovider akkreditieren, deren Kursmaterial diesem Lehrplan folgt. 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 es darf im Rahmen des Kurses eine ISTQB®-Prüfung abgelegt werden.

0.8 Detaillierungsgrad des Lehrplans

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

- Allgemeine Lernziele, die die Absicht des Advanced Level Test Analyst-Lehrplans beschreiben.
- Eine Liste von Punkten, die 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-Trainingskursen abzudecken ist. Der Lehrplan konzentriert sich auf Materialien, die für alle Softwareprojekte gelten können, einschließlich agiler Softwareentwicklung. Der Lehrplan enthält keine spezifischen Lernziele in Bezug auf einen bestimmten Softwareentwicklungslebenszyklus, aber es wird behandelt, wie diese Konzepte in agiler Softwareentwicklung, anderen Arten von iterativen und inkrementellen Lebenszyklen und in sequentiellen Lebenszyklen angewendet werden.

0.9 Aufbau des Lehrplans

Der Lehrplan besteht aus sechs Kapiteln mit prüfungsrelevanten Inhalten. Die Kapitelüberschrift spezifiziert die Mindestzeit für den Unterricht und die Übungen des Kapitels; eine weitere Aufgliederung der Zeitangabe ist nicht vorgesehen. Für akkreditierte Trainingskurse erfordert der Lehrplan eine Gesamtunterrichtszeit 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 im 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 verschiedenen Lebenszyklusmodellen 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 sind

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 Testrealisierungsaktivitäten 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, die für den Test Analyst von besonderer Bedeutung sind, vertieft. Dies ermöglicht eine weitere Verfeinerung des Testprozesses, damit dieser besser zu den verschiedenen Softwareentwicklungslebenszyklus-Modellen passt.

Schwerpunktmäßig befasst sich der Test Analyst mit der Festlegung der geeigneten Tests und Testfälle, sowie mit deren Entwurf, Realisierung und Durchführung. Auch wenn es wichtig ist, die anderen Tätigkeiten im Testprozess zu verstehen, so erfolgt der überwiegende Teil der Aufgaben von Test Analysten bei den folgenden Aktivitäten:

- Testanalyse
- Testentwurf
- Testrealisierung
- Testdurchführung

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

1.2 Testen im Softwareentwicklungslebenszyklus

Bei der Definition einer Teststrategie sollte der gesamte Softwareentwicklungslebenszyklus berücksichtigt werden. Je nach gewähltem Softwareentwicklungslebenszyklus unterscheidet sich der Zeitpunkt, zu 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 – Rückmeldungen der Anforderungsreviews
- Projektmanagement – Eingaben für den Zeitplan
- Konfigurations- und Änderungsmanagement – Ergebnisse der Tests zur Verifizierung von Softwareversionen, Informationen zur 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 Dokumentierung von Lösungsalternativen und bekannten Problemen
- Erstellen technischer Dokumentation (z.B. Datenbankdesignspezifikationen) – Eingaben für diese Dokumente sowie technisches Review dieser Dokumente

Die Testaktivitäten müssen an den gewählten Softwareentwicklungslebenszyklus angepasst werden, das sequenziell, iterativ, inkrementell oder eine Mischform aus diesen sein kann. Beim sequenziellen V-Modell lässt sich beispielsweise der fundamentale Testprozess des ISTQB® auf Stufe des Systemtests folgendermaßen einbinden:

- Der Systemtest wird gleichzeitig mit dem Projekt geplant, und Testüberwachung und -steuerung dauern bis zum Abschluss der Testaktivitäten an. Dies beeinflusst die Eingaben für die Planung, die vom Test Analyst für das Projektmanagement zur Verfügung gestellt werden.
- Systemtestanalyse und -entwurf richten sich nach Dokumenten wie die Anforderungsspezifikation, System- und (logischen) Architekturfentwurf, sowie nach dem (technischen) Komponentenentwurf.
- Die Implementierung der Systemtestumgebung kann während des Systementwurfs beginnen, obwohl der größte Aufwand normalerweise gleichzeitig mit der Realisierung und dem Komponententest anfällt. Die Arbeiten der Testrealisierung des Systemtests dauern dagegen oft bis wenige Tage vor Beginn der Testdurchführung an.
- Die Systemtestdurchführung beginnt, wenn alle Eingangskriterien erfüllt oder gegebenenfalls aufgehoben sind, was normalerweise bedeutet, dass mindestens der Komponententest und oft auch der Komponentenintegrationstest die Endkriterien erreicht haben. Die Testdurchführung des Systemtests dauert, bis die Endkriterien des Systemtest erfüllt sind.
- Die Testabschlussaktivitäten des Systemtests erfolgen, wenn die Endkriterien erfüllt sind.

Bei iterativen und inkrementellen Modellen werden die Aktivitäten möglicherweise nicht in der gleichen Reihenfolge ausgeführt, oder es werden einzelne Aufgaben ganz weggelassen. Bei einem iterativen Modell wird beispielsweise für jede Iteration eine geringere Menge von Testaktivitäten verwendet. Hierbei erfolgen Testanalyse, Testentwurf, Testrealisierung und -durchführung in jeder Iteration, während die Planung zu Beginn und die Abschlussaktivitäten zum Ende des Projektes erfolgen.

In agilen Softwareentwicklungen 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 agiler Softwareentwicklung wird von Anfang an getestet. Dies beginnt mit der Initiierung der Produktentwicklung, wenn die Entwickler einen ersten Architektur- und Systementwurf erstellen. Reviews müssen nicht formal sein, 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 Analyst vom Team übernommen werden.

Iterative und inkrementelle Modelle reichen von der agilen Softwareentwicklung, bei der Änderungen aufgrund der sich entwickelnden Kundenanforderungen erwartet werden, bis hin zu hybriden Modellen, z.B. der iterativen/inkrementellen Softwareentwicklung in Kombination mit einem V-Modell-Ansatz. In solchen hybriden Modellen sollten Test Analysten an den Planungs- und Entwurfsaktivitäten beteiligt sein und dann während der iterativen/inkrementellen Aktivitäten in eine interaktivere Rolle wechseln.

Unabhängig vom eingesetzten Softwareentwicklungslebenszyklus muss der Test Analyst verstehen, welche Erwartungen bezüglich seiner Mitwirkung bestehen und wann diese erfolgen sollte. Test Analysten leisten einen effektiven Beitrag zur Softwarequalität, indem sie ihre Aktivitäten und ihren Zeitpunkt der Beteiligung an den spezifischen Softwareentwicklungslebenszyklus anpassen, anstatt an einem vordefinierten Rollenmodell festzuhalten

1.3 Testanalyse

Bei der Testplanung wird der Umfang des Testprojekts definiert. Bei der Testanalyse nutzt der Test Analyst diese Definition 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
- Ausführung 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 gibt dokumentierte Informationen (z.B. Anforderungen, User-Stories), in denen das Testobjekt beschrieben ist, und die als Testbasis dienen können (siehe [ISTQB_FL_SYL] Abschnitte 1.4.2 und 2.2 oder eine Liste anderer möglicher Quellen der Testbasis).
- Die Testbasis hat das Review mit akzeptablen Ergebnissen bestanden, und wurde nach dem Review bei Bedarf aktualisiert. Es ist zu beachten, dass für die Definition von abstrakten Testfällen (siehe Abschnitt 1.4.1) die Testbasis möglicherweise noch nicht vollständig definiert sein muss. Bei agiler Softwareentwicklung wird dieser Review-Zyklus 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 falls es keine Dokumentation gibt, können die Testbedingungen in Gesprächen mit den relevanten Stakeholdern identifiziert werden (z.B. in Workshops oder bei der Planung der Sprints). Bei agiler Softwareentwicklung 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 zu testende Element bzw. Objekt sind, sollte der Test Analyst grundsätzlich folgendes beachten:

- In der Regel empfiehlt es sich, die Testbedingungen mit unterschiedlichem Detaillierungsgrad zu definieren. Zunächst werden die abstrakten Bedingungen identifiziert, die die allgemeinen Ziele des Testens definieren, wie z.B. "Funktionalität des Bildschirms X". Danach werden detailliertere Bedingungen als Basis für spezifische Testfälle identifiziert, wie z.B. "Bildschirm X weist eine Kontonummer zurück, die ein Zeichen zu wenig hat". Wenn die Testbedingungen nach dieser hierarchischen Methode definiert werden, kann dies sicherstellen, dass die Überdeckung 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, die für jedes der Produktrisiken notwendig sind, identifiziert werden, 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 die Aktivitäten 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 genaueren 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 sind und dass die Tests auf die Ziele des Projekts abgestimmt sind.

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

- Bestimmen, in welchen Bereichen konkrete und in welchen Bereichen abstrakte Testfälle am besten geeignet sind
- 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 Testentwurf 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 ungeskriptetes Testen verwendet werden können.
- Die bestanden/nicht bestanden-Kriterien für die Tests müssen eindeutig festgelegt werden.
- Tests sollten so entworfen werden, dass sie für andere Tester verständlich sind, und nicht nur für den Autor. Falls der Autor nicht die Person ist, die den Test durchführt, 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 die Tests beteiligt sind, und für die Auditoren, die die Tests eventuell genehmigen müssen.
- Tests sollten sämtliche alle Typen von Interaktionen mit dem Testobjekt überdecken und nicht nur die Interaktionen, die über die Benutzungsschnittstelle für den Benutzer sichtbar sind. Hierzu gehören beispielsweise auch Interaktionen mit anderen Systemen und technische oder physikalische Ereignisse. (siehe [IREB_CPRES] 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 jeweilige Situation beste Abstraktionsniveau für die Testfälle 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 Ausführung des Testfalls und zur Verifizierung der Istergebnisse benötigt.
- Tests sind ausgezeichnet reproduzierbar, d.h. andere Tester erzielen die gleichen Testergebnisse.
- 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 Implementierung 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.
- Die Verfolgbarkeit der konkreten Testfälle 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 Analyst jedoch frei, die tatsächlichen Daten und sogar den Ablauf für die Durchführung des Tests zu variieren.
- Sie können eine bessere Risikoabdeckung als konkrete Testfälle liefern, weil bei jeder Ausführung ein wenig variiert wird.
- Abstrakte Testfälle können schon früh im Anforderungsprozess spezifiziert werden.
- Sie nutzen die Erfahrung des Test Analysten, der die Tests ausführt, sowohl mit dem Testen als auch mit dem Testobjekt.
- Abstrakte Testfälle können eingesetzt 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 die detaillierte Beschreibung fehlt, die bei konkreten Testfällen vorhanden ist.
- Für die Testausführung abstrakter Testfälle kann erfahreneres Testpersonal erforderlich sein.
- Bei der Automatisierung auf der Grundlage von abstrakten Testfällen können die fehlenden Details dazu führen, dass die falschen Istergebnisse validiert werden, bzw. dass Elemente nicht validiert werden, die 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.

Zum Entwurf von Testfällen gehört die Identifizierung von:

- Zielsetzung (d.h. das beobachtbare, messbare Ziel der Testausführung)
- Vorbedingungen, wie z.B. projektbezogene oder lokale Testumgebungen (Testvorrichtungen) und deren geplante Bereitstellung, 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 ausgeführt werden kann)
- erwartete Ergebnisse mit expliziten bestanden/nicht bestanden-Kriterien
- Nachbedingungen, wie z.B. beeinflusste Daten, Zustand des Systems nach Testausführung, Auslöser für nachfolgende Prozessabläufe usw.

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

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

Bei der Entwicklung der Testbedingungen und Testfälle wird 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 Vorschriften, die eingehalten werden müssen
- der Softwareentwicklungslebenszyklus, das zum Einsatz kommt (bei agilen Methoden soll der Umfang der Dokumentation gerade ausreichend sein)
- die Anforderung 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] bietet dafür 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 eine Form des statischen Testens, da dabei Probleme in den Dokumenten der Testbasis gefunden werden können. Die Durchführung von Testanalyse und Testentwurf basierend auf einer Anforderungsspezifikation ist beispielsweise eine ausgezeichnete Vorbereitung auf die Reviewsitzung der Anforderungsspezifikation. Das Lesen der Anforderungen, um diese für den Entwurf von Tests zu verwenden, setzt voraus, dass die Anforderung verstanden wird und dass bewertet werden kann, ob die Anforderung erfüllt ist. Häufig werden bei dieser Aktivität fehlende Anforderungen entdeckt, Anforderungen die unklar sind, oder die nicht testbar sind, oder für die keine Abnahmekriterien definiert sind. Auch Arbeitsergebnisse wie Testfälle, Risikoanalysen und Testkonzepte können Reviews unterzogen werden.

Wenn das Testen eine bestimmte Testinfrastruktur benötigt, die jedoch nicht ohne weiteres verfügbar ist, sollte der Test Analyst detaillierte Anforderungen an die Testinfrastruktur während des Testentwurfs spezifizieren. Sollte diesen Anforderungen nicht rechtzeitig komplettiert werden, besteht die Gefahr, dass sich die Testrealisierung aufgrund unerwarteter Zeit- und Arbeitsaufwände verzögert.. Hinweis: Zur Testinfrastruktur gehört mehr als nur Testobjekte und Testmittel; diese Anforderungen betreffen beispielsweise Räumlichkeiten, Ausrüstungen, Personal, Software, Werkzeuge, Peripheriegeräte, Kommunikationseinrichtungen, Zugriffsberechtigungen und alles was sonst zum Durchführen des Tests nötig ist.

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

1.5 Testrealisierung

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

- Entwicklung der Testabläufe und möglicherweise das Erstellen von automatisierten Testskripten
- Organisieren von Testabläufen und automatisierten Testskripten (falls vorhanden) in Testsuiten, die in einem bestimmten Testlauf ausgeführt werden
- Beratung des Test Managers bei der Priorisierung der auszuführenden Testfälle und Testsuiten
- Erstellung eines Testausführungsplans einschließlich Ressourcenzuweisung, damit mit der Testdurchführung begonnen werden kann (siehe [ISTQB_FL_SYL] Abschnitt 5.2.4)
- Abschluss der Vorbereitung von Testdaten und Testumgebungen
- Aktualisierung der Verfolgbarkeit zwischen der Testbasis und den Testmitteln wie Testbedingungen, Testfälle, Testabläufe, Testskripte und Testsuiten.

Während der Testrealisierung identifizieren die Test Analysten eine effiziente Ausführungsreihenfolge der Testfälle, und erstellen Testabläufe. Das Spezifizieren von Testabläufen erfordert die sorgfältige Identifizierung von Einschränkungen und Abhängigkeiten, die die Reihenfolge der Testausführung beeinflussen können. In den Testabläufen sind außerdem alle Vorbedingungen dokumentiert (z.B.

Laden von Testdaten aus einem Datenspeicher) und alle Aktivitäten nach der Testausführung (z.B. Zurücksetzen des Systemzustands).

Test Analysten identifizieren Testabläufe und automatisierte Testskripte, die gruppiert werden können (z. B. wenn sie sich alle auf das Testen eines bestimmten übergeordneten Geschäftsprozesses beziehen), und organisieren sie in Testsuiten. So können zusammengehörige Testfälle gemeinsam ausgeführt werden.

Test Analysten ordnen Testsuiten innerhalb eines Testausführungsplans so an, dass eine effiziente Testausführung gewährleistet ist. Falls eine risikobasierte Teststrategie verwendet wird, wird in erster Linie die Risikostufe die Reihenfolge vorgeben, in der die Testfälle ausgeführt werden. 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 die Testausführung mit der Reihenfolge abgestimmt werden muss, in der die Software für den Test bereitgestellt wird. Insbesondere bei iterativen und inkrementellen Entwicklungsmodellen muss der Test Analyst den Ablauf mit dem Entwicklungsteam koordinieren, um sicherzustellen, dass die Software in einer Reihenfolge zum Testen freigegeben wird, die das Testen ermöglicht.

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

Wie bereits beschrieben, werden für die meisten Tests Testdaten benötigt, und in einigen Fällen kann die Datenmengen sehr umfangreich sein. Während der Testrealisierung erzeugen Test Analysten Eingabe- und Umgebungsdaten, die in Datenbanken und anderen Repositorien abgelegt werden. Diese Daten müssen zweckdienlich sein, um Fehlerzustände aufzudecken. Test Analysten können auch Daten erzeugen, die für daten- und schlüsselwortgetriebene Tests (siehe Abschnitt 6.2) sowie für manuelle Tests verwendet werden können.

Die Testrealisierung befasst sich außerdem mit der Testumgebung bzw. mit den Testumgebungen. In dieser Phase sollte die Umgebung (bzw. die Umgebungen) vollständig vorhanden und verifiziert werden, bevor die Tests ausgeführt werden. Eine zweckdienliche Testumgebung ist unerlässlich, d.h. die Testumgebung sollte so beschaffen sein, dass sich vorhandene Fehlerzustände unter definierten Rahmenbedingungen aufdecken lassen; sie sollte normal funktionieren, wenn keine Fehlerwirkungen auftreten; und schließlich sollte sie bei Bedarf in den höheren Teststufen beispielsweise die Produktiv- oder Anwendungsumgebung angemessen abbilden. Je nach unvorhergesehenen Änderungen, Testergebnissen oder aus sonstigen Erwägungen können Änderungen an der Testumgebung während der Testdurchführung nötig werden. Falls derartige Änderungen während der Testdurchführung durchgeführt werden, muss bewertet werden, wie sich diese Änderungen auf die bereits durchgeführten Tests auswirken.

In der Testrealisierungsphase 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 Testunterstützung 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 gesammelt werden, die zur Bewertung des aktuellen Status bzw. des Fortschrittes gegenüber der Endkriterien und für Berichte über die Testergebnisse verwendet 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 verwendet.

Das Testen ohne Testskripte sollte allerdings nicht zufällig oder ziellos sein, da dadurch sowohl die aufzuwendende Zeit als die erzielte Überdeckung schwierig vorherzusagen sind, und dies zu einer geringen Fehlerfindung führt. Vielmehr sollte in zeitlich begrenzten Sitzungen getestet werden, die jeweils durch eine Test-Charta grundsätzlich in eine bestimmte Richtung gelenkt werden, jedoch die Freiheit bieten von den Vorgaben der Test-Charta abzuweichen, wenn im Laufe der Sitzung potenziell produktivere Möglichkeiten für den Test 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]. Testanalyse, Testentwurf und Testrealisierung finden zwar immer noch statt, allerdings vorwiegend während der Testausführung.

Wenn solche reaktiven Teststrategien verfolgt werden, dann beeinflussen die Ergebnisse eines jeden Tests die Analyse, den Entwurf und die Realisierung der nachfolgenden Tests. Diese Teststrategien sind leichtgewichtig und oft effektiv bei der Fehlerfindung, es gibt allerdings auch einige Nachteile, darunter die folgenden:

- Erfahrung von Test Analysten ist erforderlich
- Die Dauer kann schwer abzuschätzen sein
- Die Testüberdeckung 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
- Vergleich der tatsächlichen Ergebnisse mit den erwarteten Ergebnissen
- Analyse von Anomalien zur Feststellung ihrer wahrscheinlichen Ursachen
- Meldung von Fehlerzuständen aufgrund der beobachteten Fehlerwirkungen
- Protokollierung der Istergebnisse der Testausführung
- Aktualisierung der Verfolgbarkeit zwischen der Testbasis und den Testmitteln zur Berücksichtigung der Testergebnisse
- Ausführen von Regressionstests

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

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

- Erkennen von Häufungen von Fehlerzuständen, die ein Hinweis darauf sein können, dass für einen bestimmten Teil des Testobjekts mehr Tests notwendig sind
- Vorschläge für künftige explorative Testsitzungen auf der Grundlage der Ergebnisse der explorativen Tests machen
- Identifizierung neuer Risiken aus Informationen, die während der Testdurchführung gewonnen wurden

- Vorschläge zur Verbesserung von Arbeitsprodukten der Testrealisierung machen (z.B. Verbesserungen der Testabläufe)

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

Schlüsselbegriffe

Produktisiko, Risikoidentifizierung, Risikominderung, risikobasiertes Testen ,

Lernziele für die Aufgaben des Test Analysten im risikobasierten Testen

2. Die Aufgaben des Test Analysten im 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 trägt der Test Manager die Gesamtverantwortung für das Aufsetzen und Management einer risikobasierten Teststrategie. In der Regel wird der Test Manager jedoch die Unterstützung des Test Analysten anfordern, um sicherzustellen, dass die risikobasierte Testvorgehensweise fachgerecht implementiert wird.

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

- Risikoidentifizierung
- Risikobewertung
- Risikominderung

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

Test Analysten sollten innerhalb des vom Test Manager 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 funktionale Sicherheit, geschäftliche und wirtschaftliche 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.

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

- Durchführung von Experten-Interviews mit Experten und Benutzern des Geschäftsbereichs
- Durchführung unabhängiger Bewertungen
- Anwendung von Risikovorlagen
- Teilnahme an Risiko-Workshops
- Teilnahme an Brainstorming-Sitzungen mit potenziellen und aktuellen Benutzern
- Spezifikation von Checklisten für den Test
- 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, Business Analysten) zusammenarbeiten, um die Bereiche mit Geschäftsrisiken zu bestimmen, die beim Testen berücksichtigt werden sollten. Bei agiler Softwareentwicklung ermöglicht diese enge Beziehung zu den Stakeholdern eine regelmäßige Risikoidentifizierung, z.B. während der Iterationsplanungs-Sitzungen.

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

- Probleme mit der Genauigkeit der Softwarefunktionalität, z.B. falsche Berechnungen
- Probleme mit der Gebrauchstauglichkeit, z.B. zu wenig Tastaturkürzel

- Probleme mit der Übertragbarkeit, z.B. die Unfähigkeit, die Anwendung auf bestimmten Plattformen zu installieren

2.3 Risikobewertung

Während es bei der Risikoidentifizierung darum geht, möglichst viele vorhandene Risiken zu identifizieren, befasst sich die Risikobewertung mit der Untersuchung der identifizierten Risiken. Sie kategorisiert jedes Risiko und bestimmt die Risikostufe.

Zur Bestimmung der Risikostufe wird normalerweise für jedes einzelne Risiko die Eintrittswahrscheinlichkeit des Risikos und das Schadensausmaß des Risikos bewertet. Die Eintrittswahrscheinlichkeit wird normalerweise als die Wahrscheinlichkeit interpretiert, dass das potenzielle Problem in dem System unter Test vorhanden ist bzw. beobachtet wird, wenn das System in der Produktion ist. Technical Test Analysten sollten mitwirken, wenn es darum geht, die potenzielle Eintrittswahrscheinlichkeit für jedes einzelne Risikoelement zu finden und zu verstehen. Test Analysten sollten dazu beitragen, die potenziellen geschäftlichen Auswirkungen eines Problems zu verstehen, falls dieses auftritt (bei agiler Softwareentwicklung kann diese rollenbasierte Unterscheidung weniger stark ausgeprägt sein).

Das Schadensausmaß bei Eintreten eines Risikos wird oft als das Ausmaß der Auswirkung auf die Benutzer, Kunden oder andere Betroffene interpretiert. Es geht also um ein Geschäftsrisiko. Der Test Analyst sollte mitwirken, wenn es darum geht, die potenziellen Auswirkungen auf Geschäft oder Benutzer für die einzelnen Risiken zu identifizieren und zu bewerten. Folgende Faktoren beeinflussen die Geschäftsrisiken:

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

Angesichts der vorhandenen Informationen muss der Test Analyst die Risikostufen für die Geschäftsrisiken basierend auf den vom Test Manager vorgegebenen Richtlinien festlegen. Diese könnten anhand einer Ordinalskala (konkret numerisch oder niedrig/mittel/hoch) oder anhand von Ampelfarben klassifiziert werden. Sobald die Eintrittswahrscheinlichkeit und das Schadensausmaß des Risikos zugewiesen wurden, verwenden die Test Manager diese Werte, um die Risikostufe für jedes Risikoelement zu bestimmen. Diese Risikostufe wird dann zur Priorisierung der Aktivitäten zur Risikominderung verwendet [vanVeenendaal12].

2.4 Risikominderung

Im Projekt sollten die Test Analysten folgende Aufgaben übernehmen:

- Produkt Risiken mindern indem geeignete Testfälle entworfen werden, die eindeutig zeigen, ob die Tests bestanden oder nicht bestanden haben, sowie an Reviews von Software-Artefakten 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 Wahrscheinlichkeit, die Auswirkungen oder beides gegebenenfalls anpassen
- aus den beim Testen gewonnenen Informationen neue Risiken identifizieren

Im Hinblick auf Produktrisiken trägt das 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 Bedingungen (d.h. unter 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. 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 Produkt sind. Anstatt den Benutzer-Abnahmetest abzuwarten, bei dem die Probleme entdeckt würden, erteilt der Test Analyst einem frühen Gebrauchstauglichkeitstest auf Grundlage eines Prototyps eine hohe Priorität. So sollen Probleme mit der Gebrauchstauglichkeit frühzeitig im Testprozess identifiziert und behoben werden. Eine solche Priorisierung sollte so früh wie möglich in der Planung berücksichtigt werden, damit die Zeit für die notwendigen Tests zum benötigten Zeitpunkt eingeplant werden kann.

Manchmal werden alle hohen Risikostufen vor den niedrigeren Risikostufen getestet, und die Tests erfolgen streng nach der Reihenfolge der Risiken (depth-first", d.h. Testen in die Tiefe). In anderen Fällen machen die Tester Stichproben von identifizierten Risikobereichen aller Risikostufen, gewichten die Risiken und wählen Tests aus, wobei sie allerdings dafür sorgen, dass jedes Risiko mindestens einmal überdeckt 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 an weitere Testzyklen

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

- alle neuen oder deutlich veränderten Produktrisiken
- instabile oder fehleranfällige Bereiche des Systems, die im Laufe des Testens identifiziert wurden
- Risiken als Folge behobener Fehler

- Typische Fehler, die beim Testen gefunden wurden
- Bereiche, die nicht ausreichend getestet wurden (geringe Anforderungsüberdeckung)

3. Testverfahren - 630 min

Schlüsselbegriffe

anwendungsfallbasierter Test, Äquivalenzklassenbildung, Black-Box-Testverfahren, checklistenbasiertes Testen, Entscheidungstabelletest, 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 Entscheidungstabelletests Testfälle entwerfen
- TA-3.2.4 (K4) Eines oder mehrere vorgegebene Spezifikationselemente analysieren und unter Verwendung des Zustandsübergangstest 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 Testens 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 Erfahrungs-basierte Testverfahren

- TA-3.3.1 (K2) Die Prinzipien der erfahrungs-basierten 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 spezifizieren
- 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 erfahrungs-basierten 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 Software-Qualitätsmerkmalen wird im nächsten Kapitel behandelt.

Die in den nachfolgenden Abschnitten vorgestellten Testverfahren konzentrieren sich primär auf die Bestimmung optimaler Testdaten (z.B. Äquivalenzklassen) oder auf die Ableitung von Testabläufen (z.B. Zustandsmodelle). Es ist üblich die Verfahren zu kombinieren, um vollständige Testfälle zu entwerfen.

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 den Überdeckungsgrad, die als Maß für die Testentwurfs- und Testdurchführungsaktivitäten verwendet werden können. Wenn diese Kriterien vollständig erfüllt sind, bedeutet das nicht, dass die Menge von Tests vollständig ist. Es bedeutet vielmehr, dass das Modell keine weiteren Tests vorschlägt, um mit dem vorliegenden 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 die Anforderungen nicht dokumentiert sind, sondern implizit vorliegen, beispielsweise wenn die Funktionalität eines Legacy-Systems ersetzt werden soll.

Es gibt eine Reihe von Black-Box-Testverfahren, die für unterschiedliche Arten von Software und unterschiedliche Szenarien eingesetzt werden. In den nachfolgenden Abschnitten werden die Anwendbarkeit der einzelnen Verfahren aufgezeigt, sowie einige Einschränkungen und Schwierigkeiten, auf die der Test Analyst stoßen könnte. Außerdem werden die Methoden zur Messung der Überdeckung beschrieben, sowie die Fehlerarten, die mit den Verfahren aufgedeckt werden.

Weitere Einzelheiten entnehmen Sie bitte [ISO29119-4], [Bath14], [Beizer95], [Black07], [Black09], [Copeland04], [Craig02], [Forgács19], [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 enthalten Wertemengen, die auf dieselbe Weise verarbeitet werden. Durch die Auswahl eines repräsentativen Wertes einer Äquivalenzklasse wird die Überdeckung aller Werte derselben Äquivalenzklasse angenommen.

In der Regel bestimmen mehrere Parameter das Verhalten des Testobjekts. Bei der Kombination der Äquivalenzklassen verschiedener Parameter zu Testfällen können verschiedene Verfahren angewendet werden.

Anwendbarkeit

Das 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 bei der Verwendung durch die Anwendung nicht interagieren. Eine Äquivalenzklasse kann eine beliebige nicht leere Menge von Werten sein, z. B.: geordnet, ungeordnet, diskret, kontinuierlich, unendlich, endlich oder sogar ein Singelton. Die Auswahl der Äquivalenzklassen kann sowohl für gültige als auch für ungültige Daten erfolgen (d.h. Äquivalenzklassen deren Werte für die Software unter Test als ungültig angesehen werden sollten).

Die Äquivalenzklassenbildung ist am nützlichsten, wenn sie mit der Grenzwertanalyse kombiniert wird, welche die getesteten Werte um die Grenzen der einzelnen Äquivalenzklassen erweitert. Die Äquivalenzklassenbildung, bei der Werte aus den gültigen Äquivalenzklassen verwendet werden, ist sehr gebräuchlich für den Smoke-Test eines neuen Builds oder einer neuen Version, 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 ist es möglich, dass mit diesem Verfahren Fehlerzustände übersehen werden. Es ist auch wichtig, dass die Äquivalenzklassen sorgfältig ausgewählt werden. Beispiel: Für ein Eingabefeld für positive und negative Zahlen könnte es besser sein, 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. Für einen Test Analyst ist es wichtig, die zugrunde liegenden Verarbeitungsprozesse zu verstehen, um die bestmöglichen Äquivalenzklassen zu bestimmen. Hierfür kann Unterstützung beim Verstehen des Code-Entwurfs erforderlich sein.

Der Test Analyst sollte auch mögliche Abhängigkeiten zwischen Äquivalenzklassen verschiedener Parameter berücksichtigen. Zum Beispiel darf in einem Flugreservierungssystem der Parameter "begleitender Erwachsener" nur in Kombination mit der Altersklasse "Kind" verwendet werden.

Überdeckung

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

Wenn das Verhalten des Testobjekts von einem einzigen Parameter abhängt, sollte jede Äquivalenzklasse, ob gültig oder ungültig, mindestens einmal überdeckt werden.

Bei mehr als einem Parameter sollte der Test Analyst je nach Risiko eine einfache oder kombinatorische Überdeckungsart wählen [Offutt16]. Die Unterscheidung zwischen Kombinationen, die nur gültige Klassen enthalten, und Kombinationen, die eine oder mehrere ungültige Klassen

enthalten, ist daher unerlässlich. Bei den Kombinationen mit nur gültigen Äquivalenzklassen ist die Mindestanforderung eine einfache Überdeckung aller gültigen Klassen über alle Parameter. Die minimale Anzahl von Testfällen, die in einer solchen Testsuite benötigt wird, entspricht der größten Anzahl von gültigen Klassen eines Parameters, unter der Annahme, dass die Parameter unabhängig voneinander sind. Gründlichere Überdeckungsarten, die sich auf kombinatorische Verfahren beziehen, sind das paarweise Testen (siehe Abschnitt 3.2.6) oder die vollständige Überdeckung aller Kombinationen der gültigen Äquivalenzklassen. Ungültige Äquivalenzklassen sollten mindestens einzeln, d. h. in Kombination mit gültigen Klassen für die anderen Parameter, getestet werden, um Fehlermaskierung zu vermeiden. So trägt jede ungültige Klasse einen Testfall zur Testsuite für einfache Überdeckung bei. Bei hohem Risiko können weitere Kombinationen zur Testsuite hinzugefügt werden, die z. B. nur aus ungültigen Klassen oder aus Paaren von ungültigen Klassen bestehen.

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 geordneter Äquivalenzklassen richtig verarbeitet werden. Zwei unterschiedliche Vorgehensweisen für die Grenzwertanalyse sind gebräuchlich: das Testen mit zwei und das Testen mit drei Grenzwerten.

Beim Testen mit zwei Grenzwerten wird der Grenzwert (direkt auf der Grenze) und der Wert getestet, der so knapp wie möglich außerhalb der Grenze liegt (mit dem kleinstmöglichen Inkrement, basierend auf der erforderlichen Genauigkeit). Beispiel: Für Beträge in einer Währung, die zwei Nachkommastellen hat und die Äquivalenzklasse die Werte 1 bis 10 enthält, würden zum Testen der oberen Grenze die Werte 10 und 10,01 verwendet. Für die untere Grenze würden die Werte 1 und 0,99 verwendet. Die Grenzen sind als der größte und der kleinste Wert der festgelegten Äquivalenzklasse definiert.

Für den Test mit drei Grenzwerten werden die Werte unterhalb, auf und oberhalb der Grenze verwendet. Im geschilderten Beispiel wären das die Werte 9,99, 10 und 10,01 für die obere Grenze, bzw. 0,99, 1 und 1,01 für die untere Grenze. Die Entscheidung, ob mit zwei oder mit drei Grenzwerten getestet werden soll, orientiert sich am identifizierten Risiko des zu testenden Elements, wobei Elemente mit höheren Risiken mit drei Grenzwerten getestet werden sollten.

Anwendbarkeit

Das 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 angewendet. Geordnete Äquivalenzklassen sind erforderlich, da das Konzept vorsieht, dass Werte auf oder neben der Grenze liegen. Ein Zahlenbereich ist beispielsweise eine geordnete Äquivalenzklasse. Eine Klasse, die aus einigen Zeichenketten besteht, kann auch geordnet sein, z.B. nach ihrer lexikografischen Reihenfolge. Wenn aber die Reihenfolge aus geschäftlicher oder technischer Sicht nicht relevant ist, sollten die Grenzwerte nicht im Fokus stehen. Zusätzlich zu Zahlenbereichen kann die Grenzwertanalyse zudem für nachfolgende Äquivalenzklassen angewendet werden:

- Numerische Attribute nicht-numerischer Variablen (z.B. Länge)
- Anzahl der Schleifendurchläufe, einschließlich Schleifen in Zustandsdiagrammen
- Anzahl der Iterationselemente in gespeicherten Datenstrukturen wie z. B. Arrays
- Größe von physikalischen Objekten, z.B. Speicher
- Dauer von Aktivitäten

Einschränkungen/Schwierigkeiten

Da die Genauigkeit der Grenzwertanalyse von der genauen Festlegung der Äquivalenzklassen abhängt, gelten auch dieselben Einschränkungen und Schwierigkeiten wie bei der Äquivalenzklassenbildung. Der Test Analyst muss die Präzision bei den gültigen und ungültigen Werten berücksichtigen, um die zu testenden Werte genauestens zu bestimmen. Die Grenzwertanalyse berücksichtigt nur geordnete Äquivalenzklassen, allerdings beschränkt sich dies nicht ausschließlich auf gültige Wertebereiche. Beispiel: Beim Testen der Anzahl der von einer Tabellenkalkulationsanwendung unterstützten Zellen gibt es eine Äquivalenzklasse, die die Anzahl der Zellen bis einschließlich der maximalen Anzahl enthält (der Grenze), und eine weitere Äquivalenzklasse, die mit einer Zelle über dem Maximum (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.

Ähnlich wie bei den Äquivalenzklassen sollte der Test Analyst im Fall von mehreren Parametern je nach Risiko eine einfache oder kombinatorische Abdeckungsart wählen.

Fehlerarten

Mit der Grenzwertanalyse werden verschobene oder fehlende Grenzen zuverlässig aufgedeckt, und es können zusätzliche Grenzen gefunden werden. Mit diesem Verfahren werden Fehlerzustände in Zusammenhang mit der Verarbeitung von Grenzwerten aufgedeckt, insbesondere bei logischen Bedingungen mit „kleiner-als“ und „größer-als“ (d.h. wenn Grenzverschiebungen vorliegen). Es können mit diesem Verfahren auch nicht-funktionale Fehlerzustände aufgedeckt werden, z.B. das System unterstützt 10.000 gleichzeitige Benutzer, aber nicht 10.001.

3.2.3 Entscheidungstabellentest

Eine Entscheidungstabelle ist eine tabellarische Darstellung einer Menge von Bedingungen und zugehörigen Aktionen, in der Regeln formuliert sind, die angeben, welche Aktion für welche Menge von Bedingungswerten erfolgen soll [OMG-DMN]. Test Analysten können Entscheidungstabellen verwenden, um die Regeln zu analysieren, die für die Software unter Test gelten, und Tests zu entwerfen, die diese Regeln überdecken.

Bedingungen und die daraus resultierenden Aktionen des Testobjekts bilden die Zeilen der Entscheidungstabelle, wobei normalerweise die Bedingungen oben und die Aktionen unten stehen. Die erste Spalte der Tabelle enthält die Beschreibung der Bedingungen bzw. Aktionen. Die folgenden Spalten, die als Regeln bezeichnet werden, enthalten die Werte der Bedingungen bzw. die entsprechenden Werte der Aktionen.

Entscheidungstabellen, in denen Bedingungen boolesch mit einfachen Werten "Wahr" und "Falsch" sind, werden Entscheidungstabellen mit eingeschränkter Eingabe genannt. Ein Beispiel für eine solche Bedingung ist "Einkommen des Benutzers < 1000". Entscheidungstabellen mit erweiterter Eingabe erlauben Bedingungen mit mehreren Werten, die diskrete Elemente oder Gruppen von Elementen darstellen können. Beispielsweise kann eine Bedingung "Einkommen des Benutzers" einen von drei möglichen Werten annehmen: "kleiner als 1000", "größer 1000 und kleiner 2000" und "größer als 2000".

Einfache Aktionen nehmen die booleschen Werte "Wahr" und "Falsch" an (z. B. nimmt die Aktion "Zugelassener Rabatt = 20 %" den mit "X" gekennzeichneten Wert "Wahr" an, wenn die Aktion eintreten soll, und den mit "-" gekennzeichneten Wert "Falsch", wenn die Aktion nicht eintreten soll). Genau wie bei Bedingungen können für Aktionen auch Werte aus anderen Bereichen verwendet werden. Zum Beispiel kann die Aktion "Gewährter Rabatt" einen von fünf möglichen Werten annehmen: 0%, 10%, 20%, 35% und 50%.

Entscheidungstabellentests beginnen mit dem Entwurf von Entscheidungstabellen auf der Grundlage der Spezifikation. Regeln, die undurchführbare Kombinationen von Bedingungswerten enthalten, werden ausgeschlossen oder als "undurchführbar" markiert. Als nächstes sollte der Test Analyst die Entscheidungstabellen mit den anderen Stakeholdern einem Review unterziehen. Der Test Analyst sollte sicherstellen, dass die Regeln innerhalb der Tabelle konsistent (d.h., dass sich die Regeln nicht überschneiden), vollständig (d.h., dass sie für jede mögliche Kombination von Bedingungswerten eine Regel enthalten) und korrekt (d.h., dass sie das beabsichtigte Verhalten modellieren) sind.

Das Grundprinzip beim Entscheidungstabellentest ist, dass die Regeln die Testbedingungen bilden.

Beim Entwurf eines Testfalls, der eine bestimmte Regel überdeckt, sollte der Test Analyst beachten, dass die Eingaben des Testfalls andere Parameter sein können als die Bedingungen der Entscheidungstabelle. Beispielsweise könnte der Wert "WAHR" der Bedingung "Alter > 18?" vom Test Analysten verlangen, das Alter aus weiteren Parametern (bspw. dem Geburtsdatum und dem aktuellen Datum) zu berechnen. In ähnlicher Weise könnten die erwarteten Ergebnisse des Testfalls nur indirekt mit den Aktionen der Entscheidungstabelle zusammenhängen.

Nachdem die Entscheidungstabelle fertig ist, müssen die Regeln als Testfälle implementiert werden, indem Eingabewerte (und erwartete Ergebnisse) ausgewählt werden, die die Bedingungen und Aktionen erfüllen.

Konsolidierte Entscheidungstabellen

Bei dem Versuch, jede mögliche Kombination der Bedingungen zu testen, können Entscheidungstabellen sehr groß werden. Eine vollständige Entscheidungstabelle mit eingeschränkten Eingaben und n Bedingungen hat 2^n Regeln. Ein Verfahren zur systematischen Reduzierung der Anzahl von Kombinationen wird als Testen konsolidierter Entscheidungstabellen bezeichnet [Mosley93]. Bei Anwendung dieses Verfahrens kann eine Gruppe von aktionsgleichen Regeln auf eine Regel reduziert (konsolidiert) werden, wenn innerhalb dieser Gruppe einige Bedingungen für die Aktion irrelevant sind und alle anderen Bedingungen unverändert bleiben. In der resultierenden (konsolidierten) Regel werden die Werte der irrelevanten Bedingungen als "egal" bezeichnet und üblicherweise mit einem Bindestrich "-" gekennzeichnet. Für Bedingungen mit "egal"-Werten kann der Test Analyst beliebige gültige Werte für die Testimplementierung angeben.

Ein weiterer Fall für das Konsolidieren von Regeln ist, wenn ein Bedingungswert in Kombination mit einigen anderen Bedingungswerten nicht anwendbar ist oder wenn zwei oder mehr Bedingungen widersprüchliche Werte haben. Wenn z. B. in einer Entscheidungstabelle für Kartenzahlungen die Bedingung "Karte ist gültig" falsch ist, ist die Bedingung "PIN-Code ist korrekt" nicht anwendbar.

Konsolidierte Entscheidungstabellen können viel weniger Regeln haben als die vollständigen Entscheidungstabellen, was zu einer geringeren Anzahl von Testfällen und weniger Aufwand führt. Wenn eine bestimmte Regel für eine Bedingung den Wert "egal"-enthält und nur ein Testfall diese Regel abdeckt, wird nur einer von mehreren möglichen Werten der Bedingung für diese Regel getestet, so dass ein Fehlerzustand, der bei einem der nicht verwendeten Wert besteht, möglicherweise unentdeckt bleibt. Daher sollte der Test Analyst für hohe Risikostufen in Abstimmung mit dem Test Manager separate Regeln für jede mögliche Kombination der einzelnen Bedingungswerte definieren, anstatt die Entscheidungstabelle zu konsolidieren.

Anwendbarkeit

Entscheidungstabellentests werden üblicherweise in den Teststufen Integrations-, System- und Abnahmetest angewendet. Er könnte auch beim Komponententest angewendet werden, wenn die zu testende Komponente für Entscheidungslogik verantwortlich ist. Dieses Verfahren ist besonders nützlich, wenn das Testobjekt in Form von Ablaufdiagrammen oder Tabellen mit Geschäftsregeln vorliegt.

Entscheidungstabellen können auch zur Spezifikation von Anforderungen verwendet werden, und manchmal liegen Anforderungsspezifikationen bereits in diesem Format vor. Der Test Analyst sollte dennoch an dem Review der Entscheidungstabellen teilnehmen und diese analysieren, bevor er mit dem Testentwurf beginnt.

Einschränkungen/Schwierigkeiten

Beim Aufstellen der Kombinationen von Bedingungen kann die Identifikation aller interagierenden Bedingungen eine Herausforderung darstellen, insbesondere dann, wenn die Anforderungen an das System nicht klar definiert oder überhaupt nicht dokumentiert sind. Bei der Auswahl der in einer Entscheidungstabelle betrachteten Bedingungen muss darauf geachtet werden, dass die Anzahl der Kombinationen dieser Bedingungen überschaubar bleibt. Im schlimmsten Fall wächst die Anzahl der Regeln exponentiell.

Überdeckung

Die übliche Überdeckung für dieses Verfahren ist, dass jede Regel der Entscheidungstabelle mit einem Testfall überdeckt wird. Die Überdeckung wird berechnet, indem die Anzahl der durch die Testsuite überdeckten Regeln durch die Gesamtzahl der durchführbaren Regeln dividiert wird. Das Ergebnis wird in Prozent angegeben.

Grenzwertanalyse und Äquivalenzklassenbildung können mit dem Entscheidungstabellentest kombiniert werden, insbesondere bei Entscheidungstabellen mit erweiterten Ein- und Ausgaben. Wenn Bedingungen Äquivalenzklassen enthalten, die geordnet sind, können die Grenzwerte als zusätzliche Werte verwendet werden, die zu zusätzlichen Regeln und Testfällen führen.

Fehlerarten

Zu den typischen Fehlerzuständen, die mit diesem Verfahren aufgedeckt werden, gehört eine fehlerhafte Verarbeitungslogik, die auf bestimmten Kombinationen von Bedingungen basiert und zu unerwarteten Ergebnissen führt. Beim Erstellen der Entscheidungstabellen können Fehlerzustände in den Spezifikationen aufgedeckt werden. Es ist nicht ungewöhnlich, beim Aufstellen der Bedingungen festzustellen, dass das erwartete Ergebnis für eine oder mehrere Regeln nicht spezifiziert ist. Die häufigsten Fehlerarten sind fehlende Aktionen (d.h., es gibt keine Information darüber, was in einer bestimmten Situation tatsächlich geschehen soll) und Widersprüche.

3.2.4 Zustandsübergangstest

Der Zustandsübergangstest wird eingesetzt, um die Fähigkeit des Testobjekts zu testen, über gültige Übergänge in definierte Zustände einzutreten und diese zu verlassen, sowie um zu versuchen, in ungültige Zustände einzutreten oder ungültige Übergänge abzudecken. Ereignisse sind die Auslöser für das Testobjekt, von einem Zustand in einen anderen zu wechseln und dabei Aktionen auszuführen. Ereignisse werden gegebenenfalls von Bedingungen eingeschränkt (manchmal auch als Wächterbedingung (engl. guard condition) oder Übergangsschutz (engl. transition guard) bezeichnet), die Einfluss darauf haben, welcher Pfad an Zustandsübergängen genommen wird. 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 (letztere kann auch die ungültigen Zustandsübergänge enthalten).

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 des Bildschirms). Der Zustandsübergangstest kann in jeder Teststufe eingesetzt werden. Eingebettete Software, Web-Software und jede Art von Transaktionssoftware 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 hat, werden die einzelnen Bildschirmmasken, die dem Benutzer angezeigt werden, oft durch Zustände dargestellt. Bei eingebetteter Software hängen die Zustände meist von den Zuständen der Hardware ab.

Abgesehen von den Zuständen ist der einzelne Zustandsübergang das grundlegende Überdeckungselement beim Zustandsübergangstest. Wenn einfach nur alle Zustandsübergänge einzeln getestet werden, werden sicherlich einige Fehlerzustände bei den Zustandsübergängen gefunden. Es können jedoch mehr Fehlerzustände gefunden werden, wenn Abfolgen bzw. Sequenzen von Übergängen getestet werden. Ein einzelner Zustandsübergang wird als 0-Switch bezeichnet, eine Sequenz mit zwei aufeinanderfolgenden Zustandsübergängen wird als 1-Switch bezeichnet, eine Sequenz von drei aufeinanderfolgenden Zustandsübergängen wird als 2-Switch bezeichnet, und so weiter. Im Allgemeinen repräsentiert ein N-Switch $N+1$ aufeinanderfolgende Zustandsübergänge [Chow1978]. Mit zunehmendem N wächst die Anzahl der N-Switches sehr schnell, was es schwierig macht, mit einer vernünftigen, kleinen Anzahl von Tests eine Überdeckung der N-Switches zu erreichen.

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 mindestens einmal überdeckt wurde. 100% Zustandsübergangsüberdeckung (auch bekannt als 100% 0-Switch-Überdeckung) garantiert, dass jeder Zustand besucht und jeder Zustandsübergang durchlaufen wurde, es sei denn, der Systementwurf oder das Zustandsübergangsmodell (Zustandsdiagramm oder Zustandsübergangstabelle) ist fehlerhaft. Abhängig von den Beziehungen zwischen den Zuständen und den Zustandsübergängen kann es erforderlich sein, dass manche Zustandsübergänge mehr als einmal durchlaufen werden müssen, damit andere Zustandsübergänge wenigstens einmal ausgeführt werden können.

Der Begriff "N-Switch-Überdeckung" bezieht sich auf Anzahl der überdeckten Switches der Länge $N+1$, als Prozentsatz der Gesamtzahl der Switches dieser Länge. 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 all diejenigen Zustände getestet wurden, bei denen Schleifen beginnen und enden. In einer solchen Schleife darf jeder Zustand (außer dem Anfangs-/Endzustand) nur 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 Anforderungen für die Überdeckung des Zustandsübergangstests muss hervorgehen, ob ungültige Zustandsübergänge enthalten sind.

Der Entwurf 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" auflistet [Black09].

Zur Identifizierung der Überdeckungselemente (z.B. Zustandsübergänge, Zustände oder N-Switches) kann gegebenenfalls auch ein manuelles Verfahren angewandt werden. Eine empfohlene Methode ist, das Zustandsdiagramm und die Zustandsübergangstabelle auszudrucken und mit einem Stift bzw. Bleistift die überdeckten 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 die mit diesem Verfahren aufgedeckt werden, gehören (siehe auch [Beizer95]):

- Inkorrekte Ereignisse bzw. Werte
- Inkorrekte Aktionen bzw. Werte
- Inkorrektter Ausgangszustand
- Unfähigkeit, einen oder mehrere Endzustände zu erreichen
- Unfähigkeit, zu erforderlichen Zuständen zu wechseln
- Zusätzliche (nicht benötigte) Zustände
- Unfähigkeit, einen oder mehrere gültige Zustandsübergänge korrekt auszuführen
- Fähigkeit, ungültige Zustandsübergänge auszuführen
- Falsche Wächterbedingungen

Beim Erstellen des Zustandsübergangsdiagramms können Fehlerzustände in den Spezifikationsdokumenten aufgedeckt werden. Am häufigsten handelt es sich dabei um Lücken (z.B. ist nicht spezifiziert, was in einer bestimmten Situation tatsächlich geschehen soll) oder 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 dabei in Klassifikationen und Klassen organisiert:

- **Klassifikationen:** Diese stellen Parameter innerhalb des Datenraums für das Testobjekt dar wie Eingabeparameter (die zusätzlich Umgebungszustände und Vorbedingungen enthalten können), und Ausgabeparameter. Wenn eine Anwendung beispielsweise auf viele verschiedene Arten konfiguriert werden kann, könnten die Klassifikationen Client, Browser, Sprache und Betriebssystem beinhalten.
- **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 Klassifikation Sprache Äquivalenzklassen für Englisch, Französisch und Spanisch enthalten.

Klassifikationsbäume ermöglichen es den Test Analysten, nach eigenem Ermessen Kombinationen einzugeben. Dazu gehören z.B. paarweise Kombinationen (siehe Abschnitt 3.2.6), 3-fache Kombinationen, sowie einzelne Klassen (1-fach).

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

Anwendbarkeit

Die Erstellung eines Klassifikationsbaums hilft einem Test Analysten, Parameter (Klassifikationen) und deren Äquivalenzklassen (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 der Äquivalenzklassenbildung, der Grenzwertanalyse 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 verwenden. Außerdem werden mit dem Klassifikationsbaumverfahren keine vollständigen Testfälle erstellt, sondern nur Testdatenkombinationen. Test Analysten müssen jeder Testkombination Ergebnisse zufügen, um vollständige Testfälle zu erstellen.

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 Kombinationen oder andere kombinatorische Testverfahren wie z.B. 3-fache Kombinationen überdeckt werden sollen.

Fehlerarten

Die Art der gefundenen Fehlerzustände hängt von dem/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, wenn mehr Kombinationen entstehen, als in der verfügbaren Zeit getestet werden können. Die Eingabeparameter könnten insofern unabhängig voneinander sein, als dass jeder Wert für einen beliebigen Eingabeparameter mit jedem Wert eines beliebigen anderen Parameters kombiniert werden kann. Dies ist jedoch nicht immer der Fall (siehe Hinweis zu Feature-Modellen weiter unten). Die Kombination eines bestimmten Parameters (Variable oder Faktor) mit einem bestimmten Wert dieses Parameters wird als Parameter-Wert-Paar bezeichnet (wenn z.B. 'Farbe' ein Parameter mit sieben zulässigen Werten einschließlich 'rot' ist, dann könnte 'Farbe = rot' ein Parameter-Wert-Paar sein).

Das paarweise Testen verwendet kombinatorische Verfahren, um sicherzustellen, dass jedes Parameter-Wert-Paar einmal gegen jedes Parameter-Wert-Paar jedes anderen Parameters getestet wird (d.h. es werden 'alle Paare' von Parameter-Wert-Paaren für zwei beliebige unterschiedliche Parameter getestet). Dabei wird vermieden, dass alle Kombinationen von Parameter-Wert-Paaren getestet werden. Wenn der Test Analyst einen manuellen Ansatz verwendet, wird eine Tabelle erstellt, in der Testfälle durch Zeilen und eine Spalte für jeden Parameter dargestellt werden. Der Test Analyst füllt dann die Tabelle mit Werten auf, sodass alle Wertepaare in der Tabelle identifiziert werden können (siehe [Black09]). Alle leeren Eingaben in der Tabelle können vom Test Analyst unter Verwendung seines eigenen Domänenwissens mit Werten gefüllt werden.

Es gibt verschiedene Werkzeuge, die den Test Analyst bei dieser Aufgabe unterstützen (siehe www.pairwise.org für Beispiele). Diese Werkzeuge benötigen als Eingabe eine Liste der Parameter und deren Werte und generieren eine geeignete Menge an Kombinationen von Werten, 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. Dies bedeutet nicht, dass sich die 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 Testfälle enthalten 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. Darüber hinaus 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 beherrschbaren und umsetzbaren Umfang zu identifizieren.

Bei Parametern mit vielen Werten kann zunächst Äquivalenzklassenbildung oder ein anderer Auswahlmechanismus 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 wird 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. Eine unerwartete Interaktion zwischen bestimmten Variablen kann 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 diese 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 für ein bestimmtes Überdeckungsmaß manuell zu bestimmen. Zur Bestimmung der minimalen Menge von Kombinationen können Werkzeuge eingesetzt werden. Einige Werkzeuge bieten die Möglichkeit, bestimmte 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% paarweise Überdeckung erfordert, dass jedes Wertepaar eines jeden Parameterpaares in mindestens einer Kombination enthalten sein muss.

Fehlerarten

Die häufigsten Fehlerzustände, die mit diesen Testverfahren aufgedeckt werden, sind Fehlerzustände in Zusammenhang mit den kombinierten Werten von zwei Parametern.

3.2.7 Anwendungsfallbasierter Test

Der anwendungsfallbasierte Test liefert transaktionale, auf Verhalten basierende Tests, die die beabsichtigte Nutzung der durch den Anwendungsfall spezifizierten Komponente oder des Systems nachahmen. Ein Anwendungsfall spezifiziert die Interaktionen zwischen den Akteuren und einer Komponente oder einem System, die ein Ergebnis erzielen. Die Akteure können Benutzer, externe Hardware oder andere Komponenten oder Systeme sein.

Ein allgemeiner Standard für Anwendungsfälle wird in [OMG-UML] bereitgestellt.

Anwendbarkeit

Der anwendungsfallbasierte Test wird meist beim System- und Abnahmetest angewendet. Er kann außerdem für Integrationstests verwendet werden, wenn das Verhalten der Komponenten oder Systeme durch Anwendungsfälle spezifiziert ist. Anwendungsfälle dienen häufig auch als Basis für den Performanztest, da sie eine realistische Nutzung des Systems abbilden. Die in den Anwendungsfällen beschriebenen Verhalten 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

Anwendungsfälle müssen, um gültig zu sein, realistische Benutzertransaktionen wiedergeben. Die Spezifikationen von Anwendungsfällen ist eine Form des Systementwurfs. Die Anforderungen bezüglich dessen, was Benutzer mit dem System bewältigen können, sollten von Benutzern oder Benutzervertretern kommen und vor dem Entwurf entsprechender Anwendungsfälle gegen organisatorische Anforderungen geprüft werden. Der Wert eines Anwendungsfalles wird gemindert, wenn dieser die organisatorischen oder Benutzeranforderungen nicht widerspiegelt oder die Erfüllung der Benutzeraufgaben eher behindert als unterstützt.

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

Überdeckung

Für die Mindestüberdeckung eines Anwendungsfalles ist ein Testfall für das Standardverhalten erforderlich und darüber hinaus genügend zusätzliche Testfälle, um alle alternativen Verhalten und Fehlerbehandlungen abzudecken. Wenn die Testmenge minimal sein soll, können mehrere Alternativverhalten mit einem Testfall überdeckt 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 dieser alternativen Verhalten zu einem Testfall zusammengefasst werden müssen (z.B. alternatives Abbruch- versus Nicht-Abbruch-Verhalten innerhalb eines "Retry"-Ausnahmeverhaltens).

Fehlerarten

Zu den mit diesem Testverfahren aufgedeckten Fehlerzuständen gehören die falsche Behandlung von spezifizierten Verhalten, fehlende alternative Verhalten, die inkorrekte Verarbeitung vorliegender 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, wie eine Bedingung erfüllt werden kann. Die Testfälle decken dann nicht nur jede Kombination von Bedingungen ab, sondern es sollten für die erstellten Äquivalenzklassen zusätzliche Testfälle generiert werden, um diese zu überdecken. Bei der Auswahl der geeigneten Testverfahren sollte der Test Analyst die Anwendbarkeit des Verfahrens, die Einschränkungen und Schwierigkeiten sowie die Ziele bezüglich der Überdeckung und die aufzudeckenden Fehlerzustände berücksichtigen. Diese Aspekte sind für die einzelnen, in diesem Kapitel behandelten Testverfahren beschrieben. Für bestimmte Situationen muss es nicht das einzig richtige und beste Verfahren geben. Die Kombination der passenden Testverfahren ist zumeist der effektivste Weg, die gesetzten Testziele zu erreichen – vorausgesetzt, dass für die korrekte Anwendung ausreichend Zeit und fachliche Kompetenz vorhanden sind.

3.3 Erfahrungsbasierte Testverfahren

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 Testverfahren reichen von "Schnelltests", bei denen der Tester keine formal vorgeplanten Aktivitäten durchführen muss, über vorgeplante Sitzungen unter Verwendung einer Test-Charta bis hin zu skriptbasierten Testsitzungen. Erfahrungsbasierte Testverfahren sind fast immer nützlich, sind jedoch besonders dann wertvoll, wenn damit die in der folgenden Liste enthaltenen Vorteile erzielt werden können.

Erfahrungsbasiertes Testen hat folgende Vorteile:

- Es kann eine gute Alternative zu strukturierten Vorgehensweisen sein, wenn die Systemdokumentation lückenhaft ist.
- Es kann 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 kann auch von Personen stammen, die in den Test nicht involviert sind (z.B. Business Analysten, Kunden oder Auftraggeber).
- Es kann den Entwicklern frühzeitiges Feedback liefern.
- Es hilft dem Team, sich mit der Software im Laufe ihrer Entwicklung 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 ist weniger geeignet, wenn Systeme eine detaillierte Testdokumentation benötigen.
- Es ist schwierig, eine hochgradige Wiederholbarkeit zu erreichen.
- Eine nur begrenzt möglich, die erzielte Überdeckung genau zu beurteilen.
- Die Tests sind für eine anschließende Automatisierung weniger geeignet.

Beim Einsatz reaktiver oder heuristischer Ansätze wenden Tester normalerweise erfahrungsbasiertes Testen an, da dieses schneller auf Ereignisse reagieren kann als vorgeplante Testvorgehensweisen. Außerdem finden Testausführung und Testauswertung gleichzeitig statt. Einige strukturierte Ansätze beim erfahrungsbasierten Testen sind nicht völlig dynamisch, d.h. die Tests werden nicht vollkommen gleichzeitig erstellt und durchgeführt. Dies kann beispielsweise der Fall sein, wenn Tester die intuitive Testfallermittlung nutzen, um bestimmte Aspekte des Testobjekts schon vor der Testausführung gezielt anzugehen.

Zu den nachfolgend behandelten erfahrungsbasierten Testverfahren werden zwar Ideen zur Überdeckung vorgestellt, allerdings sollte berücksichtigt werden, dass erfahrungsbasierte Testverfahren keine formalen Überdeckungskriterien haben.

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 Fehlerwirkungen zeigen wird, wenn ein ungültiges Passwort eingegeben wird, dann werden Tests ausgeführt, bei denen eine Vielzahl verschiedener Werte in das Passwortfeld eingegeben werden, um zu verifizieren, dass die Fehlhandlung tatsächlich begangen wurde und zu einem Fehlerzustand geführt hat, der bei der Testausführung 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. Die intuitive Testfallermittlung wird häufig zusammen mit anderen Testverfahren eingesetzt, um die Bandbreite der bestehenden Testfälle zu erweitern. Die intuitive Testfallermittlung kann auch beim Testen einer neuen Version der Software effektiv eingesetzt werden, um diese auf übliche 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 bewerten und variiert stark je nach Fachkompetenz und Erfahrung des Test Analysten.
- Es ist am besten, wenn das Verfahren von einem erfahrenen Tester verwendet wird, der sich gut auskennt mit den Fehlerarten, 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 Fehlertaxonomie wird die Überdeckung ermittelt, indem die Anzahl der getesteten Elemente durch die Gesamtzahl der Elemente der Taxonomie geteilt wird, d.h. die Überdeckung wird als Prozentsatz angegeben. Ohne Fehlertaxonomie ist die Überdeckung durch Erfahrung und Wissen der Tester sowie die zur Verfügung stehende Zeit beschränkt. Der Umfang der mit diesem Verfahren gefundenen Fehlerzustände hängt stark davon ab, wie gut der Tester die problematischen Bereiche treffen kann.

Fehlerarten

Typische Fehlerzustände sind solche, die in der jeweiligen Fehlertaxonomie definiert sind, oder solche, die der Test Analyst intuitiv "errät", und die durch Black-Box-Testverfahren möglicherweise nicht gefunden würden.

3.3.2 Checklistenbasiertes Testen

Beim checklistenbasierten Testen benutzen erfahrene Test Analysten eine abstrakte, allgemein gehaltene Liste von Punkten, welche beachtet, überprüft oder in Erinnerung gerufen werden müssen, oder sie nutzen eine Menge von Regeln oder Kriterien, gegen welche ein Testobjekt verifiziert werden muss. Diese Checklisten werden basierend auf Standards, Erfahrungen und anderen Überlegungen zusammengestellt. Beispiel für einen checklistenbasierten Test ist eine Checkliste mit Standards für Benutzungsschnittstellen, die als Grundlage für den Test einer Anwendung dient. Bei agiler Softwareentwicklung 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 Themenbereich der Checkliste gut kennt (Beispiel: Um eine Checkliste für die Benutzungsschnittstelle erfolgreich zu verwenden, sollte sich ein Test Analyst zwar mit dem Testen von Benutzungsschnittstellen gut auskennen, aber nicht zwingend mit dem konkreten System unter Test). Da Checklisten abstrakt sind und die Details über die einzelnen Schritte, die normalerweise in Testfällen und Testabläufen beschrieben sind, nicht enthalten, nutzt der Tester sein Wissen, um die Lücken zu füllen. Dadurch, dass die detaillierten Schritte weggelassen werden, ist der Wartungsaufwand gering und die Checklisten 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 Erstellung 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 benutzen, um die Punkte der Checklisten zu erfüllen. Dies kann zu unterschiedlichen Testergebnissen 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. Checklisten sollten stets gewartet werden, um sicherzustellen, dass sie die wichtigen Aspekte der Software unter Test überdecken.

Überdeckung

Die Überdeckung kann ermittelt werden, indem man die Anzahl der getesteten Punkte der Checkliste durch die Gesamtzahl der Checklistenpunkte dividiert, und die Überdeckung 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 Test Analyst, der die Checkliste verwendet.

Fehlerarten

Die typische Fehlerzustände, die mit diesem Verfahren aufgedeckt werden, verursachen Fehlerwirkungen während des Testens, die als Folge variierender Daten, Handlungsaktionen oder dem allgemeinen Ablauf auftreten.

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 durch, und berichten über die Testergebnisse. Die Tester passen die Testziele während der Testausführung dynamisch an und erstellen nur eine leichtgewichtige Dokumentation [Whittaker09].

Anwendbarkeit

Gutes exploratives Testen ist geplant, interaktiv und kreativ. Es ist wenig Dokumentation über das zu testende System erforderlich. Daher wird exploratives Testen häufig angewendet, wenn keine Dokumentation verfügbar ist, oder wenn die vorhandene Dokumentation für andere Testverfahren nicht ausreichend ist. Exploratives Testen wird häufig als Erweiterung anderer Testverfahren eingesetzt und dient als Grundlage für die Erstellung zusätzlicher Testfälle. Exploratives Testen wird häufig bei agiler Softwareentwicklung eingesetzt, um User-Story-basierte Tests schnell und flexibel mit nur minimaler Dokumentation durchzuführen. Das Verfahren kann jedoch auch in Projekten eingesetzt werden, die ein sequenzielles Entwicklungsmodell nutzen.

Einschränkungen/Schwierigkeiten

Exploratives Testen kennt keine Überdeckungsmessungen, zudem kann es schwierig sein, die durchgeführten Tests zu reproduzieren. Als eine Methode des Managements beim explorativen Testen können Test-Chartas erstellt werden, die die in einer Testsitzung abzudeckenden Aufgaben und den verfügbaren Zeitrahmen für das Testen festlegen. Am Ende jeder Testsitzung bzw. einer Menge von Testsitzungen hält der Test Manager eine Abschlussbesprechung, um die Testergebnisse zu sammeln und die Test-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 zugewiesene Zeit und die geplante Überdeckung gemeinsam mit den anderen Testaufwänden verfolgen.

Da Reproduzierbarkeit beim explorativen Testen schwierig zu erreichen sein kann, kann dies zu Problemen führen, wenn die Schritte zur Reproduktion einer Fehlerwirkung wieder benötigt werden. Manche Organisationen nutzen die Mitschnittfunktion (Capture/Playback) eines Testautomatisierungswerkzeugs, um die beim explorativen Testen durchgeführten Schritte aufzuzeichnen. Dies liefert eine vollständige Aufzeichnung aller Aktivitäten während einer explorativen Testsitzung (bzw. 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, da diese allerdings nicht die Interaktion mit der grafischen Benutzeroberfläche (GUI) erfassen, werden auch die erwarteten Ergebnisse nicht aufgezeichnet. 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, dass auch während der Durchführung von explorativen Tests Notizen gemacht werden, um die Reproduzierbarkeit zu unterstützen, falls erforderlich.

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 Test-Charta kann auch bestimmt werden, worauf sich der Testaufwand konzentrieren soll, was zum Umfang der Testsitzung gehört und was nicht, und welche Ressourcen für die Durchführung der geplanten Tests vorzusehen sind. Eine Testsitzung kann auf bestimmte Fehlerarten und auf andere potenzielle

Problembereiche fokussiert sein, 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 werden, Probleme in funktionalen Grenzbereichen, sowie Probleme im Zusammenhang mit dem Ablauf. Auch Performanz- und IT-Sicherheitsprobleme werden manchmal beim explorativen Testen aufgedeckt.

3.3.4 Fehlerbasiertes Testverfahren

Beim fehlerbasierten Testverfahren dient die Art des gesuchten Fehlerzustands als Basis für den Testentwurf, wobei die Tests systematisch von dem abgeleitet werden, was über die Fehlerart bekannt ist. Anders als beim Black-Box-Testverfahren, bei dem die Tests aus der Testbasis abgeleitet werden, werden beim fehlerbasierten Testverfahren die Tests aus Listen abgeleitet, die auf Fehlerzustände fokussiert sind. Diese Listen können nach Fehlerarten bzw. Fehlerzuständen, Grundursachen, Symptomen von Fehlerwirkungen und anderen fehlerbezogenen Daten kategorisiert sein. Standardtaxonomien gelten für mehrere Arten von Software und sind nicht produktspezifisch. Bei Verwendung dieser wird das branchenspezifische Standardwissen genutzt, um bestimmte Tests abzuleiten

Beim fehlerbasierten Testen können auch Listen mit identifizierten Risiken und Risikoszenarien als Grundlage für zielgerichtetes Testen verwendet werden. Dieses Testverfahren ermöglicht es dem Test Analyst, bestimmte Arten von Fehlerzuständen gezielt zu adressieren, oder eine Liste bekannter und häufig auftretender Fehlerzustände bzw. Fehlerarten systematisch abzuarbeiten. Basierend auf diesen Informationen erstellt der Test Analyst die Testbedingungen und Testfälle, 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 verschiedene Fehlertaxonomien, die auf bestimmte Testarten (z.B. Gebrauchstauglichkeitstests) fokussiert sind. Es ist wichtig, eine Taxonomie zu finden und auszuwählen, die auf die Software unter Test anwendbar ist. Für innovative Software gibt es möglicherweise (noch) gar keine Taxonomien. Manche Organisationen haben ihre eigenen Taxonomien mit wahrscheinlichen oder häufig auftretenden Fehlerzuständen zusammengestellt. Unabhängig davon, welche Fehlertaxonomie verwendet wird, muss vor Testbeginn die erwartete Überdeckung festgelegt werden.

Überdeckung

Das Verfahren stellt Überdeckungskriterien bereit, 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 der 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 Menge von Tests vollständig ist, sondern dass dieses Testverfahren für die betrachteten Fehlerarten keine weiteren sinnvollen Tests mehr nahe legt.

Fehlerarten

Die Fehlerarten, die gefunden werden, hängen gewöhnlich von der verwendeten Fehlertaxonomie ab. Wenn beispielsweise eine Benutzungsschnittstellen-Taxonomie 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-Testverfahren und erfahrungsbasierte Testverfahren sind am effektivsten, wenn sie zusammen eingesetzt werden. Erfahrungsbasierte Testverfahren füllen die Lücken bei der Erreichung der Testziele, die sich aus systematischen Schwächen von Black-Box-Testverfahren ergeben können.

Es gibt nicht ein einziges perfektes 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 müssen die Art des Projektes, der Zeitplan, der Zugang zu Informationen, die Fähigkeiten des Testers und weitere Faktoren berücksichtigt werden, die die Auswahl beeinflussen können.

Bei der Auswahl der am 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 (SUMI), Website Analysis and MeasureMent Inventory (WAMMI), 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, funktionale Korrektheit und funktionale Angemessenheit zu testen
- TA-4.2.2 (K2) Die typischen Fehlerzustände erläutern, auf die das Testen der funktionalen Vollständigkeits-, Korrektheits- und Angemessenheitsmerkmale abzielen soll
- TA-4.2.3 (K2) Erläutern, in welcher Stufe des Softwareentwicklungslebenszyklus 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 vorherige 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], der als Leitlinie verwendet wird. Das Softwarequalitätsmodell nach ISO unterteilt die Produktqualität in verschiedene Produktqualitätsmerkmale, die jeweils weitere Untermerkmale haben können. Diese sind in der nachstehenden Tabelle aufgeführt, aus der ebenfalls hervorgeht, welche Qualitätsmerkmale/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 ¹ | Erkennbare Angemessenheit, Erlernbarkeit, Operabilität, Ästhetik der Benutzungsschnittstelle, Benutzerfehlerschutz, Zugänglichkeit | X | |
| Performanz | Zeitverhalten, Ressourcennutzung, Kapazität | | X |
| Wartbarkeit | Analysierbarkeit, Änderbarkeit, Testbarkeit, Modularität, Wiederverwendbarkeit | | X |
| Übertragbarkeit | Anpassbarkeit, Installierbarkeit, Austauschbarkeit | X | X |
| IT-Sicherheit | Vertraulichkeit, Datenintegrität, Nichtabstreitbarkeit, Zurechenbarkeit, Authentizität | | X |
| Kompatibilität | Koexistenz | | X |
| | Interoperabilität | X | |

Auch wenn die Zuständigkeit in verschiedenen Organisationen unterschiedlich aufgeteilt sein kann, halten sich zugehörige 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. Für das 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

¹ In vorliegenden Lehrplan wird das englische Qualitätsmerkmal „Usability“ mit Gebrauchstauglichkeit und das Untermerkmal „accessibility“ mit Zugänglichkeit übersetzt. Der ISTQB Lehrplan Usability-Testing verwendet für die zwei englischen Qualitätsmerkmale die Begriffe Usability und Barrierefreiheit.

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 Experten für Gebrauchstauglichkeit oder Benutzererfahrung durchgeführt werden.

Auch wenn Test Analysten üblicherweise nicht für jene Qualitätsmerkmale verantwortlich sind, die einen mehr technisch ausgerichteten Ansatz erfordern, sollten sie diese anderen Qualitätsmerkmale kennen und wissen, welche Bereiche sich beim Testen überschneiden. Beispiel 1: Ein Produkt, das den Performanztest nicht besteht, wird wahrscheinlich auch den Gebrauchstauglichkeitstest nicht bestehen, wenn es zu langsam ist und die Benutzer es nicht effektiv nutzen können. Beispiel 2: Wenn es bei einem Produkt Probleme mit der Interoperabilität bei einigen Komponenten gibt, dann ist es wahrscheinlich nicht bereit für den Übertragbarkeitstest, 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 Aufgabenschwerpunkt von Test Analysten. Funktionalen Eignungstests sind darauf fokussiert, "was" das Produkt leistet. Diese Tests basieren in der Regel auf Anforderungen, einer Spezifikation, spezifischem Branchenwissen oder einem implizierten Bedarf. Funktionale Eignungstests 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 durch Schnittstellen gekoppelten Komponenten getestet, die eine einzelne definierte Funktion erbringen. In der Systemteststufe beinhalten die funktionalen Eignungstests das Testen der funktionalen Eignung des gesamten Systems. Bei Multisystemen werden mit funktionalen Eignungstests hauptsächlich die gesamten integrierten Systeme End-to-End getestet. Bei den funktionalen Eignungstests werden viele unterschiedliche Testverfahren eingesetzt (siehe Kapitel 3).

Bei agiler Softwareentwicklung 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 Funktionalität, die nicht geändert wurde

Zusätzlich zu den funktionalen Eignungstests, die in diesem Abschnitt beschrieben sind, gibt es auch bestimmte Qualitätsmerkmale, die in den Zuständigkeitsbereich von Test Analysten fallen, und als nicht-funktional gelten (d.h. sie konzentrieren sich darauf, "wie" das Produkt die Funktionalität liefert).

4.2.1 Testen der funktionalen Korrektheit

Tests der funktionalen Korrektheit prüfen, ob die spezifizierten oder impliziten Anforderungen eingehalten wurden; dabei kann auch die Richtigkeit bzw. Genauigkeit von Berechnungen geprüft werden. Funktionale Korrektheitstests nutzen viele der in Kapitel 3 beschriebenen Testverfahren. Oft dient eine Spezifikation oder ein vorhandenes System als Testorakel. Funktionale Korrektheitstests können in jeder Teststufe durchgeführt werden. Mit diesen Tests soll die inkorrekte Handhabung von Daten oder Situationen aufgedeckt werden.

4.2.2 Testen der funktionalen Angemessenheit

Tests der funktionalen Angemessenheit bewerten und validieren, 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. Das Testen auf funktionaler Angemessenheit wird meist beim Systemtest durchgeführt, kann aber auch in späteren Phasen des Integrationstests

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

Das Testen auf funktionalen Vollständigkeit wird durchgeführt, um die Überdeckung der spezifizierten Aufgaben und Benutzerziele durch die implementierte Funktionalität zu ermitteln. Die Verfolgbarkeit zwischen Spezifikationselementen (z.B. Anforderungen, User-Stories, Anwendungsfälle) und der implementierten Funktionalität (z.B. Funktion, Komponente, Ablauf) ist wesentlich, um die erforderliche funktionale Vollständigkeit zu ermitteln. Die Messung der funktionalen Vollständigkeit kann je nach Teststufe und/oder verwendetem Softwarelebenszyklus variieren. Beispielsweise kann die funktionale Vollständigkeit für agile Softwareentwicklung auf den implementierten User-Stories und Features basieren. Beim Systemintegrationstest kann die funktionale Vollständigkeit auf die Überdeckung von übergeordneten Geschäftsprozessen fokussiert sein.

Die Ermittlung der funktionalen Vollständigkeit wird in der Regel durch Testmanagementwerkzeuge unterstützt, vorausgesetzt der Test Analyst sorgt für die Verfolgbarkeit zwischen den Testfällen und den Elementen der funktionalen Spezifikation. Wenn der Grad an funktionaler Vollständigkeit geringer als erwartet ist, so ist das ein Hinweis darauf, 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 dabei auf die Fähigkeit der Systeme bzw. Komponenten, Informationen auszutauschen und die ausgetauschten Informationen anschließend zu nutzen. Die Tests sollten 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 Testfällen der funktionalen Eignung getestet, die die verschiedenen Datenaustauschpunkte in den Umgebungen prüfen.

Interoperabilität betrifft das Zusammenwirken verschiedener Komponenten und Softwaresysteme. Software mit guten Interoperabilitätseigenschaften lässt sich leicht mit verschiedenen anderen Systemen integrieren, ohne dass größere Änderungen nötig sind oder erhebliche Auswirkungen auf das nicht-funktionale Verhalten hat. 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 Designmerkmale 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 Multisystem basieren
- Systeme auf Basis des Internets der Dinge
- Web Services mit Konnektivität zu anderen Systemen

Diese Testart wird während des Komponentenintegrations- und Systemintegrationstests durchgeführt. Beim Systemintegrationstest wird mit dieser Testart geprüft, wie gut das fertig entwickelte System mit anderen Systemen interagiert. 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 stattfinden können. 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 ausschließlich in den Systemarchitektur- und Entwurfsdokumenten spezifiziert. Daher muss der Test Analyst in der Lage und bereit sein, diese Dokumente zu untersuchen, um die Punkte des Datenaustausches zwischen den Systemen und zwischen dem System und seiner Umgebung zu bestimmen. Nur so kann sichergestellt werden, dass auch alle Datenaustauschpunkte getestet werden. Testverfahren wie die Äquivalenzklassenbildung, Grenzwertanalyse, der Entscheidungstabellentest, der Zustandsübergangstest, das anwendungsfallbasierte sowie paarweise 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

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

Es ist wichtig zu verstehen, warum Benutzer bei der Benutzung des Systems Schwierigkeiten oder kein positives Benutzererlebnis haben könnten (z.B. bei der Nutzung von Entertainment-/Infotainmentsoftware). 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 gemäß des ISO-Standards 25010
- Benutzererlebnis (engl. user experience, UX) als eine Generalisierung der Gebrauchstauglichkeit
- Zugänglichkeit als Untermerkmal von Gebrauchstauglichkeit

Gebrauchstauglichkeit

Gebrauchstauglichkeitstests zielen 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 oder zufriedenstellend 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 Untermerkmale der Gebrauchstauglichkeit [ISO 25010] aufgeführt; die jeweiligen Definitionen sind unter [ISTQB_GLOSSARY] zu finden:

- Erkennbare Angemessenheit (d.h. Verständlichkeit)
- Lernfähigkeit
- Operabilität
- Ästhetik der Benutzungsschnittstelle (d.h. Attraktivität)
- Benutzerfehlerschutz

- Zugänglichkeit (siehe unten)

Benutzererlebnis

Bei der Evaluierung des Benutzererlebnisses geht es um das gesamte Benutzererlebnis und nicht nur um die direkte Interaktion mit dem Softwareprodukt. Dies ist besonders wichtig bei Testobjekten, bei denen Faktoren wie Spaß 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 (auch Barrierefreiheit genannt) zur Software für Menschen mit besonderen Bedürfnissen oder Einschränkungen zu berücksichtigen. Dies gilt auch für Menschen mit Behinderungen. Beim Testen der Zugänglichkeit sollten die einschlägigen Normen, wie die Richtlinie für zugängliche 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) und Section 508 (US), berücksichtigt werden. Wie die Gebrauchstauglichkeit muss auch die Zugänglichkeit beim Entwurf berücksichtigt werden. Das Testen erfolgt oft im Integrationstest und wird im Systemtest und 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 adressieren die Möglichkeiten, die Benutzern mit Behinderungen die Interaktion mit der Anwendung ermöglichen. Dazu gehören folgende Maßnahmen:

- Spracherkennung für Eingaben
- Sicherstellen, dass dem Benutzer gleichwertige Textalternativen für alle angezeigten Nicht-Text-Inhalte zur Verfügung gestellt werden
- Möglichkeit bieten, die Textgröße zu skalieren, ohne dass Inhalte oder Funktionalität verloren gehen

Leitlinien für Zugänglichkeit unterstützen den Test Analyst beim Testen, indem sie als Informationsquelle und Checkliste 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 ungünstige Farbauswahl in Webseiten, die gegen die Richtlinien für Farbenblindheit verstoßen.

4.2.5.2 Verfahren für die Evaluierung der Gebrauchstauglichkeit

Gebrauchstauglichkeit, Benutzererlebnis und Zugänglichkeit können durch einen oder mehrere der folgenden Verfahren getestet werden:

- Gebrauchstauglichkeitstests
- Reviews der Gebrauchstauglichkeit
- Benutzerbefragungen und Fragebögen

Gebrauchstauglichkeitstest

Gebrauchstauglichkeitstests bewerten, wie einfach es für die Benutzer ist, das System zu benutzen bzw. zu erlernen, um damit spezifizierte Ziele in bestimmten Anwendungskontexten zu erreichen. Beim Gebrauchstauglichkeitstest wird folgendes gemessen:

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

Es ist zu beachten, dass Entwurf und Spezifikation von Gebrauchstauglichkeitstests häufig vom Test Analyst in Zusammenarbeit mit Testern durchgeführt wird, die auf den Gebrauchstauglichkeitstest spezialisiert sind, und mit Gebrauchstauglichkeits-Experten, die sich mit dem menschenzentrierten Gestaltungsprozess auskennen (siehe [ISTQB_UT_SYL] für Details).

Reviews auf Gebrauchstauglichkeit

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

Benutzerbefragungen und Fragebögen

Mit Befragungen 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 (engl. 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 Ende-/Abnahmekriterien verwendet werden können.

4.2.6 Übertragbarkeitstest

Übertragbarkeitstests beziehen sich auf den Grad der Übertragbarkeit (auch Portabilität genannt) einer Softwarekomponente oder eines Systems in ihre vorgesehene Umgebung, entweder als eine neue Installation oder von einer Umgebung in eine andere.

Die Klassifikation der Produktqualitätsmerkmale nach ISO 25010 umfasst die folgenden Untermerkmale für Übertragbarkeit:

- 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 Installierbarkeitstest

Installierbarkeitstests betreffen 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:

- Validierung, 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).
- Testen der funktionalen Korrektheit von Installations- und Deinstallationsverfahren.
- Durchführung von funktionalen Eignungstests nach einer Installation oder Deinstallation, um eventuell eingeschleuste Fehlerzustände (z.B. falsche Konfigurationen, nicht verfügbare Funktionen) zu erkennen.
- Erkennen von Gebrauchstauglichkeitsproblemen bei Installations- und Deinstallationsprozeduren (z.B. Überprüfung, ob die Benutzer bei Durchführung der Prozedur verständliche Anweisungen und Feedback/Fehlermeldungen erhalten).

4.2.6.2 Anpassbarkeitstest

Beim Anpassbarkeitstest wird geprüft, ob eine bestimmte Softwareanwendung effektiv und effizient angepasst werden kann, um in allen vorgesehenen Zielumgebungen (Hardware, Software, Middleware, Betriebssystem etc.) korrekt zu funktionieren. Der Test Analyst unterstützt den Anpassbarkeitstest, indem er die vorgesehenen Zielumgebungen (z.B. Versionen verschiedener unterstützter mobiler Betriebssysteme, verschiedene Browserversionen, die verwendet werden können) identifiziert und Tests entwirft, die die Kombinationen dieser Umgebungen abdecken. Diese Zielumgebungen werden dann anhand einer Auswahl funktionaler Eignungstestfälle 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, die auf dem Internet der Dinge basieren, relevant sein, wo der Austausch unterschiedlicher Hardware-Geräte und/oder Softwareinstallationen häufig vorkommt. Beispiel: Ein Hardwaregerät, das in einem Warenlager zur Erfassung und Kontrolle der Lagerbestände verwendet wird, kann durch ein moderneres Hardware-Gerät (z.B. mit einem besseren Scanner) ersetzt werden, oder es gibt ein Software-Upgrade für eine installierte Software, das es ermöglicht, automatisch Nachbestellungen für Lagerbestände an das System eines Lieferanten zu senden.

Austauschbarkeitstests können vom Test Analyst 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 Checkliste identifizieren

5.1 Einführung

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 gelieferten Qualität.

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 für alle Reviews allgemein gehalten sein, oder sie können sich zielgerichtet mit bestimmten Qualitätsmerkmalen, Themenbereichen oder Dokumenten 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 Punkte enthält, die noch offen sind, dass Formatierung und ähnliche Elemente korrekt bzw. konform sind). Eine spezifische Checkliste für ein Anforderungsdokument könnte vorgeben, dass überprüft wird, ob die Begriffe "soll" und "sollte" richtig verwendet sind, oder ob jede Anforderung testbar ist.

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

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

- An die spezifischen Kompetenzen von Programmierern/Systemarchitekten oder Testern – im Fall des Test Analysten wäre eine Checkliste für Testerkompetenzen am besten geeignet
- 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 ein bestimmtes Spezifikationselement, 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 das Review von Code oder Architektur verwendet werden

5.2.1 Reviews von Anforderungen

Checklisten, die nach Anforderungen ausgerichtet sind, können beispielsweise die folgenden Punkte enthalten:

- Quelle der Anforderung (z.B. Person, Abteilung)
- Testbarkeit der einzelnen Anforderungen
- Priorität der einzelnen Anforderungen
- Abnahmekriterien für die einzelnen Anforderungen
- Verfügbarkeit einer Aufrufstruktur für Anwendungsfälle (falls zutreffend)
- Eindeutige Kennung 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 eine Anforderung, die nicht testbar ist (d.h. die so definiert ist, dass der Test Analyst nicht bestimmen kann, wie sie zu testen ist) einen Fehlerzustand enthält. Beispiel: Eine Anforderung, die besagt "Die Software sollte sehr benutzerfreundlich sein", ist nicht testbar. Wie soll ein Test Analyst bestimmen, ob die Software benutzerfreundlich oder gar sehr benutzerfreundlich ist? Wenn die Anforderung stattdessen festlegt "Die Software muss den Gebrauchstauglichkeitsstandards entsprechen, die im Dokument Gebrauchstauglichkeitsstandard, Version xxx, spezifiziert sind", dann ist die Anforderung testbar, vorausgesetzt das spezifizierte Dokument existiert. Hierbei handelt es sich außerdem um eine übergreifende Anforderung, die jedes Element der Benutzungsschnittstelle betrifft. In diesem Fall könnte diese eine Anforderung bei einer komplexen Anwendung zu vielen einzelnen Testfällen führen. Auch ist die Verfolgbarkeit von dieser Anforderung bzw. vom festgelegten Gebrauchstauglichkeitsstandard zu den Testfällen kritisch; falls es nämlich Änderungen der Gebrauchstauglichkeitsspezifikation gibt, auf die in der Anforderung verwiesen wird, müssten alle 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 bestanden hat oder nicht, oder wenn kein Test entwickelt werden kann, der bestehen oder fehlschlagen kann. Beispiel für eine nicht testbare Anforderung: "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 Checkliste² für Reviews von Anwendungsfällen könnten folgende Fragen enthalten sein:

- Ist das Standardverhalten genau spezifiziert?
- Sind alle alternativen Verhalten identifiziert, einschließlich der Fehlerbehandlung?
- 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

Bei agiler Softwareentwicklung liegen die Anforderungen in der Regel als User-Stories mit kleinen Inkrementen nachweisbarer Funktionalitäten vor. Während es sich bei Anwendungsfällen um eine Benutzertransaktion handelt, die mehrere Funktionalitätsbereiche durchläuft, ist eine User-Story eine isoliertere Einheit, deren Umfang im Allgemeinen durch die Zeit bestimmt wird, die für ihre Entwicklung notwendig ist. Eine Checkliste³ 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 sind diese testbar?
- Ist die Funktionalität klar spezifiziert?
- Gibt es Abhängigkeiten zwischen dieser User-Story und anderen User-Stories?
- Ist die User-Story priorisiert?
- Ist die User-Story nach dem üblichen Format erstellt, d.h.: Als ein <Art des Benutzer>, möchte ich <irgendein Ziel>, so dass <irgendein Grund>. [Cohn04]

Falls eine User-Story eine neue Benutzungsschnittstelle definiert, dann wäre es angebracht, eine allgemeine Checkliste (siehe Beispiel oben) für die User-Story und eine detaillierte Checkliste für die Benutzungsschnittstelle zu verwenden.

² Die Prüfungsfrage liefert zur Beantwortung einen Teil einer Anwendungsfall-Checkliste

³ Die Prüfungsfrage liefert zur Beantwortung einen Teil einer Anwendungsfall-Checkliste

5.2.3 Checklisten anpassen

Checklisten können individuell an die folgenden Grundlagen angepasst werden:

- Organisation/Unternehmen (z.B. Berücksichtigung von Unternehmensrichtlinien, -standards, -konventionen, rechtlichen Einschränkungen)
- Projekt-/Entwicklungsaufwand (z.B. Schwerpunkt, technische Standards, Risiken)
- Art des Arbeitsergebnisses, das einem Review unterzogen wird (beispielsweise können Code-Reviews auf eine bestimmte Programmiersprache ausgerichtet sein)
- Risikostufe des des Arbeitsergebnisses, das einem Review unterzogen wird
- Vorgesehene Testverfahren

Gute Checklisten decken Probleme auf, und sie stoßen Diskussionen über andere Punkte an, auf die die Checkliste möglicherweise nicht verweist. Wenn beim Review verschiedene Checklisten kombiniert werden, dann ist das 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.

Für weitere Informationen über Reviews und Inspektionen, siehe [Gilb93] und [Wiegers03]. Weitere Beispiele für Checklisten finden Sie in den Referenzen in 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 Schlüsselwortgetriebenes Testen

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 beim schlüsselwortgetriebenen Testen mit und bringen ihre Erfahrung aus dem Geschäftsbereich und hinsichtlich der Systemfunktionalität ein.

Weitere Informationen zum Thema Testautomatisierung und zur Rolle des Testautomatisierungsentwicklers finden Sie im ISTQB Advanced Level-Lehrplan Test Automation Engineer [ISTQB_TAE_SYL].

6.2 Schlüsselwortgetriebenes Testen

Schlüsselwortgetriebenes Testen ist einer der wichtigsten Testautomatisierungsansätze 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 übergeordnete 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 Abfolge von Schlüsselwörtern (zusammen mit den relevanten Testdaten) spezifiziert. [Buwalda02]

Für die Testautomatisierung werden die einzelnen Schlüsselwörter als ein oder mehrere auszuführende Testskripte implementiert. Die Werkzeuge lesen die mit Schlüsselwörtern erstellten Testfälle ein und rufen die entsprechenden Testskripte auf, welche die Funktionalität des jeweiligen Schlüsselworts implementieren. Die Testskripte sind sehr modular implementiert, damit sie leicht den einzelnen Schlüsselwörtern zugeordnet werden können. Für die Implementierung dieser modularen Skripte sind Programmierkenntnisse erforderlich.

Die wichtigsten Vorteile des schlüsselwortgetriebenen Testens sind:

- Schlüsselwörter, die sich auf eine bestimmte Anwendung oder Geschäftsbereich beziehen, können von Fachexperten definiert werden. Das macht die Spezifikation der Testfälle effizienter.
- Personen mit vorwiegend Fachexpertise können, nachdem die Schlüsselwörter in Form von Testskripten implementiert wurden, von der automatischen Testfallausführung profitieren, ohne den zugrunde liegenden Automatisierungscode verstehen zu müssen.
- Testfälle, die modular unter Verwendung von Schlüsselwörtern erstellt wurden, sind für Testautomatisierungsentwickler leichter 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 sich im Klaren sein, dass für die Implementierung der Schlüsselwörter noch die entsprechenden Skripte entwickelt werden müssen. Wenn die zu verwendenden Schlüsselwörter und Daten definiert sind, erstellen Technical Test Analysten oder Testautomatisierungsentwickler aus den Schlüsselwörtern und untergeordneten Aktionen der Geschäftsprozesse automatisierte Testskripte.

Während das schlüsselwortgetriebene Testen normalerweise in der Systemtestphase durchgeführt werden, kann die Erstellung des Codes bereits während des Testentwurfs beginnen. In iterativen

Vorgehensweisen, insbesondere bei kontinuierlicher Integration/kontinuierlichem Deployment, ist die Entwicklung der automatisierten Tests ein kontinuierlicher Prozess.

Nachdem die Schlüsselwörter und Daten erstellt wurden, sind die Test Analysten für die Ausführung der Testskripte, die die Schlüsselwörter beinhalten, und die Analyse der aufgedeckten 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 wurde. Der erste Schritt bei der Fehlersuche ist meist die manuelle Durchführung des betroffenen Tests mit denselben Daten, um herauszufinden, ob die Fehlerwirkung durch die Anwendung selbst versucht wurde. Wenn beim manuellen Test keine Fehlerwirkung aufgedeckt wurde, dann sollte der Test Analyst die Reihenfolge der Tests prüfen, die zur Fehlerwirkung geführt haben. 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 Fehlerdiagnose an den Technischen Test Analyst 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 Testaufwands steigern können (z.B. indem sie helfen, in der verfügbaren Zeit eine bessere Testüberdeckung 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, bereitgestellt durch den Test Analysten. Testentwurfswerkzeuge werden häufig so konzipiert und erstellt, dass sie mit bestimmten Formaten und bestimmten Werkzeugen arbeiten, z.B. mit bestimmten Anforderungsmanagementwerkzeugen.

Testentwurfswerkzeuge können Informationen liefern, anhand derer der Test Analyst bestimmen kann, welche weiteren Tests noch 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 folgende Vorteile:

- Analyse eines Dokuments (z.B. Anforderungsdokument) oder sogar von Quellcode, um die beim Testen benötigten Daten zur Erzielung einer vordefinierten Überdeckung zu bestimmen.
- Bereinigung oder Anonymisierung von Datensätzen aus einem Produktionssystem, so dass alle persönlichen Informationen entfernt, die interne Integrität dieser Daten jedoch erhalten bleibt. Die so bereinigten Daten können dann im Test verwendet werden, ohne eine IT-Sicherheitslücke oder den Missbrauch der persönlichen 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, die Datenbankstruktur zu analysieren, um zu bestimmen, welche Eingaben vom Test Analyst benötigt werden.

6.3.3 Automatisierte Testausführungswerkzeuge

Testausführungswerkzeuge werden von Test Analysten in allen Teststufen eingesetzt, um automatisierte Tests durchzuführen und das Istergebnis der Tests zu überprüfen. Testausführungswerkzeuge verfolgen normalerweise eines oder mehrere der folgenden Ziele:

- Kostenreduktion (in Bezug auf Aufwand und/oder Zeit)
- Durchführung von weiteren Tests
- Durchführung derselben Tests in unterschiedlichen Umgebungen
- Testausführung wird wiederholbarer gemacht
- Ausführung von Tests, die manuell nicht ausgeführt werden können (z.B. umfangreiche Prüfungen der Datenvalidierung)

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

Die Rentabilität von Testausführungswerkzeugen ist am höchsten, wenn diese zur Automatisierung von Regressionstests eingesetzt werden; Gründe 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. Diese Tests werden häufig ausgeführt, das Testergebnis wird schnell benötigt, und es ist 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 (2011) 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
- [OMG-DMN] Object Management Group: OMG® Decision Model and Notation™, Version 1.3, December 2019; url: www.omg.org/spec/DMN/, Chapter 8
- [OMG-UML] Object Management Group: OMG® Unified Modeling Language®, Version 2.5.1, December 2017; url: www.omg.org/spec/UML/
- [RTCA DO-178C/ED-12C] Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12C, 2013, Kapitel 1

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_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 2019
- [ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2018
- [ISTQB_GLOSSARY] Standard glossary of terms used in Software Testing, url: <https://glossary.istqb.org/>
- [ISTQB_TAE_SYL] ISTQB® Advanced Level Test Automation Engineer Syllabus, Version 2017
- [ISTQB_UT_SYL] ISTQB® Foundation Level Specialist Syllabus Usability Testing, Version 2018

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
- [Chow1978] T.S. Chow, Testing Software Design Modeled by Finite-State Machines, IEEE Transactions on Software Engineering vol. SE-4, issue 3, May 1978, pp. 178-187
- [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
- [Forgács19] István Forgács, Attila Kovács, "Practical Test Design", BCS, 2019, ISBN 978-1-780-1747-23
- [Gilb93]: Tom Gilb, Dorothy Graham, "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
- [Mosley93] Daniel J. Mosley, The Handbook of MIS Application Software Testing, Yourdon Press, Prentice-Hall. 1993, ISBN 978-0-13-907007-5
- [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: 978-1-107-17201-2
- [vanVeenendaal12]: Erik van Veenendaal, "Practical risk-based testing." Product Risk Management: The PRISMA Method", UTN Publishers, 2012, ISBN 978-94-9098-607-0
- [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
 - Defect taxonomy: www.testingeducation.org/a/bsct2.pdf
 - Sample defect 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 Eignung | 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 Benutzungsschnittstelle | Attraktivität | Neue Bezeichnung |
| Zugänglichkeit | | Neues Untermerkmal |
| Kompatibilität | | Neue Definition |
| Interoperabilität | | |
| Koexistenz | Koexistenz | Siehe Technical Test Analyst-Lehrplan Ehemals Untermerkmal von Übertragbarkeit |
| Übertragbarkeit (Portabilität) | Übertragbarkeit | |
| Installierbarkeit | Installierbarkeit | |
| Anpassbarkeit | Anpassbarkeit | |
| Austauschbarkeit | Austauschbarkeit | |

9. Index

- 0-Switch, 34
- abstrakter Testfall, 11, 15, 16
- agile Softwareentwicklung, 13, 14, 55
- Aktionswörter, 58
- Aktivitäten, 13
- Angemessenheitstest, 47
- Anpassbarkeitstest, 52
- anwendungsfallbasierter Test, 27, 38
- Äquivalenzklassenbildung, 27, 29
- Ästhetik der Benutzungsschnittstelle, 45
- Austauschbarkeitstest, 52
- Benutzererlebnis, 45
- Benutzererlebnis-Evaluierung, 50
- Benutzerfehlerschutz, 45
- Black-Box-Testverfahren, 27, 28
- breadth-first, 25
- Checklisten in Reviews, 54
- checklistenbasiertes Review, 54
- checklistenbasiertes Testen, 27
- checklistenbasiertes Testen, 41
- checklistenbasiertes Testen, 41
- das beste Verfahren anwenden, 44
- depth-first, 25
- Eintrittswahrscheinlichkeit des Risikos, 24
- Endekriterien, 11
- Entscheidungstabelle, 27
- Entscheidungstabellentest, 31
- erfahrungsbasierte Testverfahren, 20
- erfahrungsbasiertes Testen, 27
- erfahrungsbasiertes Testverfahren, 27, 40, 44
- erfahrungsbasiertes Testverfahren, 39
- Erlernbarkeit, 45
- exploratives Testen, 27, 42
- fehlerbasiertes Testentwurfsverfahren, 27
- fehlerbasiertes Testverfahren, 43
- Fehlertaxonomie, 27
- funktionale Angemessenheit, 45
- funktionale Eignung, 45
- funktionale Korrektheit, 45
- funktionale Vollständigkeit, 45
- Gebrauchstauglichkeit, 45
- Gebrauchstauglichkeitstest, 49
- Genauigkeitstest, 47
- Grenzwertanalyse, 27, 30
- Heuristik, 51
- Installierbarkeit, 52
- Interoperabilität, 45
- Interoperabilitätstest, 48
- intuitive Testfallermittlung, 27
- Intuitive Testfallermittlung, 40
- Klassifikationsbaum, 27, 35, 36
- Kompatibilität, 45
- konkreter Testfall, 11, 15, 16
- Normen
 - DO-178C, 19
 - ED-12C, 19
- N-switch, 34
- N-Switch-Überdeckung, 34
- Operabilität, 45
- paarweises Testen, 27, 36, 37
- Produktrisiko, 14, 22
- Qualitätsmerkmale, 46
- Qualitätsuntermerkmale, 46
- risikobasierte Teststrategie, 19
- risikobasiertes Testen, 22
- Risikobewertung, 24
- Risikoidentifizierung, 22, 23
- Risikominderung, 22, 24
- Risikostufe, 24
- Schadensausmaß des Risikos, 24
- Schlüsselwörter, 58
- SDLC, 12
- sequentielles Entwicklungsmodell, 42
- Softwareentwicklungslebenszyklus, 12
- Softwarelebenszyklusmodell
 - agil, 13, 18
 - iterativ, 13
 - sequentiell, 13
- Softwarequalitätsmerkmale testen, 45
- Standard ISO 25010, 18
- Standards
 - ISO 25010, 64
 - OMG-DMN, 31
 - OMG-UML, 38
- SUMI, 45
- Test, 11
- Testablaufspezifikation, 11
- Testanalyse, 11, 14
- Testausführungsplan, 11
- Testausführungswerkzeug, 60
- Testbasis, 17
- Testbedingung, 11
- Testbedingungen, 14
- Test-Charta, 20, 27, 42
- Testdaten, 11
- Testdurchführung, 11, 20

| | |
|---|---|
| Testen der funktionalen Angemessenheit, 47 | Testsuite, 11, 19 |
| Testen der funktionalen Korrektheit, 47 | Testumgebung, 19 |
| Testen der funktionalen Vollständigkeit, 48 | Testverfahren, 27 |
| Testen ohne Testskript, 20 | Testverfahren kombinieren, 39 |
| Testentwurf, 11, 15 | Übertragbarkeitstest, 51 |
| Testentwurfswerkzeug, 59 | User-Story, 55 |
| Testfall, 17 | WAMMI, 45 |
| Testorakel, 17 | Werkzeuge für die Testdatenvorbereitung, 59 |
| Testrealisierung, 11, 18 | Zugänglichkeit, 45 |
| Testskript, 15 | Zustandsübergangstest, 27, 33 |